Lattice Bingo: Asynchrony and One-Shot Adaptivity in Verifiable Secret Sharing via Erasure Code Commitments

Ittai Abraham[1,2], Aniket Kate[3,4], Aditi Partap[5], and Ertem Nusret Tas[1]

a16z Crypto Research
Hebrew University
Purdue University
Mysten Labs
Stanford University

Abstract. Asynchronous verifiable secret sharing (AVSS) allows a dealer to distribute a secret among parties while preserving privacy and consistent reconstruction, despite Byzantine faults and arbitrary message delays. It is a fundamental building block for Byzantine agreement, distributed key generation, and secure multiparty computation. Packed AVSS protocols share many secrets together to reduce communication per secret. The best known AVSS protocols achieve quadratic communication while sharing linearly many secrets. In particular, Bingo (Abraham et al., CRYPTO 2023) shares a batch of $\Theta(k)$ secrets among k parties using $O(\lambda k^2)$ words of communication, while also achieving adaptive security, but relies on assumptions that are not post-quantum secure. Known post-quantum alternatives have higher worst-case communication for comparable batch sizes or require substantially larger batches to achieve linear amortized communication.

We present Lattice Bingo, a lattice-based packed AVSS protocol with quadratic communication and linear packing. For every constant $\epsilon > 0$, it shares $\Theta(k)$ secrets among $k \geq (3+\epsilon)f$ parties, tolerates f Byzantine parties, and uses $O(k^2 \operatorname{poly}(\lambda))$ words of communication. It additionally achieves one-shot adaptive secrecy: the adversary may adaptively choose its corruption and reconstruction requests based on the public transcript observed so far, but must submit these requests together in a single batch.

Our main technical contribution is a commitment scheme for two-dimensional linear codes based on the PowerBASIS assumption. We combine PowerBASIS commitments with the linear code based secret sharing of Applebaum, Nir, and Pinkas, replacing the Reed–Solomon encoding and KZG commitments used in Bingo. Our commitment supports proof interpolation, enabling parties to recover missing shares and their opening proofs. It additionally achieves a notion that we call one-shot simulation hiding. Together, these properties allow us to adapt Bingo's framework to obtain Lattice Bingo with the same efficiency and one-shot adaptive security.

1 Introduction

Secret sharing allows a dealer to distribute a secret among k parties. In its basic form, the dealer is honest, and any set of at least $f+1$ parties can reconstruct the secret, while any set of f parties learns nothing about it. This primitive is a fundamental building block for Byzantine agreement, distributed key generation for threshold cryptography, and secure multiparty computation [CKS00, CKLS02, GJM+21, AJM+23, PCR15, SJK+17].

There are several extensions to secret sharing that have been proposed to provide additional guarantees. Verifiable secret sharing (VSS) [CGMA85] ensures that parties can verify the validity of their shares even in the face of a malicious dealer. Packed secret sharing [FY92] allows a dealer to share multiple secrets together more efficiently than sharing each secret separately. The asynchronous setting [BCG93, CR93] addresses the network model: messages between honest parties are eventually delivered, but may arrive in any order and after arbitrary delays. In this work, we focus on packed asynchronous verifiable secret sharing (AVSS).

The best known AVSS protocols today achieve quadratic communication while sharing linearly many secrets.[1] In particular, Bingo [AJM+23] shares $\Theta(k)$ secrets among k parties using $O(\lambda k^2)$ words of communication, where λ is the security parameter, while tolerating f adaptively corrupted parties at the optimal resilience $k \geq 3f + 1$. Here, adaptivity allows the adversary to choose which parties to corrupt as the protocol proceeds, based on the information it has learned so far. However, Bingo relies on discrete-logarithm-based cryptography and is therefore not post-quantum secure. This raises a natural question:

Can post-quantum AVSS achieve the same quadratic communication and linear packing?

As an answer to this question, we present *Lattice Bingo*, a lattice-based packed AVSS protocol that achieves quadratic communication and linear packing. For every fixed $\epsilon > 0$, it shares $\Theta(k)$ secrets with $O(k^2 \operatorname{poly}(\lambda))$ words of communication and tolerates f corrupted parties whenever $k \geq (3+\epsilon)f$. It additionally achieves *one-shot adaptive secrecy*, wherein the adversary can adaptively choose which parties to corrupt at any point during the protocol, but it has to choose all f parties in one-shot[2].

Table 1 compares Lattice Bingo with prior schemes, in terms of the total communication for sharing $\Theta(k)$ secrets, and the *minimum* batch sizes required for linear amortized communication.

Our starting point is Bingo [AJM+23], in which the dealer samples a bivariate polynomial, commits to it, and sends each party evaluations on one row together with opening proofs. To achieve Bingo's efficiency and adaptivity guarantees, the bivariate commitment must satisfy two crucial properties. *Proof interpolation* allows parties to compute values and opening proofs for all points on a row, given sufficiently many valid opening proofs in that row, even when those openings were maliciously generated. The same must hold for any column. *Simulation hiding* requires a simulator to reproduce the adversary's view, including opening proofs for arbitrary points, using only the evaluations of the polynomial at those points. Bingo constructs a bivariate polynomial commitment scheme with these properties using the hiding version of KZG [KZG10]: its homomorphism supports Lagrange interpolation in the exponent, enabling proof interpolation. On the other hand, the hiding polynomial provides randomness that the simulator can use to generate consistent openings. Obtaining these properties from lattices is challenging. Lattice opening proofs must remain short, whereas polynomial evaluation and naive interpolation can introduce large integer coefficients. The simulation argument based on the hiding polynomial also does not directly carry over.

In our construction, we replace the bivariate polynomial in Bingo with a random matrix encoded along its rows and columns using Bounded-Encoder Linear Codes, linear erasure codes whose encoder matrices have small integer entries. We adapt the secret-sharing ideas of Applebaum, Nir, and Pinkas [ANP23] to this two-dimensional structure, and design a commitment scheme for it using the PowerBASIS framework [WW23, FMN24]. This framework provides the homomorphism needed for proof interpolation, while the small entries of the encoder matrix help keep opening proofs short. We prove one-shot simulation hiding by extending the techniques of [ANP23] and using a Gaussian density argument to simulate the joint distribution of opening proofs from the disclosed values alone. These properties allow us to achieve one-shot adaptive secrecy for our AVSS protocol, as well as the same efficiency as Bingo. We give a detailed overview of our techniques in Section 2.

[1] The classical Dolev–Reischuk lower bound gives $\Omega(k^2)$ worst-case messages for deterministic Byzantine agreement when $f = \Theta(k)$ [DR82]. The reduction of Backes, Datta, and Kate [BDK12, Appendix C] shows the same lower bound for AVSS when counting both sharing and reconstruction.

[2] This notion is similar to *one-shot adaptive security* defined in the context of threshold encryption in [HASW25].

Table 1. Comparing Lattice Bingo with prior post-quantum secure AVSS protocols, with Bingo as a classical baseline.

Construction	Total communication for $O(k)$ secrets	Sufficient batch size for $O(k)$ per secret	Privacy & Reconstruction thresholds	Secrecy Notion Proven
Bingo [AJM+23] (classical)	$O(k^2)$	$O(k)$	$(\frac{1}{3}, \frac{1}{3})$	Adaptive
GoSSamer [XLL+26]	$O(k^3)$	$O(k^2)$	$(\frac{1}{3}, \frac{1}{3})$	Static
Ji et al. [JS26]	$O(k^4)$	$O(k^3)$	$(\frac{1}{3}, \frac{1}{3})$	Static
Zhang et al. [ZLG+24]	$O(k^3)$	—	$(\frac{1}{3} - O(1/k), \frac{1}{3})$	Static
Shoup and Smart [SS24]	$O(k^3)$	—	$(\frac{1}{3}, \frac{1}{3})$	Static
NI-VSS [ACF+26, PNSNA26]	$O(k^3)$	—	$(\frac{1}{3}, \frac{1}{3})$	Static
Lattice Bingo	$O(k^2)$	$O(k)$	$(\frac{1}{3} - \epsilon, \frac{1}{3} + \epsilon)$	One-shot adaptive

Secrets are field elements in the compared instantiations. We suppress polynomial factors in λ and assume $\log k < \lambda$. Communication bounds concern the worst case. — means that no batch size achieves linear amortized communication under the stated bounds. Thresholds are fractions of k, with integer rounding and the one-share offset omitted. For Zhang et al., the privacy reduction is $2q/k$, where $q = O(\lambda)$ is the FRI query count. NI stands for non-interactive.

Open Problems. We note that, while Lattice Bingo obtains most of the desired properties of Bingo, it leaves a few gaps that we pose as open questions. The first is achieving the optimal resilience $k \geq 3f+1$, closing the arbitrarily small constant loss caused by the gap between our privacy and reconstruction thresholds. The second is extending our one-shot adaptive secrecy guarantee to full adaptive security. Our guarantee allows the adversary to adaptively choose the corruptions, as long as it specifies all the corruptions in one shot. Full adaptivity would additionally allow the adversary to choose successive corruptions adaptively based on the view of the previously corrupted parties. Third, our construction currently requires a CRS of size $O(k^2)$ – we leave the task of reducing the CRS size or achieving transparent setup as another open problem.

Additionally, our commitment scheme may have applications beyond verifiable secret sharing in protocols requiring commitments to erasure-coded data, such as verifiable information dispersal and data availability sampling (see [HASW24, NNT21, BNNP25] and the references therein). We leave exploring these applications to future work.

1.1 Related Work

Asynchronous VSS. Let B denote the number of field-element secrets being shared. The closest classical benchmark is *Bingo* [AJM+23], which packs $f + 1 = \Theta(k)$ secrets using $\mathcal{O}(k^2)$ total communication in the number of secrets, and hence $\mathcal{O}(k)$ communication per secret, while tolerating adaptive corruptions. Its pairing-based construction is not post-quantum secure. Our protocol

targets the same linear amortized dependence while simultaneously providing post-quantum and (one-shot) adaptive security.

Among post-quantum candidates, GoSSamer [XLL+26] proves security only against static corruptions and communicates $\mathcal{O}(\lambda Bk + \lambda k^3 \log k)$; thus its per-secret cost becomes linear only for $B = \Omega(k^2 \log k)$ (and is $\mathcal{O}(\lambda k^2 \log k)$ when $B = \Theta(k)$). The information-theoretic construction of Ji, Li, and Song [JLS24] has communication $\mathcal{O}(Bk + k^{12})$ when security-parameter factors are suppressed, so its linear term dominates only for $B = \Omega(k^{11})$. A follow-up work by Ji and Song [JS26] reduces the k^{12} additive term to k^4[3]. Shoup and Smart [SS24] use lightweight cryptography but analyze static corruptions; on their adversarial path, communication can grow as $\mathcal{O}(\lambda Bk^2 + \lambda k^3 \log k)$, leaving quadratic amortized cost. FRISS [ZLG+24] builds AVSS from FRI-based commitments, but shares one secret with $\widetilde{\mathcal{O}}(k^2)$ network communication and does not give an adaptive-security proof. Because its opening proofs are not homomorphic, the dealer must distribute additional evaluations and proofs—a useful illustration of a central obstacle to post-quantum packing. Additionally, they do not achieve optimal resilience: secrecy holds for their protocol only for a $1/3 - \Theta(1/k)$ fraction of corruptions.

Some post-quantum asynchronous protocols instead use weaker AVSS notions [BBB+24, BJK+25a, BJK+25b]: with a faulty dealer they may output $\perp$, or may fail to leave every honest party with a usable share. These primitives are valuable for their intended beacon or MPC applications, but do not realize complete AVSS properties.

Non-interactive and publicly verifiable secret sharing. Recent lattice- or hash-based Non-interactive VSS (NI-VSS) and publicly verifiable secret sharing (PVSS) constructions include Abdolmaleki et al. [ACF+26], Pham et al. [PNSNA26], and Kate et al. [KMS+26]. For a scalar secret, the dealer publishes an $\mathcal{O}(k)$-size transcript; disseminating that transcript with asynchronous extended-message reliable broadcast yields $\mathcal{O}(k^2)$ network communication. Moreover, these works do not establish the combination of adaptive security and linear amortized AVSS considered here, and the fully post-quantum construction of Kate et al. is not additively homomorphic.

Adaptive PVSS is known in the pre-quantum setting: Bacho and Loss [BL23] obtain aggregatable adaptive PVSS in an algebraic group model, and Bacho, Chen, and Loss [BCL26] subsequently give a standard-model construction from DDH/co-CDH-type assumptions. Both remain vulnerable to quantum attacks. Class-group NI-VSS schemes [KMM+24, CD24] are likewise not post-quantum, while the hybrid construction of Gentry, Halevi, and Lyubashevsky [GHL22] uses LWE encryption but discrete-log-based Bulletproofs. For broader coverage of discrete-log-based AVSS, we refer to the survey and simplifications of Das et al. [DXT+25].

Synchronous VSS. Post-quantum or lightweight VSS is also known with synchrony assumptions: Espitau, Niot, and Prest [ENP24] study short-secret sharing and lattice DKG; Cascudo, Cozzo, and Giunta [CCG24] use symmetric-key cryptography; Baghery [Bag25] gives a framework with a post-quantum instantiation but static corruptions; and Atapoor et al. [ABN+25] give batched and packed constructions. Earlier, Atapoor et al. [ABCP23] constructed designated-verifier VSS from lightweight primitives. These protocols are complementary, but their use of synchrony does not address the asynchronous adaptive setting that we consider.

Lattice-based Commitments. Several works construct polynomial and vector commitments from lattices [WW23, FMN24, AFLN24, CMNW24, NS24, HSS24, WYZJ24], offering different trade-

[3] The additive term is k^5 across k instances of AVSS, equivalently, we are considering k^4 for a single AVSS instance.

offs between computational assumptions, setup requirements, proof size, and prover and verifier efficiency. Related work also constructs succinct functional commitments supporting more general computations [Wee25]. However, to the best of our knowledge, none of these constructions has been proven to simultaneously achieve proof interpolation and one-shot simulation hiding as defined in this work, which are the two properties needed for our AVSS construction.

2 Technical Overview

We start by reviewing Bingo [AJM+23] in Section 2.1, an adaptively secure packed AVSS scheme that serves as the starting point of our construction. In Section 2.2, we explain the difficulty with a direct lattice-based instantiation, and how we address it using a special class of linear codes that we call Bounded-Encoder Linear Codes along with secret-sharing ideas of [ANP23] as building blocks.

2.1 Starting Point: Bingo AVSS

Bingo [AJM+23] constructs an adaptively secure packed asynchronous VSS scheme. It shares up to $f+1$ secrets among $k = 3f+1$ parties, of which at most f may be faulty.

Bingo for omission failures. First, consider parties that may omit messages but otherwise follow the protocol. Due to asynchrony, the dealer can interact with only $2f+1$ parties, and since f of them may have omission failures, the remaining $f+1$ honest parties must help all honest parties to eventually receive their share of the secret. To achieve this, the dealer in Bingo samples a bivariate polynomial $\Phi(X, Y)$ of degree at most $2f$ in X and f in Y. The secrets $s_0, \ldots, s_f$ are placed at publicly specified points on a single row of the polynomial:

$$\Phi(\omega_{-j}, \omega_0) = s_j \qquad (0 \le j \le f),$$

for some fixed points $\omega_{-f}, \ldots, \omega_0$. We additionally define $\omega_1, \ldots, \omega_k$ as the points corresponding to the k parties. Visually, think of the $k \times k$ matrix of the evaluations of $\Phi(X, Y)$ whose (i, u) entry is $\Phi(\omega_u, \omega_i)$. The polynomial $\Phi(X, \omega_i)$ corresponds to the i-th row of the matrix, denoted by α_i, and the polynomial $\Phi(\omega_i, Y)$ to the i-th column of the matrix, denoted by β_i. Here, $\alpha_i(\omega_u) = \beta_u(\omega_i)$.

The dealer sends each party i (denoted by P_i) the i-th row α_i of the matrix. P_i then waits for $2f+1$ parties to acknowledge receiving their rows. This is because once $(2f+1-f) = f+1$ honest parties acknowledge, all honest parties can eventually recover their shares in the following way: first, each honest party P_j that received its row from the dealer sends $\alpha_j(\omega_u) = \Phi(\omega_u, \omega_j)$ to every P_u. Hence, each honest party P_j receives at least $f+1$ points on its j-th column β_j and is able to reconstruct it (since β_j is a degree f polynomial for any j). Once P_j reconstructs its column, it sends each P_i the value $\beta_j(\omega_i) = \Phi(\omega_i, \omega_j)$. This way, all honest parties P_i can eventually reconstruct their rows from the $2f+1$ honest shares. Each party needs to send just $O(k)$ words, and the protocol takes a constant number of rounds.

To reconstruct the j-th secret s_j, the parties send $\alpha_i(\omega_{-j})$. With $f+1$ such values, we can interpolate the degree f polynomial $\beta_{-j} = \Phi(\omega_{-j}, Y)$, and compute $\beta_{-j}(\omega_0)$ to learn the secret.

Intuitively, we can think of the bivariate polynomial as sampling a matrix of coefficients, and applying the Reed-Solomon code on the rows and columns of this matrix (with different rates). The column and row reconstruction processes in the protocol above are essentially Reed-Solomon decoding given enough points. By the end of the protocol, each party receives the i-th rows and columns

of the encoded matrix. Looking ahead, this coding-theoretic view will be useful for understanding our construction.

Bingo for malicious failures. To move from omission failures to malicious failures with adaptive security, Bingo relies on a commitment scheme for bivariate polynomials: the dealer uses this to commit to the matrix and broadcasts the commitment to everyone. Each party then accepts a message, i.e. a row from the dealer or a point from another party, only if it is accompanied with valid opening proofs with respect to its commitment. This essentially forces the malicious parties to act as if they can only have omission failures.

For this to work, Bingo requires the bivariate commitment scheme to have the following properties:

1. *Efficiency.* The total commitment size is linear in k, and the individual opening proofs have size logarithmic in k.
2. *Evaluation binding.* It should be hard to open the same commitment to two different values at the same point.
3. *Proof interpolation.* From any $f+1$ valid evaluations and opening proofs in a column, we can generate opening proofs at any other point of that column. Similarly, any $2f+1$ valid row opening proofs suffice to generate all the other opening proofs for that row. This interpolation must work even when the original openings were produced by a malicious party, e.g., a malicious dealer. Essentially, this implies some level of homomorphism among the opening proofs of the commitment scheme.
4. *Simulation hiding.* To prove secrecy, Bingo requires the commitment scheme to satisfy a notion called simulation hiding. Namely, there must exist a simulator that can produce the public commitment before learning the secrets. Later on, when the adversary corrupts parties, the simulator must be able to produce opening proofs for those parties that are consistent with the commitment, using only the rows and columns revealed to the adversary. Bingo shows that an adaptive adversary cannot learn anything about the unreconstructed secrets if the commitment scheme supports such a simulator.

Bingo uses the hiding version of KZG commitments [KZG10] as the key building block to construct such a bivariate polynomial commitment scheme. Writing $\Phi(X,Y) = \sum_{b=0}^{f} \Phi_b(X)Y^b$, it commits separately to the $f+1$ polynomials $\Phi_b(X)$, each of degree at most $2f$, using KZG. The homomorphism of KZG allows interpolating the required proofs in the exponent. Moreover, the random hiding polynomial helps achieve simulation hiding.

2.2 Building Lattice Bingo

We start with describing the PowerBASIS assumption [FMN24, WW23], which our constructions rely on. We work over $\mathbb{F}_q = \mathbb{Z}_q$ for an odd prime q.

The PowerBASIS assumption. As in the (Module-)SIS problem, the adversary needs to find a non-zero vector $\boldsymbol{s}$ of small norm such that $\boldsymbol{A}\boldsymbol{s} = 0$ for a uniformly random matrix $\boldsymbol{A}$. However, in the PowerBASIS setting, the adversary is given additional information, namely a trapdoor for a related matrix. In particular, the challenger samples $\boldsymbol{A}_\star \in \mathbb{F}_q^{(n-1)\times m}$ and $\boldsymbol{a} \in \mathbb{F}_q^m$ uniformly, and sets

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}^\top \\ \boldsymbol{A}_\star \end{bmatrix}.$$

It also samples an invertible square matrix $\boldsymbol{W} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times n}$. It then computes:

$$\boldsymbol{B}_r = \left[\begin{array}{cccc|c} \boldsymbol{A} & 0 & \cdots & 0 & -\boldsymbol{G} \\ 0 & \boldsymbol{W}\boldsymbol{A} & \cdots & 0 & -\boldsymbol{G} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{W}^{r-1}\boldsymbol{A} & -\boldsymbol{G} \end{array}\right] \in \mathbb{Z}_q^{rn\times(rm+d_g)}.$$

where $d = \lceil \log_2 q \rceil$, $d_g = nd$, $\boldsymbol{G} = \boldsymbol{I}_n \otimes (1, 2, \ldots, 2^{d-1})$ is the gadget matrix, and r is a parameter.

Then, the adversary is given the matrices $(\boldsymbol{A}, \boldsymbol{W})$ as well as a trapdoor $\boldsymbol{T}_r$ for $\boldsymbol{B}_r$. This trapdoor can be used to sample short preimages $\boldsymbol{s}$ from $\boldsymbol{B}_r^{-1}(\boldsymbol{t})$ for any $\boldsymbol{t} \in \mathbb{Z}_q^{rn}$. The PowerBASIS assumption states that it is hard for the adversary to solve SIS with respect to $\boldsymbol{A}_\star$ even if it is given all of this additional information.

2.2.1 A (failed) first attempt As a starting point, we try to instantiate the Bingo AVSS framework by replacing KZG with an existing lattice-based (univariate) polynomial commitment scheme. Commitment schemes from [FMN24, WW23] based on PowerBASIS are a natural candidate since they exhibit homomorphism in their opening proofs. We now describe this commitment scheme.

To commit to (univariate) polynomials of degree $r-1$, the setup algorithm generates a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ along with a trapdoor $\boldsymbol{T}_{\boldsymbol{A}}$. It then samples a uniformly random invertible matrix $\boldsymbol{W} \in \mathbb{Z}_q^{n\times n}$, computes $\boldsymbol{B}_r$ as described above and uses $\boldsymbol{T}_{\boldsymbol{A}}$ to sample a trapdoor $\boldsymbol{T}_r$ for $\boldsymbol{B}$. The public parameters then contain $(\boldsymbol{A}, \boldsymbol{W}, \boldsymbol{T}_r)$ (precisely the information given to a PowerBASIS adversary).

In order to commit to a polynomial $h(X) = \sum_{a=0}^{r-1} h_a X^a$, we use the trapdoor $\boldsymbol{T}_r$ to sample short vectors $\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{r-1}$ and $\widehat{\boldsymbol{c}}$ as follows:

$$\begin{bmatrix} \boldsymbol{s}_0 \\ \vdots \\ \boldsymbol{s}_{r-1} \\ \widehat{\boldsymbol{c}} \end{bmatrix} \leftarrow\!\!\$\ \boldsymbol{B}_r^{-1}\left(\begin{bmatrix} -h_0\boldsymbol{W}^0\mathbf{e}_1 \\ -h_1\boldsymbol{W}^1\mathbf{e}_1 \\ \vdots \\ -h_{r-1}\boldsymbol{W}^{r-1}\mathbf{e}_1 \end{bmatrix}\right),$$

where $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^\top \in \mathbb{Z}_q^n$. The commitment is defined as $\boldsymbol{c} = \boldsymbol{G}\widehat{\boldsymbol{c}}$, and we refer to $\boldsymbol{s}_i$ as the i-th coefficient opening. Observe that

$$\boldsymbol{A}\boldsymbol{s}_a + h_a\mathbf{e}_1 = \boldsymbol{W}^{-a}\boldsymbol{c} \ \wedge\ \boldsymbol{s}_a \text{ is short } \forall\ 0 \le a < r$$

Next, to open this commitment to any point ω_i, we take advantage of the linearity of the above equations. We can multiply the a-th equation above by ω_i^a and add them up to obtain

$$\boldsymbol{A}(\Sigma_{0\le a<r}\omega_i^a\boldsymbol{s}_a) + h(\omega_i)\mathbf{e}_1 = (\Sigma_{0\le a<r}\omega_i^a\boldsymbol{W}^{-a})\boldsymbol{c}$$

Hence, we can define the opening proof at ω_i as $\boldsymbol{\pi}_i = \sum_{a=0}^{r-1}\omega_i^a\boldsymbol{s}_a$, and use the equation above to verify it.

Proof interpolation. Linearity of the opening proofs enable us to view the proofs $\boldsymbol{\pi}_i$ as 'evaluations' of a polynomial with $\{\boldsymbol{s}_i\}$ as the 'coefficients'. Hence, we can interpolate any r evaluations $\{h(\omega_i)\}$ and opening proofs $\{\boldsymbol{\pi}_i\}$ to get $h_0, \ldots, h_{r-1}$ as well as the coefficient openings $\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{r-1}$. Once we derive these, we can compute the opening proof for any point ω_j as $\boldsymbol{\pi}_j = \sum_{a=0}^{r-1}\omega_j^a\boldsymbol{s}_a$.

The problem with evaluation binding. Unfortunately, the above scheme does not achieve evaluation binding. The reason is that, although the coefficient openings themselves are short, the opening proofs derived by taking a linear combination may not be short, since powers of ω_i can have size comparable to q. The resulting proof $\boldsymbol{\pi}_i$ may have a large norm, and hence, we cannot reduce evaluation binding to PowerBASIS.

In more detail, suppose that two accepted proofs $\boldsymbol{\pi}_i, \boldsymbol{\pi}'_i$ open the same commitment at the same point to distinct values v_i, v'_i. Subtracting their verification equations gives

$$\boldsymbol{A}(\boldsymbol{\pi}_i - \boldsymbol{\pi}'_i) = (v'_i - v_i)\boldsymbol{e}_1 \pmod q.$$

Deleting the first row yields $\boldsymbol{A}_\star(\boldsymbol{\pi}_i - \boldsymbol{\pi}'_i) = 0 \pmod q$. Thus, $\boldsymbol{\pi}_i - \boldsymbol{\pi}'_i$ would be a valid PowerBASIS solution, but only if both $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}'_i$ are short. However, if we require the proofs to be short, we would lose correctness since even honest proofs would not verify.

The key issue. The key problem here is that the encoding matrix of Reed–Solomon code (used to encode rows and columns in Bingo) is a Vandermonde matrix with *large* entries that are powers of evaluation points. One might consider instantiating this commitment over a ring $\mathbb{Z}_q[\zeta]/(\zeta^N+1)$ and using monomials ζ^j as evaluation points, since all their powers are signed monomials of coefficient norm one. But there are only $2N$ signed monomials, so we would need to set $N = O(k)$ to have k distinct points to assign to the k rows and columns. The ring dimension, and consequently the size of ring elements in the commitment, would then grow with the number of parties, making the resulting scheme inefficient.

2.2.2 Solution: Replacing Reed-Solomon with Bounded-Encoder Linear Codes

To get around the above issue, we replace Reed-Solomon codes with a special type of linear code called Bounded-Encoder Linear Code. Such a code has a linear encoder over $\mathbb{F}_q$ represented by a matrix with small integer entries, and admits linear decoding from any sufficiently large set of codeword coordinates (note that we do not require the decoder matrix to have small entries). Applebaum et al. [ANP23] showed how to use such codes to construct secret sharing schemes, analogous to the use of Reed–Solomon codes in Shamir secret sharing. Below, we first review their secret sharing scheme, before describing how we build our plausibly post-quantum secure Bingo variant using Bounded-Encoder Linear Codes. We end the section with a concrete construction of Bounded-Encoder Linear Codes that can be used to instantiate our scheme.

Starting Point: Secret Sharing from Bounded-Encoder Linear Codes using [ANP23]. Let $\boldsymbol{G}_{\mathsf{code}} \in \mathbb{Z}^{k\times r}$ denote the encoding matrix of a bounded-encoder linear code, where the message length is r and the codeword length is k. Let $t_c \geq r$ denote the minimum number of codeword symbols that enable decoding with high probability. Then, for every set $I \subseteq [k]$ of size t_c, there is a decoding matrix $\boldsymbol{D}(I) \in \mathbb{Z}_q^{r\times t_c}$ such that

$$\boldsymbol{D}(I)(\boldsymbol{G}_{\mathsf{code}})_I = \boldsymbol{I}_r \qquad \text{over } \mathbb{F}_q.$$

To construct a secret sharing scheme from such a code, we first sample a vector $\boldsymbol{a}$ uniformly at random from $\{0,\ldots,D-1\}^r$ for some small D, and publish it as part of public parameters. Then, to share a secret $s \in \mathbb{F}_q$, the dealer samples a random message $\boldsymbol{m} \in \mathbb{F}_q^r$, computes its encoding $\boldsymbol{v} = \boldsymbol{G}_{\mathsf{code}}\boldsymbol{m}$, and gives party P_i, $v_i = \boldsymbol{g}_i^\top \boldsymbol{m}$ as its share. Additionally, it outputs $z = s + \boldsymbol{a}^\top \boldsymbol{m}$ as a *public share* available to everyone. Here $\boldsymbol{g}_i^\top$ is the i-th row of $\boldsymbol{G}_{\mathsf{code}}$. For correctness, observe that

any t_c parties can recover $\boldsymbol{m}$ by running the decoding algorithm on their codeword symbols, and then output $z - \boldsymbol{a}^\top \boldsymbol{m}$ as the reconstructed secret.

For privacy, we need to prove that the mask $\boldsymbol{a}^\top \boldsymbol{m}$ remains uniform even after a corrupted set of parties J of size t_p learns its shares $(\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{m}$. For this purpose, [ANP23] showed that, with an appropriate gap between the privacy threshold t_p and the message dimension r, a randomly sampled $\boldsymbol{a}$ remains outside the row-span of $(\boldsymbol{G}_{\mathsf{code}})_J$ over $\mathbb{F}_q$. This then implies that the mask indeed looks uniformly random to any set of up to t_p parties. Note that this is a ramp scheme: the number of shares against which privacy is guaranteed is smaller than the number required for reconstruction. However, the gap between these thresholds can be made arbitrarily small for our concrete instantiation, as we will see later in this section.

Bingo with Bounded-Encoder Linear Codes. We now explain how to extend the masking idea above to get the two-dimensional sharing structure required by Bingo. Recall that Bingo uses Reed-Solomon codes on both the rows and columns. Similarly, we will apply a bounded-encoder linear code on both the rows and columns (but again with different rates). Let $\boldsymbol{G}_{\mathsf{code}} \in \mathbb{Z}^{k \times r}$ and $\boldsymbol{H}_{\mathsf{code}} \in \mathbb{Z}^{k \times c}$ denote the encoder matrices for two bounded-encoder linear codes. Define t_c^G and t_c^H as their respective reconstruction thresholds, and $\boldsymbol{g}_i^\top$ and $\boldsymbol{h}_i^\top$ as the i-th rows of the matrices. The parameters of this scheme consist of r, c, t_c^G, t_c^H; we describe how to set these at the end of this section.

The dealer samples $\boldsymbol{M} \in \mathbb{F}_q^{r \times c}$ uniformly at random as in the case of the bivariate polynomial in the original Bingo scheme. It is then encoded as $\boldsymbol{X} = \boldsymbol{G}_{\mathsf{code}} \boldsymbol{M} \boldsymbol{H}_{\mathsf{code}}^\top$. We define $\boldsymbol{\alpha}_i = \boldsymbol{g}_i^\top \boldsymbol{M}$ as the i-th row and $\boldsymbol{\beta}_i = \boldsymbol{M} \boldsymbol{h}_i$ as the i-th column of the encoded matrix. Their intersections satisfy $X_{i,u} = \boldsymbol{\alpha}_i \boldsymbol{h}_u = \boldsymbol{g}_i^\top \boldsymbol{\beta}_u$. Additionally, Setup samples short public vectors $\boldsymbol{b}_0 \in \mathbb{Z}^r$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_\ell \in \mathbb{Z}^c$ with all entries less than a parameter D. These will be used for masking the secrets. In more detail, to share secrets $s_1, \ldots, s_\ell$, where $\ell = O(k)$ is a parameter, the dealer publishes

$$z_t = s_t + \boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_t \qquad (t \in [\ell]).$$

These are the two-dimensional counterparts of the public masked value z above. The parameters of the scheme are chosen so that the masks of the unreconstructed secrets remain jointly uniform given the rows and columns of the f corrupt parties as well as the mask values of the secrets that have already been reconstructed.

The rest of the protocol follows the same steps as Bingo: the dealer privately sends the i-th row $\boldsymbol{\alpha}_i$ to P_i. Each P_i then sends $\boldsymbol{\alpha}_i \boldsymbol{h}_u = \boldsymbol{g}_i^\top \boldsymbol{\beta}_u$ to party u. Once party u receives t_c^G such values, say $(\boldsymbol{g}_{i_1}^\top \boldsymbol{\beta}_u, \ldots, \boldsymbol{g}_{i_{t_c^G}}^\top \boldsymbol{\beta}_u)$, it can run the decoding algorithm of the $\boldsymbol{G}_{\mathsf{code}}$ code to recover its column $\boldsymbol{\beta}_u$. Formally,

$$\boldsymbol{\beta}_u = \boldsymbol{D}_G(I) \big(\boldsymbol{g}_{i_1}^\top \boldsymbol{\beta}_u, \ldots, \boldsymbol{g}_{i_{t_c^G}}^\top \boldsymbol{\beta}_u\big)^\top,$$

Then, each party u sends $\boldsymbol{g}_i^\top \boldsymbol{\beta}_u$ to each P_i. Any t_c^H such values can be used to recover the i-th row $\boldsymbol{\alpha}_i$ via the decoding algorithm corresponding to $\boldsymbol{H}_{\mathsf{code}}$.

To reconstruct the t-th secret s_t, each party sends $\boldsymbol{\alpha}_i \boldsymbol{a}_t = \boldsymbol{g}_i^\top \boldsymbol{M} \boldsymbol{a}_t$. These values form the codeword $\boldsymbol{G}_{\mathsf{code}} \boldsymbol{M} \boldsymbol{a}_t$. Hence, given any t_c^G such values, we can decode to recover $\boldsymbol{M} \boldsymbol{a}_t$, from which the parties compute the mask $\boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_t$ and subtract it from z_t to get s_t.

For achieving correctness and termination, we do require the codes to satisfy certain constraints on the reconstruction thresholds t_c^G and t_c^H. We will describe these constraints and our code instantiations that satisfy these constraints at the end of this section.

Key building block: a two-dimensional code commitment. Recall that, to achieve security against malicious failures, we need a way to commit to the dealer's matrix. In particular, we must commit to a message matrix $\boldsymbol{M}$ such that we can open the commitment to any position of the matrix encoding, i.e., $\boldsymbol{X} = \boldsymbol{G}_{\mathsf{code}} \boldsymbol{M} \boldsymbol{H}_{\mathsf{code}}^{\top}$, as well as the mask values $\{\boldsymbol{b}_0^{\top} \boldsymbol{M} \boldsymbol{a}_t\}$. We refer to this as an erasure code commitment[4] for bounded-encoder linear codes, or a *Bounded-Encoder Linear Code* commitment. As in Bingo, we need this commitment scheme to satisfy the same four properties: efficiency, evaluation binding, proof interpolation in both directions, and simulation hiding.

Below, we first construct a one-dimensional code commitment scheme via PowerBASIS, and then extend it to a two-dimensional scheme.

Our one-dimensional commitment. We use the same PowerBASIS sampling procedure as in the first attempt, with the message $\boldsymbol{m} = (m_0, \ldots, m_{r-1})$ in place of polynomial coefficients. The public parameters contain $(\boldsymbol{A}, \boldsymbol{W}, \boldsymbol{T}_r)$. To commit to a message $\boldsymbol{m} = (m_0, \ldots, m_{r-1})$, we use the trapdoor $\boldsymbol{T}_r$ to sample short vectors $\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{r-1}$ and $\widehat{\boldsymbol{c}}$ as follows:

$$\begin{bmatrix} \boldsymbol{s}_0 \\ \vdots \\ \boldsymbol{s}_{r-1} \\ \widehat{\boldsymbol{c}} \end{bmatrix} \leftarrow\!\$\; \boldsymbol{B}_r^{-1} \left(\begin{bmatrix} -m_0 \boldsymbol{W}^0 \mathbf{e}_1 \\ -m_1 \boldsymbol{W}^1 \mathbf{e}_1 \\ \vdots \\ -m_{r-1} \boldsymbol{W}^{r-1} \mathbf{e}_1 \end{bmatrix} \right).$$

As before, the commitment becomes $\boldsymbol{c} = \boldsymbol{G}\widehat{\boldsymbol{c}}$, and we refer to $\boldsymbol{s}_i$ as the i-th message opening. Therefore,

$$\boldsymbol{A}\boldsymbol{s}_a + m_a \boldsymbol{e}_1 = \boldsymbol{W}^{-a} \boldsymbol{c} \pmod{q}, \qquad \|\boldsymbol{s}_a\|_2 \le \beta_s,$$

where β_s bounds the norm of the short message openings. To open position i of the codeword $\boldsymbol{G}_{\mathsf{code}} \boldsymbol{m}$, set

$$v_i = \boldsymbol{g}_i^{\top} \boldsymbol{m}, \qquad \boldsymbol{\pi}_i = \sum_{a=0}^{r-1} g_{i,a} \boldsymbol{s}_a.$$

Multiplying and adding the message-opening equations gives

$$\boldsymbol{A}\boldsymbol{\pi}_i + v_i \boldsymbol{e}_1 = \left(\sum_{a=0}^{r-1} g_{i,a} \boldsymbol{W}^{-a} \right) \boldsymbol{c} \pmod{q}. \tag{1}$$

This time however, the norm of the opening proof remains bounded since the encoder matrix $\boldsymbol{G}_{\mathsf{code}}$ has bounded entries: $\|\boldsymbol{\pi}_i\|_2 \le \|\boldsymbol{g}_i\|_1 \beta_s$. Hence, to verify this opening proof, we check the above equation and norm bound.

Intuitively, if we think of the message openings $\{\boldsymbol{s}_a\}$ as the 'message', the opening proofs $\{\boldsymbol{\pi}_i\}$ are its encoding under $\boldsymbol{G}_{\mathsf{code}}$, applied separately to each coordinate of these vectors. This view will be useful in achieving proof interpolation later on.

From one dimension to two. We now use this construction to commit to the matrix $\boldsymbol{M} \in \mathbb{F}_q^{r \times c}$. To do so, we create a separate (one-dimensional) commitment $\boldsymbol{c}_b$ to each column b of the matrix, using the same public parameters. Formally, we obtain a matrix of message openings $\{\boldsymbol{s}_{a,b}\}$ that satisfy:

$$\boldsymbol{A}\boldsymbol{s}_{a,b} + M_{a,b} \boldsymbol{e}_1 = \boldsymbol{W}^{-a} \boldsymbol{c}_b \pmod{q}, \qquad \|\boldsymbol{s}_{a,b}\|_2 \le \beta_s.$$

[4] The notion of erasure code commitments for arbitrary codes was formally introduced in [HASW24]. We focus on constructing such a commitment scheme for a special family of codes i.e. bounded-encoder linear codes.

The commitment consists of the c commitments $(\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{c-1})$. To open the committed matrix at the encoding position (i, j), we return

$$y_{i,j} = \boldsymbol{g}_i^\top \boldsymbol{M} \boldsymbol{h}_j, \qquad \boldsymbol{\pi}_{i,j} = \sum_{a=0}^{r-1} \sum_{b=0}^{c-1} g_{i,a} h_{j,b} \boldsymbol{s}_{a,b}.$$

The verification equation is a generalization of (1) to two dimensions:

$$\boldsymbol{A}\boldsymbol{\pi}_{i,j} + y_{i,j}\boldsymbol{e}_1 = \left(\sum_{a=0}^{r-1} g_{i,a} \boldsymbol{W}^{-a}\right)\left(\sum_{b=0}^{c-1} h_{j,b} \boldsymbol{c}_b\right) \pmod q,$$
$$\|\boldsymbol{\pi}_{i,j}\|_2 \le B(i,j) := \|\boldsymbol{g}_i\|_1 \|\boldsymbol{h}_j\|_1 \beta_s.$$

The same equations can also be used to authenticate the values used to reconstruct the secrets, as follows. Recall that the public masked values are $z_t = s_t + \boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_t$. We extend the position notation by setting $\boldsymbol{g}_0 = \boldsymbol{b}_0$ and $\boldsymbol{h}_{1-t} = \boldsymbol{a}_t$ for $t \in [\ell]$. An opening at $(i, 1-t)$ then authenticates $\boldsymbol{g}_i^\top \boldsymbol{M} \boldsymbol{a}_t = \boldsymbol{\alpha}_i \boldsymbol{a}_t$, which is precisely the value sent by party i when reconstructing secret t.

Efficiency. There are c individual commitments, each of which have size independent of r. Since $c = O(k)$, this means that the overall commitment size is linear in the number of parties. Additionally, the opening proofs are still a single vector in $\mathbb{Z}^m$, and hence have size independent of k.

Evaluation binding. We can reduce evaluation binding for our scheme directly to the PowerBASIS assumption. Suppose an adversary gives two accepted openings at the same position (i, j) for distinct values y and y'. Subtracting their verification equations gives

$$\boldsymbol{A}(\boldsymbol{\pi} - \boldsymbol{\pi}') = (y' - y)\boldsymbol{e}_1 \pmod q, \qquad \boldsymbol{A}_\star(\boldsymbol{\pi} - \boldsymbol{\pi}') = 0 \pmod q.$$

(Here $\boldsymbol{A}_\star$ includes all but the first row of $\boldsymbol{A}$.) The difference $\boldsymbol{\pi} - \boldsymbol{\pi}'$ must be nonzero, since otherwise the first equation would imply $y = y'$. Moreover, both proofs pass the norm check, so $\|\boldsymbol{\pi} - \boldsymbol{\pi}'\|_2 \le 2B(i,j) \le 2B$ (here $B = \max_{i,j}(B(i,j))$). This breaks the PowerBASIS assumption for $\boldsymbol{A}_\star$.

Proof interpolation. For our protocol to work, we must be able to derive the values and opening proofs for all positions on a column, given the values and opening proofs for any t_c^G points in that column. Similarly, we require this from any t_c^H points on a row.

The linearity of the opening proofs can be used to achieve this. For a fixed row i, define $\boldsymbol{t}_{i,b} = \sum_{a=0}^{r-1} g_{i,a} \boldsymbol{s}_{a,b}$. The proofs for this row then satisfy

$$\boldsymbol{\pi}_{i,j} = \sum_{b=0}^{c-1} h_{j,b} \boldsymbol{t}_{i,b} \;\forall\; j \in [k].$$

Thus, in each coordinate, the proofs form a 'codeword' of $\boldsymbol{H}_{\mathsf{code}}$ with $\{\boldsymbol{t}_{i,b}\}$ as the 'message'. Hence, given proofs at any t_c^H positions in this row, we can apply the decoder algorithm to recover $\{\boldsymbol{t}_{i,b}\}_b$ modulo q. We can then compute the opening proof at any position by using the equation above. Similarly, the opening proofs in a fixed column j can be viewed as a $\boldsymbol{G}_{\mathsf{code}}$-encoding of $\{\sum_b h_{j,b} \boldsymbol{s}_{a,b}\}_a$. Any t_c^G proofs in that column therefore suffice to recover the proof at another position.

But the above is not sufficient: as in Bingo, we require interpolation to work even for maliciously generated proofs. For such proofs, however, verification only tells us that each opening proof has

norm at most B and satisfies its equation. If we try to estimate the norm of the opening proof for (i', j) reconstructed using t_c^G column proofs as above, we get,

$$\left| \sum_{i \in I} \left(\boldsymbol{g}_{i'}^\top \boldsymbol{D}_G(I) \right)_i \boldsymbol{\pi}_{i,j} \right|_2 \leq |\boldsymbol{g}_{i'}^\top \boldsymbol{D}_G(I)|_1 B.$$

(Here $I \subseteq [k]$ denotes the set of t_c^G opening proofs used for interpolation.) Even if the coefficients of the decoding matrix $\boldsymbol{D}_G(I)$ were polynomially bounded, this can still be much larger than B. Moreover, if we now try to run interpolation again using these reconstructed proofs as inputs, the norm grows even further.

To avoid this issue, we include with our commitment a zero-knowledge proof of knowledge π_{wf} of $\{\boldsymbol{s}_{a,b}\}$ for a *simple SIS relation*:

$$\|\boldsymbol{s}_{a,b}\|_2 \leq \beta_s, \qquad \boldsymbol{A}_\star \boldsymbol{s}_{a,b} = (\boldsymbol{W}^{-a} \boldsymbol{c}_b)_\star \pmod q \quad (0 \leq a < r,\ 0 \leq b < c).$$

(The subscript $\star$ denotes deletion of the first coordinate or row.) The verifier now checks this well-formedness proof alongside the opening proof $\boldsymbol{\pi}_{i,j}$ as before.

Roughly speaking, with such a well-formedness proof, we can guarantee that any valid opening proof must have been generated from a common set of short message openings $\{\boldsymbol{s}_{a,b}\}$. In more detail, we can extract the message openings $\{\boldsymbol{s}_{a,b}\}$ from π_{wf}, and if the adversary produces an opening proof for some position (i, j) that is valid but does not equal the linear combination of the extracted openings, then, we can subtract the two to break the PowerBASIS assumption.

With the above change, we can now apply the decoding algorithm to any set of t_c^G accepted proofs, even if they were maliciously generated. Decoding can recover the common short message openings for those rows/columns, and re-encoding produces an opening proof of norm at most $\|\boldsymbol{g}_{i'}\|_1 \|\boldsymbol{h}_j\|_1 \beta_s$. The norm bound now depends only on the target position, even if the source proofs themselves were reconstructed earlier.

Simulation hiding. Recall that to prove secrecy of the Bingo protocol, we require the commitment scheme to achieve simulation hiding. Namely, there must exist an efficient simulator that can produce the dealer's messages, i.e., the commitment, without learning the secrets or the matrix used by the dealer. Later on, the adversary may corrupt a party i, at which point the simulator must produce the opening proofs for the i-th row and column, using knowledge of only the values of this row $\boldsymbol{\alpha}_i$ and column $\boldsymbol{\beta}_i$ (along with the rows and columns of the parties corrupted so far). Additionally, the adversary may ask to reconstruct a secret s_t. For such a query, the simulator must produce the opening proofs of the honest parties' messages for this secret, i.e., $\boldsymbol{\alpha}_i \boldsymbol{a}_t$, again with the knowledge of only these values along with the revealed rows and columns. We consider a *one-shot adaptive* notion of simulation-hiding, wherein, after seeing the dealer's messages, the adversary adaptively chooses the parties it wants to corrupt and the secrets it wants to reconstruct, but it must output all this information in one-shot. Looking ahead, this will allow us to prove *one-shot adaptive secrecy* for the resulting Bingo protocol which is defined in the same manner.

As a first attempt, the simulator can sample the public parameters $(\boldsymbol{A}, \boldsymbol{W}, \boldsymbol{T}_r)$ but keep the trapdoor for $\boldsymbol{A}$. It can then independently sample all the opening proofs from a Gaussian distribution such that they satisfy the verification equations. In more detail, for any position (i, j) for which we need an opening proof, we sample $\boldsymbol{\pi}_{i,j} \leftarrow\!\!\$\ \boldsymbol{A}^{-1}(\left(\sum_{a=0}^{r-1} g_{i,a} \boldsymbol{W}^{-a}\right) \left(\sum_{b=0}^{c-1} h_{j,b} \boldsymbol{c}_b\right) - y_{i,j} \boldsymbol{e}_1)$. However, independently sampled opening proofs do not match the distribution of the real opening

proofs. In a real execution, all proofs are linear combinations of the same message openings, so their distributions are correlated. We therefore need to simulate their joint distribution directly.

Our simulator samples the public parameters and keeps the trapdoor for $\boldsymbol{A}$. It samples the commitment $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{c-1}$ as uniformly random vectors. This is indeed indistinguishable from the real commitment due to the properties of discrete Gaussian distributions, as proven in [WW23]. It then simulates the well-formedness proof π_{wf} as well.

Let $J \subseteq [k]$ denote the corrupted set of f parties. For simplicity, suppose the adversary does not query any reconstructions in its one-shot adaptive query. Then, the simulator receives as input the rows and columns for all parties in J:

$$(\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{M},\ \boldsymbol{M}(\boldsymbol{H}_{\mathsf{code}})_J^\top$$

We now need to simulate the joint distribution for the opening proofs of all these values. For this purpose, the simulator samples a matrix $\boldsymbol{M}^\star$ uniformly at random, but conditioned on its corrupt rows and columns being equal to the real values:

$$(\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{M}^\star = (\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{M},\ \boldsymbol{M}^\star(\boldsymbol{H}_{\mathsf{code}})_J^\top = \boldsymbol{M}(\boldsymbol{H}_{\mathsf{code}})_J^\top.$$

Then, it samples all the message openings for $\boldsymbol{M}^\star$ using the trapdoor for $\boldsymbol{A}$:

$$\boldsymbol{s}^\star_{a,b} = \boldsymbol{A}^{-1}(\boldsymbol{W}^{-a}\boldsymbol{c}_b - M^\star_{a,b}\mathbf{e}_1)\ \forall\ 0 \le a < r, 0 \le b < c.$$

It uses these openings to compute the opening proofs for all the required points, as in the real scheme: for any i, j belonging to the corrupt rows or columns, $\boldsymbol{\pi}^\star_{i,j} = \sum_{a=0}^{r-1}\sum_{b=0}^{c-1} g_{i,a}h_{j,b}\boldsymbol{s}^\star_{a,b}$.

We next explain why the simulation matches the real distribution from the adversary's view. The main property we use is a generalization of [ANP23, Claim 4.1]: for every possible corrupt set J of size f and every secret index $t \in [\ell]$, there exist short integer vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ such that

$$\begin{gathered}(\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{u} = 0, \qquad (\boldsymbol{H}_{\mathsf{code}})_J \boldsymbol{v} = 0, \qquad \boldsymbol{a}_{t'}^\top \boldsymbol{v} = 0 \quad \forall\ t' \neq t, \\ \boldsymbol{b}_0^\top \boldsymbol{u} \neq 0 \pmod q, \qquad \boldsymbol{a}_t^\top \boldsymbol{v} \neq 0 \pmod q.\end{gathered} \tag{2}$$

The zero equalities hold over the integers. Our parameter choices ensure that this property holds simultaneously for all such J and t, except with negligible probability over setup. We now make the following observations to analyze the distribution of the matrix and opening proofs sampled by the simulator:

(i) The scalar values. Suppose $\boldsymbol{M}$ is the real matrix and define $\boldsymbol{M'} = \boldsymbol{M} + \delta \boldsymbol{u}\boldsymbol{v}^\top$ over $\mathbb{F}_q$ for some $\delta \in \mathbb{F}_q$. Because of the zero equalities above, the matrix $\boldsymbol{M'}$ has exactly the same corrupted rows and columns, and the same mask for every secret other than t:

$$(\boldsymbol{G}_{\mathsf{code}})\boldsymbol{M'} = (\boldsymbol{G}_{\mathsf{code}})\boldsymbol{M} + \delta(\boldsymbol{G}_{\mathsf{code}})\boldsymbol{u}\boldsymbol{v}^\top = (\boldsymbol{G}_{\mathsf{code}})\boldsymbol{M}$$

Similarly, $(\boldsymbol{H}_{\mathsf{code}})_J \boldsymbol{M'} = (\boldsymbol{H}_{\mathsf{code}})_J \boldsymbol{M}$ and $\boldsymbol{b}_0^\top \boldsymbol{M'} \boldsymbol{a}_{t'} = \boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_{t'}$ for all $t' \neq t$. But the t-th mask changes by $\delta(\boldsymbol{b}_0^\top \boldsymbol{u})(\boldsymbol{a}_t^\top \boldsymbol{v})$ which is non-zero. Hence, we can change this mask to any field value by choosing δ appropriately, while keeping the corrupted rows and columns and all other masks fixed. Since $\boldsymbol{M}$ is sampled uniformly in the real scheme, this shows that all the masks are jointly uniform conditioned on the corrupted rows and columns. The simulator samples $\boldsymbol{M}^\star$ uniformly subject to these same row and column constraints, so it matches the distribution of $\boldsymbol{M}$ from the adversary's view.

(ii) The opening proofs. Let $\boldsymbol{S_M} = (\boldsymbol{s}_{a,b})_{a,b}$ denote the Gaussian message-opening state for $\boldsymbol{M}$, under the fixed commitments $\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{c-1}$. Here $\boldsymbol{S_M}$ is an $r \times c$ array whose entries are the message opening vectors in $\mathbb{Z}^m$. In particular,

$$\boldsymbol{g}_i^\top \boldsymbol{S_M} \boldsymbol{h}_j = \sum_{a=0}^{r-1} \sum_{b=0}^{c-1} g_{i,a} h_{j,b} \boldsymbol{s}_{a,b} = \boldsymbol{\pi}_{i,j}.$$

For the matrix $\boldsymbol{M'} = \boldsymbol{M} + \delta \boldsymbol{u}\boldsymbol{v}^\top$, consider any integer vector $\boldsymbol{w}$ satisfying $\boldsymbol{A}\boldsymbol{w} = -\delta \mathbf{e}_1 \pmod q$ (it does not need to be short), and define the shifted openings

$$\widetilde{\boldsymbol{s}}_{a,b} = \boldsymbol{s}_{a,b} + u_a v_b \boldsymbol{w}.$$

These shifted openings satisfy the message-opening equations for $\boldsymbol{M'}$ under the same commitments:

$$\begin{aligned} \boldsymbol{A}\widetilde{\boldsymbol{s}}_{a,b} + M'_{a,b}\boldsymbol{e}_1 &= \boldsymbol{A}\boldsymbol{s}_{a,b} - \delta u_a v_b \boldsymbol{e}_1 + M'_{a,b}\boldsymbol{e}_1 \\ &= \boldsymbol{A}\boldsymbol{s}_{a,b} + M_{a,b}\boldsymbol{e}_1 \\ &= \boldsymbol{W}^{-a}\boldsymbol{c}_b. \end{aligned}$$

Moreover, writing $\widetilde{\boldsymbol{S}} = (\widetilde{\boldsymbol{s}}_{a,b})_{a,b}$, for every corrupt $i \in J$ or $j \in J$ we have

$$\boldsymbol{g}_i^\top \widetilde{\boldsymbol{S}} \boldsymbol{h}_j = \boldsymbol{g}_i^\top \boldsymbol{S_M} \boldsymbol{h}_j + (\boldsymbol{g}_i^\top \boldsymbol{u})(\boldsymbol{v}^\top \boldsymbol{h}_j)\boldsymbol{w} = \boldsymbol{g}_i^\top \boldsymbol{S_M} \boldsymbol{h}_j.$$

Thus, even though we changed the t-th mask by adding $\delta \boldsymbol{u}\boldsymbol{v}^\top$, we can correspondingly shift the message openings for $\boldsymbol{M}$; so that the opening proofs for all the corrupted rows and columns remain exactly unchanged!

It remains to compare the Gaussian distributions of the message-openings for $\boldsymbol{M}$ as in the real scheme vs $\boldsymbol{M'}$ as sampled by the simulator. The shifted openings above need not themselves have the correct Gaussian distribution for $\boldsymbol{M'}$. But we prove that, if the Gaussian distributions for message openings have a sufficiently large sampling width, then,

$$\left((\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{S_M},\ \boldsymbol{S_M} (\boldsymbol{H}_{\mathsf{code}})_J^\top \right) \approx_{\mathsf{stat}} \left((\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{S_{M'}},\ \boldsymbol{S_{M'}} (\boldsymbol{H}_{\mathsf{code}})_J^\top \right),$$

where each opening state is sampled from its correct Gaussian distribution under the same fixed commitments. The intuition is that, for each possible list of leaked proofs (i.e. proofs for corrupt rows and columns), changing the t-th mask changes only the value of the opening state along the hidden $\boldsymbol{w}$-direction, based on the above analysis. Then, if the sampling width is large enough, summing the Gaussian weights over these shifted hidden values gives almost the same total as the original values for $\boldsymbol{M}$. Hence, each leaked proof list has almost the same probability under the two matrices, and the leaked proof distributions are statistically close. The full proof is more involved – we defer the reader to Section 5 for more details.

Combining the arguments above, we see that both the matrix $\boldsymbol{M}^\star$ and the opening proofs sampled by the simulator indeed are statistically indistinguishable from the view of the adversary in the real scheme.

Putting everything together. The dealer in our bounded-encoder linear code-based Bingo protocol commits to the sampled random matrix $\boldsymbol{M}$ using the two-dimensional commitment scheme described above. It then reliably broadcasts this commitment and the masked secrets $(z_t)_{t \in [\ell]}$, and

privately sends each party its row with opening proofs for the row's entries. Parties follow the row and column exchanges described above, checking every received opening. When a party recovers a row or column, it uses proof interpolation of our commitment scheme to obtain the proofs needed for its outgoing messages. Each party sends a completion message after obtaining its row and column data as well as sending both sets of messages. Finally, each party completes the sharing protocol after receiving a total of $k - f$ completion messages. This gives us a lattice-based AVSS scheme for $\ell = O(k)$ secrets from any Bounded-Encoder Linear Code.

To prove that our protocol satisfies correctness, termination and one-shot adaptive secrecy, we need to instantiate it with a bounded-encoder linear code whose parameters satisfy certain constraints. Recall that a party completes sharing only after receiving $k - f$ completion messages. At least $k-2f$ of their senders are nonfaulty, so they have already obtained their rows and sent the corresponding intersection values to every party. These values must suffice for every nonfaulty party to recover its column, which requires $t_c^G \leq k - 2f$. Once all nonfaulty parties have their columns, they send the values at the intersection with the other parties' rows to help recover the missing rows. Every nonfaulty party then receives at least $k-f$ such values, so we must have $t_c^H \leq k - f$ to ensure that it can recover its row. These bounds guarantee termination: once one nonfaulty party completes sharing, every other participating nonfaulty party can eventually complete as well.

Additionally, we need both $r - f$ and $c - f - \ell + 1$ to be linear in k, so that there do indeed exist short directions as described in (2), which are crucial for the simulation-hiding proof to go through.

Fix any constant $0 < \epsilon \leq 1$, and let $\eta = \epsilon/8$. We choose

$$k = \lceil (3+\epsilon)f \rceil, \qquad r = \lceil (1+\eta)f \rceil, \qquad c = \lfloor (2-\eta)f \rfloor,$$
$$\ell = \lfloor (1-2\eta)f \rfloor, \qquad t_c^G = k - 2f, \qquad t_c^H = k - f.$$

For sufficiently large f, these choices satisfy all the inequalities above and give $r, c, \ell = \Theta(k)$.

A concrete Bounded-Encoder Linear Code. We now give a concrete construction of a bounded-encoder linear code that works for the above parameters. We describe how we set the remaining parameters of the scheme, i.e., the bound D on the entries of $\boldsymbol{b}_0$ and $\{\boldsymbol{a}_i\}$, the widths of the Gaussian distributions, and the modulus q in Section 6.2. In our design, every entry of $\boldsymbol{G}_{\mathsf{code}} \in \mathbb{Z}_q^{k \times r}$ and $\boldsymbol{H}_{\mathsf{code}} \in \mathbb{Z}_q^{k \times c}$ is sampled independently and uniformly from $\{0, \ldots, D_{\mathsf{code}} - 1\}$. Then, for a sufficiently large constant D_{code} depending on ϵ, every set of t_c^G rows of $\boldsymbol{G}_{\mathsf{code}}$ has rank r, and every set of t_c^H rows of $\boldsymbol{H}_{\mathsf{code}}$ has rank c, except with negligible probability over setup. This enables decoding a column from any t_c^G values on that column, and a row from any t_c^H values on that row.

Lattice Bingo. Instantiating our Bingo protocol and commitment scheme with the above code and parameters, we obtain LatBingo, an f-resilient packed AVSS with correctness, termination, and one-shot adaptive secrecy. It shares $\ell = \Theta(k)$ secrets among $k = (3+\epsilon)f + O(1)$ parties. Any $t_c^G/k = (1+\epsilon)/(3+\epsilon)$ fraction of the parties can reconstruct a secret. Thus, both the privacy and reconstruction thresholds approach $1/3$ from below and above respectively, as the fixed constant ϵ is made smaller.

Writing L_{wf} for the length of the well-formedness proof, the sharing protocol has a total communication of $O(k^2 + kL_{\mathsf{wf}})$ words. In particular, it uses $O(k^2)$ words when $L_{\mathsf{wf}} = O(k)$[5]. Reconstruction of one secret uses $O(k^2)$ words, and both procedures take a constant number of asynchronous rounds

[5] Since the well-formedness proof is a NIZK proof of knowledge for a simple SIS relation, we can easily obtain proofs of this size using existing lattice-based proof systems. We expand on this in Remark 2.

under their termination conditions. As an additional feature, LatBingo also allows reconstructing all ℓ secrets with $O(k^2)$ communication.

Paper outline. Section 3 recalls the lattice background, the PowerBASIS assumption, and the non-interactive arguments we rely on. Section 4 defines reliable broadcast and packed AVSS with one-shot adaptive secrecy. Section 5 defines our 2D bounded-encoder linear code commitment, gives its construction, and proves evaluation binding, proof interpolation, and one-shot adaptive simulation hiding. Section 6 presents the Lattice Bingo protocol and its security analysis (Section 6.1), followed by its instantiation with random integer encoder matrices, the choice of parameters, and the efficiency guarantees (Section 6.2).

3 Preliminaries

Throughout, $\lambda \in \mathbb{N}$ denotes the security parameter. We write $\mathsf{negl}(\lambda)$ for a negligible function and $\mathsf{poly}(\lambda)$ for an unspecified polynomially bounded function in λ. All parameters are functions of λ, and all matrix dimensions are polynomially bounded. For integers $a \le b$, let $[a,b] = \{a, a+1, \dots, b\}$ and $[a] = [1,a]$. Unless stated otherwise, q is an odd prime and $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$. We identify elements of $\mathbb{Z}_q$ with their centered representatives in $(-q/2, q/2]$ when taking norms. Vectors and matrices are written in bold. The ith standard basis vector is $\boldsymbol{e}_i$. Unless stated otherwise, $\|\cdot\|$ denotes the Euclidean norm. For a matrix, $\|\cdot\|$ denotes the Frobenius norm obtained by concatenating its columns. We write $s_1(\boldsymbol{B})$ for the spectral norm of $\boldsymbol{B}$ and $\|\widetilde{\boldsymbol{B}}\|$ for the length of the longest Gram–Schmidt vector of a basis $\boldsymbol{B}$. We write $\Delta(X,Y)$ for the statistical distance between two distributions. Short vectors, trapdoors, and discrete Gaussian samples are integer-valued; their norms are taken before reduction modulo q.

3.1 Lattices

A rank-r lattice $\Lambda \subseteq \mathbb{R}^m$ is the discrete additive subgroup generated by linearly independent vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_r \in \mathbb{R}^m$. Equivalently, it has the following representation.

$$\Lambda = \mathcal{L}(\boldsymbol{B}) := \{\boldsymbol{B}\boldsymbol{z} : \boldsymbol{z} \in \mathbb{Z}^r\}, \qquad \boldsymbol{B} = (\boldsymbol{b}_1 | \cdots | \boldsymbol{b}_r).$$

An integer lattice is a lattice contained in $\mathbb{Z}^m$. Its dual lattice is defined as follows.

$$\Lambda^* := \{\boldsymbol{y} \in \mathrm{span}(\Lambda) : \langle \boldsymbol{y}, \boldsymbol{x} \rangle \in \mathbb{Z} \text{ for every } \boldsymbol{x} \in \Lambda\}.$$

For $p \in [1,\infty]$, the first minimum is $\lambda_1^{(p)}(\Lambda) := \min_{\boldsymbol{0} \neq \boldsymbol{x} \in \Lambda} \|\boldsymbol{x}\|_p$. When $p = 2$, we abbreviate this quantity to $\lambda_1(\Lambda)$. Given $\boldsymbol{c} \in \mathbb{R}^m$, the corresponding lattice coset is $\Lambda + \boldsymbol{c} = \{\boldsymbol{x} + \boldsymbol{c} : \boldsymbol{x} \in \Lambda\}$.

For any integer $q \ge 2$, $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, and $\boldsymbol{u} \in \mathbb{Z}_q^n$, define the q-ary kernel lattice and its target sets as follows.

$$\begin{aligned} \Lambda_q^{\perp}(\boldsymbol{A}) &:= \{\boldsymbol{z} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \bmod q\}, \\ \Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) &:= \{\boldsymbol{z} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{z} = \boldsymbol{u} \bmod q\}. \end{aligned} \tag{3}$$

The kernel lattice is a full-rank integer lattice because it contains $q\mathbb{Z}^m$. Whenever the second set is nonempty, it is a coset of $\Lambda_q^{\perp}(\boldsymbol{A})$.

3.2 Arithmetic modulo q and integer lattices

We work with integer vectors and matrices modulo q. All modular linear algebra is over the field $\mathbb{Z}_q$. In particular, a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ has full row rank if and only if its columns span $\mathbb{Z}_q^n$. Norms of modular vectors and matrices refer to their entrywise centered integer representatives. For an integer vector $\boldsymbol{x}$, the norm $\|\boldsymbol{x}\|_\infty$ is its largest absolute coordinate.

For a scalar $a \in \mathbb{Z}$ and a vector $\boldsymbol{x} \in \mathbb{Z}^m$, we use the elementary bound

$$\|a\boldsymbol{x}\|_2 = |a|\,\|\boldsymbol{x}\|_2. \tag{4}$$

For every real matrix $\boldsymbol{S}$ and compatible vector $\boldsymbol{x}$, we also have

$$s_1(\boldsymbol{S}) \le \|\boldsymbol{S}\|, \qquad \|\boldsymbol{S}\boldsymbol{x}\|_2 \le s_1(\boldsymbol{S})\,\|\boldsymbol{x}\|_2. \tag{5}$$

These bounds apply to products of integer lifts; reduction modulo q cannot increase the Euclidean norm when centered representatives are used.

If $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ has full row rank, then the target set $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ defined in (3) is nonempty for every $\boldsymbol{u} \in \mathbb{Z}_q^n$. We write $\mathsf{GL}_n(\mathbb{Z}_q)$ for the group of invertible $n \times n$ matrices over $\mathbb{Z}_q$.

For $m \ge n$, a uniform $\boldsymbol{A} \leftarrow\!\$\, \mathbb{Z}_q^{n\times m}$ satisfies

$$\Pr[\operatorname{rank}(\boldsymbol{A}) < n] = 1 - \prod_{i=0}^{n-1}(1 - q^{i-m}) < \frac{q^{n-m}}{q-1}.$$

Consequently, full row rank holds except with negligible probability whenever $(m - n + 1)\log q = \omega(\log \lambda)$. For the augmented matrix used by PowerBASIS, we apply this statement with $n+1$ rows.

Definition 1 (The Short Integer Solution (SIS) assumption). *For parameters (q, n, m, β), the $\mathsf{SIS}_{q,n,m,\beta}$ problem is to find, given a uniform $\boldsymbol{A} \leftarrow\!\$\, \mathbb{Z}_q^{n\times m}$, a nonzero $\boldsymbol{z} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \pmod q$ and $\|\boldsymbol{z}\| \le \beta$. The SIS assumption states that every PPT algorithm succeeds with probability at most $\mathsf{negl}(\lambda)$.*

3.3 Discrete Gaussian distributions

For $s > 0$ and $\boldsymbol{c} \in \mathbb{R}^m$, define the Gaussian mass functions

$$\rho_{s,\boldsymbol{c}}(\boldsymbol{x}) := \exp\!\left(-\pi\frac{\|\boldsymbol{x}-\boldsymbol{c}\|^2}{s^2}\right), \qquad \rho_{s,\boldsymbol{c}}(S) := \sum_{\boldsymbol{x}\in S}\rho_{s,\boldsymbol{c}}(\boldsymbol{x}).$$

For a lattice $\Lambda \subseteq \mathbb{R}^m$, the discrete Gaussian over $\Lambda + \boldsymbol{c}$ with parameter s, centered at the origin, is

$$D_{\Lambda+\boldsymbol{c},s}(\boldsymbol{x}) := \frac{\rho_{s,\boldsymbol{0}}(\boldsymbol{x})}{\rho_{s,\boldsymbol{0}}(\Lambda+\boldsymbol{c})} \qquad (\boldsymbol{x} \in \Lambda + \boldsymbol{c}).$$

We write D_s^m for the discrete Gaussian over $\mathbb{Z}^m$ centered at the origin. Sampling $\boldsymbol{x} \leftarrow\!\$\, D_s^m$ is equivalent to sampling its m integer coordinates independently from the one-dimensional distribution $D_{\mathbb{Z},s}$.

Smoothing parameter. For $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the least $s > 0$ satisfying

$$\sum_{\boldsymbol{0}\neq\boldsymbol{y}\in\Lambda^*} e^{-\pi s^2\|\boldsymbol{y}\|^2} \le \epsilon.$$

Let $\boldsymbol{B}$ be any basis of a full-rank lattice $\Lambda \subseteq \mathbb{R}^m$. The standard smoothing bounds are [MR04, GPV08]

$$\eta_\epsilon(\Lambda) \le \min\left\{ \frac{1}{\lambda_1^{(\infty)}(\Lambda^*)}, \|\widetilde{\boldsymbol{B}}\| \right\} \sqrt{\frac{\ln(2m(1+1/\epsilon))}{\pi}}. \tag{6}$$

Gaussian tail bounds [MP12]. Let $\Lambda \subseteq \mathbb{R}^m$ be full rank and let $0 < \epsilon < 1$. If $\boldsymbol{x} \hookleftarrow\$ D_{\Lambda+\boldsymbol{c},s}$ and $s \ge \eta_\epsilon(\Lambda)$, then

$$\Pr[\|\boldsymbol{x}\| \ge s\sqrt{m}] \le \frac{1+\epsilon}{1-\epsilon}\, 2^{-m}. \tag{7}$$

For $\boldsymbol{x} \hookleftarrow\$ D_s^m$ and $t > 1$, we have

$$\Pr\left[\|\boldsymbol{x}\| > ts\sqrt{m/(2\pi)}\right] < \left(te^{(1-t^2)/2}\right)^m. \tag{8}$$

For a full-row-rank $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and target $\boldsymbol{u} \in \mathbb{Z}_q^n$, the notation $\boldsymbol{x} \hookleftarrow\$ \boldsymbol{A}_s^{-1}(\boldsymbol{u})$ denotes a sample from $D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),s}$, supported on $\mathbb{Z}^m$. The notation extends columnwise to matrix targets, with independent samples for distinct columns.

Gaussian Splitting Lemma [FMN24, WW23]. Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m_1}$, $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m_2}$, and $\boldsymbol{C} = [\boldsymbol{A} \mid \boldsymbol{B}]$. Assume that the columns of $\boldsymbol{A}$ span $\mathbb{Z}_q^n$, let $\boldsymbol{t} \in \mathbb{Z}_q^n$, and let $s \ge \eta_\epsilon(\Lambda_q^\perp(\boldsymbol{A}))$ for $0 < \epsilon < 1/2$. Consider the following sample.

$$\begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} \hookleftarrow\$ \boldsymbol{C}_s^{-1}(\boldsymbol{t}). \tag{9}$$

Its distribution is within statistical distance $O(\epsilon)$ of the following two-stage experiment.

$$\boldsymbol{v}_2 \hookleftarrow\$ D_s^{m_2}, \qquad \boldsymbol{v}_1 \hookleftarrow\$ \boldsymbol{A}_s^{-1}(\boldsymbol{t} - \boldsymbol{B}\boldsymbol{v}_2).$$

The target in the second step is computed modulo q.

3.4 Gadget trapdoors

A matrix $\boldsymbol{G} \in \mathbb{Z}_q^{n\times t}$ is *primitive* if its columns span $\mathbb{Z}_q^n$. Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{H} \in \mathsf{GL}_n(\mathbb{Z}_q)$. A $\boldsymbol{G}$-trapdoor for $\boldsymbol{A}$ with tag $\boldsymbol{H}$ is an integer matrix $\boldsymbol{R} \in \mathbb{Z}^{m\times t}$ satisfying $\boldsymbol{A}\boldsymbol{R} = \boldsymbol{H}\boldsymbol{G} \pmod q$. Its quality is measured by $s_1(\boldsymbol{R})$. The default tag is $\boldsymbol{H} = \boldsymbol{I}_n$.

Fix an integer radix $\delta \ge 2$ and set $k_q = \lfloor \log_\delta q \rfloor + 1$. This radix is also fixed in the assumptions below. Define the gadget vector and matrix as follows.

$$\boldsymbol{g}^\top = (1, \delta, \ldots, \delta^{k_q-1}), \qquad \boldsymbol{G}_n = \boldsymbol{I}_n \otimes \boldsymbol{g}^\top \in \mathbb{Z}_q^{n\times nk_q}.$$

Entrywise radix decomposition, using representatives in $[0, q-1]$, gives an efficient map

$$\boldsymbol{G}_n^{-1} : \mathbb{Z}_q^{n\times r} \longrightarrow \{0, \ldots, \delta-1\}^{nk_q\times r}$$

such that $\boldsymbol{G}_n\boldsymbol{G}_n^{-1}(\boldsymbol{V}) = \boldsymbol{V} \pmod q$ for every compatible matrix $\boldsymbol{V}$. This map is a deterministic digit decomposition, distinct from the Gaussian preimage notation $\boldsymbol{A}_s^{-1}$.

A $\boldsymbol{G}_n$-trapdoor $\boldsymbol{R}$ induces an efficiently computable basis $\boldsymbol{S}_{\boldsymbol{A}}$ of $\Lambda_q^\perp(\boldsymbol{A})$. The induced basis satisfies the following bound.

$$\|\widetilde{\boldsymbol{S}}_{\boldsymbol{A}}\| \le \big(s_1(\boldsymbol{R}) + 1\big)\sqrt{\delta^2 + 1}. \tag{10}$$

In particular, if $\|\boldsymbol{R}\| \le \beta$, then

$$\eta_\epsilon(\Lambda_q^\perp(\boldsymbol{A})) \le (\beta+1)\sqrt{\delta^2+1}\sqrt{\frac{\ln(2m(1+1/\epsilon))}{\pi}}. \tag{11}$$

Consequently, a width $(\beta+1)\delta\cdot\omega(\sqrt{\log(m\lambda)})$ exceeds the smoothing parameter for a negligible choice of ϵ.

We use the integer-lattice trapdoor generation and preimage sampling algorithms of Micciancio and Peikert [MP12]. Let $t = nk_q$, $d = m - t > n$, and $dt = \omega(\log\lambda)$, and choose $s_0 \ge 1$ so that

$$\epsilon_0 := \frac{t}{2}\sqrt{q^n\left(\frac{1}{s_0}+\frac{1}{q}\right)^d} = \mathsf{negl}(\lambda).$$

Then $(\boldsymbol{A},\boldsymbol{R}) \leftarrow\!\$\ \mathsf{TrapGen}(1^\lambda, n, m, q, \delta, s_0)$ outputs a matrix $\boldsymbol{A}$ within statistical distance ϵ_0 of uniform in $\mathbb{Z}_q^{n\times m}$ and an integer trapdoor satisfying $\boldsymbol{A}\boldsymbol{R} = \boldsymbol{G}_n \pmod q$. Concretely, sample $\overline{\boldsymbol{A}} \leftarrow \mathbb{Z}_q^{n\times d}$ and an integer matrix $\overline{\boldsymbol{R}} \in \mathbb{Z}^{d\times t}$ with independent entries from $D_{\mathbb{Z},s_0}$, and set

$$\boldsymbol{A} = [\overline{\boldsymbol{A}} \mid \boldsymbol{G}_n - \overline{\boldsymbol{A}}\,\overline{\boldsymbol{R}}] \pmod q, \qquad \boldsymbol{R} = \begin{bmatrix}\overline{\boldsymbol{R}}\\ \boldsymbol{I}_t\end{bmatrix}.$$

Except with negligible probability, the output satisfies $\|\boldsymbol{R}\| \le s_0\sqrt{2t(m-t)}$. Suppose that $\boldsymbol{R}$ is a $\boldsymbol{G}_n$-trapdoor for $\boldsymbol{A}$ with invertible tag $\boldsymbol{H}$ and that

$$s \ge \delta\big(s_1(\boldsymbol{R})+1\big)\cdot\omega(\sqrt{\log(m\lambda)}).$$

Then $\mathsf{SamplePre}(\boldsymbol{A},\boldsymbol{R},\boldsymbol{H},\boldsymbol{u},s)$ outputs $\boldsymbol{x}\in\mathbb{Z}^m$ satisfying $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \pmod q$ whose distribution is negligibly close to $\boldsymbol{A}_s^{-1}(\boldsymbol{u})$. The algorithm is applied independently to the columns of a matrix target.

3.5 The PowerBASIS assumption

We now state an integer-lattice analogue of the auxiliary-input assumption underlying the PowerBASIS commitment [FMN24]. We first give its generic form over $\mathbb{Z}_q$.

Definition 2 (BASIS assumption). *Let* Samp *be an efficient randomized algorithm that, on input a uniform* $\boldsymbol{A}\in\mathbb{Z}_q^{n\times m}$*, outputs* $(\boldsymbol{B},\mathsf{pp})$*, where* $\boldsymbol{B}\in\mathbb{Z}_q^{n'\times m'}$ *has full row rank and* pp *denotes the remaining public parameters. The sampler may instead output* $\perp$*, in which case the experiment aborts and the adversary does not win. On a non-aborting execution, sample* $\boldsymbol{T}\leftarrow\boldsymbol{B}_s^{-1}(\boldsymbol{G}_{n'})$ *columnwise, so that* $\boldsymbol{T}\in\mathbb{Z}^{m'\times n'k_q}$*. The* $\mathsf{BASIS}[\mathsf{Samp}]_{q,n,m,\beta,s}$ *assumption requires that, for every* PPT *adversary* $\mathcal{A}$*, the following probability bound holds.*

$$\begin{aligned}\Pr\big[&\boldsymbol{0}\neq\boldsymbol{z}\in\mathbb{Z}^m,\ \|\boldsymbol{z}\|\le\beta,\ \boldsymbol{A}\boldsymbol{z}=\boldsymbol{0}\bmod q:\\ &\boldsymbol{A}\leftarrow\!\$\ \mathbb{Z}_q^{n\times m};\ (\boldsymbol{B},\mathsf{pp})\leftarrow\!\$\ \mathsf{Samp}(\boldsymbol{A});\\ &\boldsymbol{T}\leftarrow\!\$\ \boldsymbol{B}_s^{-1}(\boldsymbol{G}_{n'});\ \boldsymbol{z}\leftarrow\!\$\ \mathcal{A}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{T},\mathsf{pp})\big]\le\mathsf{negl}(\lambda).\end{aligned}$$

PowerBASIS instantiates Samp with a structured block matrix. Let $L\ge 1$, $m\ge n+1$, and sample $\boldsymbol{a}^\top\leftarrow\!\$\ \mathbb{Z}_q^{1\times m}$. Define $\boldsymbol{A}^\star$ as follows.

$$\boldsymbol{A}^\star = \begin{bmatrix}\boldsymbol{a}^\top\\ \boldsymbol{A}\end{bmatrix}\in\mathbb{Z}_q^{(n+1)\times m}.$$

The sampler aborts if $\boldsymbol{A}^\star$ does not have full row rank. We choose parameters with $(m-n)\log q = \omega(\log\lambda)$, so this event has negligible probability for uniform $\boldsymbol{A}$ and $\boldsymbol{a}$. Sample $\boldsymbol{W} \leftarrow\!\!\$\ \mathsf{GL}_{n+1}(\mathbb{Z}_q)$. Set $t = (n+1)k_q$ and define the structured matrix as follows.

$$\boldsymbol{B}_L = \begin{bmatrix} \boldsymbol{W}^0\boldsymbol{A}^\star & \boldsymbol{0} & \cdots & \boldsymbol{0} & -\boldsymbol{G}_{n+1} \\ \boldsymbol{0} & \boldsymbol{W}^1\boldsymbol{A}^\star & \cdots & \boldsymbol{0} & -\boldsymbol{G}_{n+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{W}^{L-1}\boldsymbol{A}^\star & -\boldsymbol{G}_{n+1} \end{bmatrix} \in \mathbb{Z}_q^{L(n+1)\times(Lm+t)}. \tag{12}$$

For $L=1$, this is $\boldsymbol{B}_1 = [\boldsymbol{A}^\star \mid -\boldsymbol{G}_{n+1}]$. Because each $\boldsymbol{W}^i\boldsymbol{A}^\star$ has full row rank, $\boldsymbol{B}_L$ has full row rank as well. Finally, sample the short preimage matrix

$$\boldsymbol{T} \leftarrow\!\!\$\ (\boldsymbol{B}_L)_s^{-1}(\boldsymbol{G}_{L(n+1)}) \in \mathbb{Z}^{(Lm+t)\times Lt}.$$

In particular, $\boldsymbol{B}_L\boldsymbol{T} = \boldsymbol{G}_{L(n+1)} \pmod q$.

Definition 3 (PowerBASIS assumption). *The* $\mathsf{PowerBASIS}_{q,n,m,L,\beta,s}$ *assumption requires that, for every* PPT *adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c|l} & \boldsymbol{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}; \\ & \boldsymbol{a}^\top \leftarrow\!\!\$\ \mathbb{Z}_q^{1\times m}; \\ & \boldsymbol{A}^\star := \begin{bmatrix}\boldsymbol{a}^\top \\ \boldsymbol{A}\end{bmatrix}; \\ \boldsymbol{0} \neq \boldsymbol{z} \in \mathbb{Z}^m, & \\ \|\boldsymbol{z}\| \le \beta, & \text{abort if } \operatorname{rank}(\boldsymbol{A}^\star) < n+1; \\ \boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \bmod q & \boldsymbol{W} \leftarrow\!\!\$\ \mathsf{GL}_{n+1}(\mathbb{Z}_q); \\ & \boldsymbol{B}_L \text{ is defined as in (12)}; \\ & \boldsymbol{T} \leftarrow\!\!\$\ (\boldsymbol{B}_L)_s^{-1}(\boldsymbol{G}_{L(n+1)}); \\ & \boldsymbol{z} \leftarrow\!\!\$\ \mathcal{A}(\boldsymbol{A}^\star, \boldsymbol{W}, \boldsymbol{B}_L, \boldsymbol{T}) \end{array}\right] \le \mathsf{negl}(\lambda).$$

The adversary receives a trapdoor for the structured matrix $\boldsymbol{B}_L$, yet must be unable to solve the original SIS instance defined by $\boldsymbol{A}$. This is an auxiliary-input assumption that implies the underlying SIS assumption for the same parameters. We use the name PowerBASIS for the integer-lattice formulation above and assume its hardness for the selected parameters (q, n, m, L, β, s).

3.6 Non-interactive zero-knowledge arguments of knowledge

Let $\mathcal{R}_{\mathsf{zk}} \subseteq \{0,1\}^* \times \{0,1\}^*$ be an efficiently decidable NP relation. Its language is $\mathcal{L}_{\mathsf{zk}} = \{x : \exists\ w \text{ such that } (x,w) \in \mathcal{R}_{\mathsf{zk}}\}$. A non-interactive zero-knowledge argument of knowledge for $\mathcal{R}_{\mathsf{zk}}$ consists of three PPT algorithms $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$:

$$\begin{aligned} \mathsf{pp} &\leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda), \\ \pi &\leftarrow\!\!\$\ (\mathsf{pp}, x, w), \\ b &\leftarrow \mathsf{Verify}(\mathsf{pp}, x, \pi) \in \{0,1\}. \end{aligned}$$

We require the following properties.

Completeness. For every $(x,w) \in \mathcal{R}_{\mathsf{zk}}$, completeness requires the following bound.

$$\Pr\Big[\mathsf{Verify}(\mathsf{pp}, x, \pi) = 1 : \mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda),\ \pi \leftarrow\!\!\$\ \mathsf{Prove}(\mathsf{pp}, x, w)\Big] \ge 1 - \mathsf{negl}(\lambda). \tag{13}$$

Knowledge extractability. There exist PPT algorithms ExtSetup and Ext such that $(\mathsf{pp}, \tau_{\mathsf{ext}}) \leftarrow\!\!\$\ \mathsf{ExtSetup}(1^\lambda)$ produces public parameters distributed identically to, or computationally indistinguishably from, honest public parameters. For every PPT adversary $\mathcal{A}$, extractability requires

$$\Pr\left[\begin{array}{c|l} \mathsf{Verify}(\mathsf{pp}, x, \pi) = 1 & (\mathsf{pp}, \tau_{\mathsf{ext}}) \leftarrow\!\!\$\ \mathsf{ExtSetup}(1^\lambda); \\ \wedge\ (x, w) \notin \mathcal{R}_{\mathsf{zk}} & (x, \pi) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pp}); \\ & w \leftarrow\!\!\$\ \mathsf{Ext}^{\mathcal{A}}(\tau_{\mathsf{ext}}, x, \pi) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Note that we do not require the extractor to be straight-line for our applications.

Zero knowledge. There exist PPT algorithms SimSetup and SimProve such that no PPT adversary can distinguish the following experiments:

$$\begin{aligned} \mathsf{Real}: &\quad \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), && \pi \leftarrow \mathsf{Prove}(\mathsf{pp}, x, w); \\ \mathsf{Sim}: &\quad (\mathsf{pp}_{\mathsf{sim}}, \tau_{\mathsf{sim}}) \leftarrow \mathsf{SimSetup}(1^\lambda), && \widetilde{\pi} \leftarrow \mathsf{SimProve}(\tau_{\mathsf{sim}}, x). \end{aligned}$$

In both experiments, the adversary may adaptively request a proof for any valid pair $(x, w) \in \mathcal{R}_{\mathsf{zk}}$. In the simulated experiment, the witness is ignored. The resulting joint views must be computationally indistinguishable.

4 Definitions

4.1 Reliable broadcast

A *reliable broadcast* is an asynchronous protocol with a designated *sender*. The sender has some input value m from a known domain $\mathcal{M}$ and each party may output a value in $\mathcal{M}$. A reliable broadcast has the following properties assuming all nonfaulty parties participate in the protocol:

- **Validity.** If the sender is nonfaulty, then every nonfaulty party that completes the protocol outputs the sender's input value m.
- **Agreement.** The values output by any two nonfaulty parties are the same.
- **Termination.** If the dealer is nonfaulty, then all nonfaulty parties complete the protocol and output a value. Furthermore, if some nonfaulty party completes the protocol, every nonfaulty party completes the protocol.

4.2 Packed asynchronous verifiable secret sharing (PAVSS)

We define a packed AVSS using two interactive protocols that take place between n parties: Share and Reconstruct and a PPT algorithm Setup that outputs a common reference string. In Share, the designated dealer receives as input a set of secrets $s_0, \ldots, s_{\ell-1}$ from a finite field $\mathbb{F}$ and all other parties receive no input. None of the parties have any output at the end of Share, but they do update their local state. Because the AVSS is packed, there are ℓ possible invocations of Reconstruct, one for each index k. Each participant thus provides k as input to the protocol, and has as output a field element $v_k \in \mathbb{F}$, which represents their local view of the k-th secret shared by the dealer.

We formally define the environment for a PAVSS below, in terms of capturing the ways in which the adversary can control the network (i.e., when honest parties receive messages) and the other actions the adversary can take. We then formally define three security properties for a PAVSS, which we informally summarize here. Our first definition, *termination*, sets the conditions under

which nonfaulty parties can be guaranteed to complete Share and $\mathsf{Reconstruct}$. Briefly, it says that (1) if the dealer is nonfaulty then all nonfaulty parties will complete Share; (2) if one nonfaulty party completes Share then all nonfaulty parties will; and (3) if all nonfaulty parties complete Share and invoke $\mathsf{Reconstruct}(k)$, then they all will complete $\mathsf{Reconstruct}(k)$. Our next definition, *correctness*, captures the requirement that all nonfaulty parties who complete $\mathsf{Reconstruct}(k)$ should agree on the same secrets, which in turn should be the same as the ones used by the dealer (if it was also nonfaulty). For both these properties, we assume that the Setup algorithm is run honestly and the CRS is available to all the parties. Our final definition, *secrecy*, captures the requirement that an adversary should not be able to learn anything about the k-th secret until the point at which some nonfaulty party invokes $\mathsf{Reconstruct}(k)$.

Our specific PAVSS scheme, $\mathsf{LatBingo}$, is also complete in the sense that every party has a share of each of the secrets. Beyond the above three properties, this thus makes $\mathsf{LatBingo}$ a packed asynchronous complete secret sharing (ACSS) scheme.

4.2.1 Interactive protocols with corruptions A verifiable secret-sharing protocol is an interactive protocol among multiple parties. We assume that messages exchanged between parties are encrypted and authenticated, and identify at least their sender and recipient. Messages may also contain session identifiers or other metadata needed to distinguish concurrent protocol executions.

For every party i, let state_i denote its local state, initially consisting of its secret key, the public keys of the other parties, and its random tape. Let trans_i be the ordered sequence of messages sent and received by party i. The local *view* of party i is $\mathsf{view}_i := (\mathsf{trans}_i, \mathsf{state}_i)$. To capture how this view evolves, we write $\mathsf{view}_{i,t}$ for the view of party i after the t-th step of the protocol; the participant does not know themselves at which step they are but this is defined in a global sense. The *global view* at step t, denoted view_t, contains $\mathsf{view}_{i,t}$ for every party i that is *nonfaulty* at step t.

To model an adversary participating in an interactive protocol, we consider a game that allows them to send messages to nonfaulty participants and, indirectly, cause nonfaulty participants to send messages to each other. To capture the strongest adversary, we allow them to both (1) have complete control over the scheduling of messages and (2) adaptively corrupt nonfaulty participants. To capture this first ability, we provide the adversary with an oracle $\mathsf{O}_{\mathsf{buffer}}$ that acts as a message buffer. The adversary can query this oracle in three different ways: (1) $\mathsf{O}_{\mathsf{buffer}}(\mathsf{add}, i, j, x)$ adds to the message buffer a message x from party i to party j, where i must be a participant the adversary controls, and thus allows the adversary to send a message to a nonfaulty participant on behalf of a corrupted one; (2) $\mathsf{O}_{\mathsf{buffer}}(\mathsf{deliver}, t)$ delivers the t-th message in the buffer to its designated recipient; and (3) $\mathsf{O}_{\mathsf{buffer}}(\mathsf{see})$ shows the adversary the contents of the message buffer, including the sender and recipient of every message. When messages are delivered to a nonfaulty participant, the game also runs the code for that participant on that message. This may in turn result in updates to their internal state and in new messages being added to the buffer and thus allowing nonfaulty participants to send messages to each other (with the adversary controlling the delay with which they are delivered). Messages generated by nonfaulty participants are added to the buffer with their sender and recipient identities, and the adversary learns the content of such a message only when it controls its recipient.

To capture the ability to corrupt nonfaulty participants, we have the game start with an initial set $\mathcal{NF}$ of nonfaulty participants. We then provide the adversary with access to a corruption oracle

$\mathsf{O}_{\mathsf{corr}}$ that, when queried on a given index i at step t, provides the adversary with $\mathsf{view}_{i,t}$ and removes i from $\mathcal{NF}$.

4.2.2 Packed asynchronous verifiable secret sharing Let k be the number of parties, indexed by $[k]$, and let f be the corruption threshold. The dealer is party 1. Let $\mathbb{F}$ be a finite field, let ℓ be the number of shared secrets. A packed asynchronous verifiable secret-sharing scheme (*PAVSS*) consists of two interactive protocols, Share and Reconstruct, executed among the k parties and a PPT Setup algorithm that outputs a common reference string crs. This CRS will be an implicit input to all other protocols. In Share, the designated dealer receives $\boldsymbol{s} = (s_0, \ldots, s_{\ell-1}) \in \mathbb{F}^\ell$ as input, while every other party receives no input. Parties produce no external output upon completing Share, but update their local states. Let $\mathcal{J} = \{0, \ldots, \ell - 1\}$. For every $j \in \mathcal{J}$, an invocation of Reconstruct(j) reconstructs the secret s_j. A party invoking Reconstruct(j) receives an output $v_j \in \mathbb{F}$.

Protocol flags. To capture security definitions for a packed AVSS, we follow the outline described above, starting with a global view view. We augment the view of every nonfaulty party i with $2\ell+4$ Boolean flags. The flags $\mathsf{startedS}_i$ and $\mathsf{finishedS}_i$ indicate whether party i has started and completed Share, respectively. For every $j \in \mathcal{J}$, the flags $\mathsf{readyR}_{i,j}$ and $\mathsf{finishedR}_{i,j}$ indicate whether party i is ready to invoke, and has completed, Reconstruct(j).

A participant's readiness to invoke Reconstruct(j) depends on the context in which the VSS is used; we can think of an external predicate $P(j)$ that tells the participant if the j-th secret is ready to be reconstructed or not. The ith party then invokes Reconstruct(j) once $\mathsf{finishedS}_i = \mathsf{readyR}_{i,j} = \mathsf{true}$. As in Bingo, we allow the adversary to also determine when these external predicates evaluate to true, as well as when participants start the Share protocol. This means that in addition to the $\mathsf{O}_{\mathsf{buffer}}$ and $\mathsf{O}_{\mathsf{corr}}$ oracles described above, we give the adversary access to two oracles:

- On input $i \in \mathcal{NF}$, the oracle $\mathsf{O}_S(i)$ sets $\mathsf{startedS}_i \leftarrow \mathsf{true}$ and instructs party i to execute the first step of Share according to their current view.
- On input $(i, j) \in \mathcal{NF} \times \mathcal{J}$, the oracle $\mathsf{O}_R(i, j)$ sets $\mathsf{readyR}_{i,j} \leftarrow \mathsf{true}$. If $\mathsf{finishedS}_i = \mathsf{true}$, it immediately starts Reconstruct(j) for party i. Otherwise, Reconstruct(j) starts once a call to $\mathsf{O}_{\mathsf{buffer}}(\mathsf{deliver}, \cdot)$ causes the i-th party to finish Share.

Except for the one-shot adaptive secrecy experiment defined below, all our security definitions thus follow the same implicit game structure, in a "meta" game that we call $\mathsf{G}_{\mathcal{A}}^{\mathrm{PAVSS}}(\lambda)$: a PPT adversary $\mathcal{A}$ plays a game with access to four oracles: the message buffer $\mathsf{O}_{\mathsf{buffer}}$, the corruption oracle $\mathsf{O}_{\mathsf{corr}}$, and the two oracles O_S and O_R that dictate when nonfaulty parties are ready to start participating in, respectively, the Share and Reconstruct protocols. In every game, security holds only if an adversary can corrupt at most f participants.

Definition 4 (Correctness). *Suppose that some nonfaulty party completes* Share *and outputs ℓ values $r_0, \ldots, r_{\ell-1} \in \mathbb{F}$. Let t be the first global step at which* view_t *contains* $\mathsf{finishedS}_i = \mathsf{true}$ *for some nonfaulty party i. An extractor* Ext*, given the global view* view_t*, outputs* $(r_0, \ldots, r_{\ell-1}) \leftarrow\!\!\$\ \mathsf{Ext}(\mathsf{view}_t)$*. The PAVSS scheme satisfies correctness if there exists a polynomial-time extractor* Ext *such that, for every* PPT *adversary $\mathcal{A}$, except with negligible probability, the following properties hold simultaneously:*

1. *If the dealer is nonfaulty, then*

$$(r_0, \ldots, r_{\ell-1}) = (s_0, \ldots, s_{\ell-1}).$$

2. *For every $j \in \mathcal{J}$, every nonfaulty party that completes* $\mathsf{Reconstruct}(j)$ *outputs* r_j.

Definition 5 (Termination). *The scheduling in* $\mathsf{G}_{\mathcal{A}}^{PAVSS}(\lambda)$ *is* valid *if the message buffer is empty when* $\mathcal{A}$ *signals that the execution is over; that is, every buffered message is eventually delivered. In every execution with valid scheduling, the following properties must hold except with negligible probability:*

1. *If the dealer is nonfaulty and all nonfaulty parties start* Share*, then all nonfaulty parties eventually complete* Share*.*
2. *If some nonfaulty party completes* Share *and all nonfaulty parties start* Share*, then all nonfaulty parties eventually complete* Share*.*
3. *If all nonfaulty parties complete protocol* Share *and invoke* $\mathsf{Reconstruct}(j)$ *for some* $j \in \mathcal{J}$*, they all eventually complete* $\mathsf{Reconstruct}(j)$*.*

One-shot adaptive Secrecy. We define one-shot adaptive secrecy, wherein we replace the adaptive corruption and reconstruction oracles $\mathsf{O}_{\mathsf{corr}}$ with a one-shot batch oracle $\mathsf{O}_{\mathsf{batch}}$. The adversary may invoke this oracle at any point during the execution and may choose its query as a function of its entire view up to that point. It must, however, submit all corruption queries in one batch:

$$\mathcal{C} \subseteq \mathcal{NF}, \qquad |\mathcal{C}| \leq f \text{ and } \mathcal{Q}_R \subseteq (\mathcal{NF} \setminus \mathcal{C}) \times \mathcal{J}$$

The oracle jointly returns the current views of all parties in $\mathcal{C}$ and also starts the $\mathsf{Reconstruct}(j)$ algorithm for party i, for all $(i, j) \in \mathcal{Q}_R$. After this query, the adversary cannot corrupt any additional parties. Thus, the corrupted set is chosen adaptively, but the individual corruptions cannot be chosen sequentially based on views returned by earlier corruptions.

Definition 6 (One-shot adaptive secrecy). *Let* Sim *be a* PPT *simulator that interacts with* $\mathcal{A}$ *on behalf of all nonfaulty parties, including the dealer. The experiment* $\mathsf{G}_{\mathcal{A}}^{os\text{-}secrecy}(\lambda)$ *is defined below. The bit* $b = 1$ *corresponds to the real execution, while* $b = 0$ *corresponds to the simulated execution. The set* $\mathcal{J}_{\mathsf{rec}}$ *records the secrets whose reconstruction has already been invoked.*

The oracles whose behavior is identical to the PAVSS environment are omitted. In particular, O_S*,* $\mathsf{O}_{\mathsf{buffer}}(\mathsf{add}, \cdot)$*, and* $\mathsf{O}_{\mathsf{buffer}}(\mathsf{see})$ *behave as described above.*

Define the one-shot adaptive secrecy advantage of $\mathcal{A}$ *as*

$$\mathsf{Adv}_{\mathcal{A}}^{os\text{-}secrecy}(\lambda) := \left|2 \Pr\left[\mathsf{G}_{\mathcal{A}}^{os\text{-}secrecy}(\lambda) = 1\right] - 1\right|.$$

The PAVSS scheme satisfies one-shot adaptive secrecy *if there exists a* PPT *simulator* Sim *such that, for every* PPT *adversary* $\mathcal{A}$ *making at most one batch-corruption query for at most* f *parties, there exists a negligible function* $\nu(\cdot)$ *satisfying*

$$\mathsf{Adv}_{\mathcal{A}}^{os\text{-}secrecy}(\lambda) \leq \nu(\lambda).$$

5 A Two-Dimensional Linear Code Commitment

We start with formally defining a 2D bounded-encoder linear code commitment scheme in Section 5.1 before presenting our construction in Section 5.2.

Game $\mathsf{G}_{\mathcal{A}}^{\text{os-secrecy}}(\lambda)$

$b \leftarrow\$ \{0,1\}$
$\mathcal{NF} = [k]$
$\boldsymbol{S} \leftarrow (\perp, \ldots, \perp)$
$\mathcal{C}_{\mathsf{batch}} \leftarrow \emptyset$
$\mathcal{J}_{\mathsf{rec}} \leftarrow \emptyset$
$q_{\mathsf{init}} \leftarrow 0$; $q_{\mathsf{batch}} \leftarrow 0$
if $b = 0$, run $(\mathsf{crs}, \tau_s) \leftarrow\$ \mathsf{Sim}(1^\lambda)$
if $b = 1$, run $\mathsf{crs} \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}}(1^\lambda, \mathsf{crs})$, where
  $\mathcal{O} = \{\mathsf{O}_{\mathsf{buffer}}(\mathsf{add}, \cdot), \mathsf{O}_{\mathsf{buffer}}(\mathsf{see}),$
    $\mathsf{O}^*, \mathsf{O}_{\mathsf{batch}}, \mathsf{O}_{\mathsf{init}}, \mathsf{O}_S\}$
return $(b' = b)$

$\mathsf{O}^*(h)$

Run delivery of the h-th buffered message:
  if $b = 0$, run $\mathsf{Sim}(\tau_s, h)$;
  if $b = 1$, run $\mathsf{O}_{\mathsf{buffer}}(\mathsf{deliver}, h)$.

$\mathsf{O}_{\mathsf{init}}(s_0, \ldots, s_{\ell-1})$

If $q_{\mathsf{init}} = 1$, return $\perp$
$q_{\mathsf{init}} \leftarrow 1$
$S_a \leftarrow s_a \ \forall\ a \in \{0, \ldots, \ell - 1\}$
If $b = 1$, add $\boldsymbol{S}$ to the dealer's state.
If $b = 0$ and $q_{\mathsf{batch}} = 1$:
  if $1 \in \mathcal{C}_{\mathsf{batch}}$,
    add $\boldsymbol{S}$ to $\mathsf{state}_{\mathsf{Sim}}$;
  otherwise, add $\{S_j\}_{j \in \mathcal{J}_{\mathsf{rec}}}$ to $\mathsf{state}_{\mathsf{Sim}}$.

$\mathsf{O}_{\mathsf{batch}}(\mathcal{C}, \mathcal{Q}_R)$

If $q_{\mathsf{batch}} = 1$, return $\perp$.
If $\mathcal{C} \not\subseteq \mathcal{NF}$ or $|\mathcal{C}| > f$, return $\perp$.
If $\mathcal{Q}_R \not\subseteq (\mathcal{NF} \setminus \mathcal{C}) \times \mathcal{J}$, return $\perp$.
$q_{\mathsf{batch}} \leftarrow 1$; $\mathcal{C}_{\mathsf{batch}} \leftarrow \mathcal{C}$
$\mathcal{J}_{\mathsf{rec}} \leftarrow \{j \in \mathcal{J} :$
  $\exists\ i,\ (i, j) \in \mathcal{Q}_R\}$
If $b = 0$ and $q_{\mathsf{init}} = 1$:
  if $1 \in \mathcal{C}$, add $\boldsymbol{S}$ to $\mathsf{state}_{\mathsf{Sim}}$;
  otherwise, add $\{S_j\}_{j \in \mathcal{J}_{\mathsf{rec}}}$ to $\mathsf{state}_{\mathsf{Sim}}$.
If $b = 0$, obtain $\{\mathsf{view}_i\}_{i \in \mathcal{C}}$ jointly from $\mathsf{Sim}(\tau_s, \mathsf{batch}, \mathcal{C}, \mathcal{Q}_R)$.
If $b = 1$, obtain the actual $\{\mathsf{view}_i\}_{i \in \mathcal{C}}$.
$\mathcal{NF} \leftarrow \mathcal{NF} \setminus \mathcal{C}$
For every $(i, j) \in \mathcal{Q}_R$:
  $\mathsf{readyR}_{i,j} \leftarrow \mathsf{true}$;
  if $\mathsf{finishedS}_i = \mathsf{true}$,
    start $\mathsf{Reconstruct}(j)$ for party i.
Return $\{\mathsf{view}_i\}_{i \in \mathcal{C}}$.

Fig. 1. The one-shot adaptive secrecy game for a packed asynchronous verifiable secret sharing scheme. The adversary submits all corruption and reconstruction queries in a single call to $\mathsf{O}_{\mathsf{batch}}$.

5.1 Definition

We first define bounded-encoder linear codes, a special type of linear codes that we rely on in our construction.

Definition 7 (Bounded-Encoder Linear Code). *Let $k, d, t_c \in \mathbb{N}$ with $d \le t_c \le k$, and let $L \ge 1$. A* bounded-encoder linear code *with block length k, dimension d, reconstruction threshold t_c, and encoder bound L is given by an integer encoder matrix $\boldsymbol{G}_{\mathsf{code}} \in \mathbb{Z}^{k \times d}$ whose rows have ℓ_1-norm at most L, such that for every $I \subseteq [k]$ with $|I| \ge t_c$ there is an efficiently computable decoder matrix $\boldsymbol{D}(I) \in \mathbb{F}_q^{d \times |I|}$ satisfying*

$$\boldsymbol{D}(I)(\boldsymbol{G}_{\mathsf{code}})_I = \boldsymbol{I}_d \quad \text{over } \mathbb{F}_q. \tag{14}$$

The encoding of a message $\boldsymbol{x} \in \mathbb{F}_q^d$ is $\boldsymbol{G}_{\mathsf{code}}\boldsymbol{x}$. We require L to be polynomial in k. Note that no bound is required on the entries of the decoder matrices. The encoder may be sampled by a randomized setup algorithm Setup*, in which case the decoding guarantee is required to hold simultaneously for all $I \subseteq [k]$ with $|I| \ge t_c$, except with negligible probability over the setup randomness.*

Notation. Let λ be the security parameter, let $\mathbb{F}_q$ be a finite field, and let k be the number of parties. Fix matrix dimensions r, c, a packing parameter $\ell \ge 1$, and a corruption bound f.

The CRS specifies two bounded-encoder linear codes with encoding matrices $\boldsymbol{G}_{\mathsf{code}} \in \mathbb{F}_q^{k \times r}$ and $\boldsymbol{H}_{\mathsf{code}} \in \mathbb{F}_q^{k \times c}$. We write $\boldsymbol{g}_i^\top$ and $\boldsymbol{h}_i^\top$ for their respective i-th rows. The component codes have reconstruction thresholds t_c^G and t_c^H: for every $I, J \subseteq [k]$ with $|I| \ge t_c^G$ and $|J| \ge t_c^H$, there are efficient decoding maps satisfying

$$\boldsymbol{D}_G(I)\boldsymbol{G}_I = \boldsymbol{I}_r, \qquad \boldsymbol{D}_H(J)\boldsymbol{H}_J = \boldsymbol{I}_c \quad \text{over } \mathbb{F}_q. \tag{15}$$

The CRS additionally specifies packing vectors $\boldsymbol{b}_0 \in \mathbb{F}_q^r$, and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_\ell \in \mathbb{F}_q^c$. To use one evaluation interface for both codeword entries and packing-related values, we define $\boldsymbol{g}_0 = \boldsymbol{b}_0$ and $\boldsymbol{h}_{1-t} = \boldsymbol{a}_t$ for all $t \in [\ell]$. Thus $\boldsymbol{h}_0 = \boldsymbol{a}_1$, $\boldsymbol{h}_{-1} = \boldsymbol{a}_2$, and $\boldsymbol{h}_{1-\ell} = \boldsymbol{a}_\ell$. Define the index domains

$$\mathcal{I} = \{0\} \cup [k], \qquad \mathcal{J} = \{1-\ell, \ldots, 0\} \cup [k], \qquad \mathcal{D} = \mathcal{I} \times \mathcal{J}.$$

For $\boldsymbol{M} \in \mathbb{F}_q^{r \times c}$ and $(i, j) \in \mathcal{D}$, write $\mathsf{val}_M(i, j) = \boldsymbol{g}_i^\top \boldsymbol{M} \boldsymbol{h}_j$. In particular, the restriction to $[k]^2$ is the encoded array $\boldsymbol{G}_{\mathsf{code}} \boldsymbol{M} \boldsymbol{H}_{\mathsf{code}}^\top$, and $\mathsf{val}_M(0, 1-t) = \boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_t$ are the masks for all $t \in [\ell]$.

A 2D bounded-encoder linear code commitment scheme $\mathsf{2DECCom}$ consists of the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, k, r, c, \ell, f) \to \mathsf{crs}$ outputs the public parameters including the code description.
- $\mathsf{Commit}(\mathsf{crs}, \boldsymbol{M}) \to (\mathsf{com}, \mathsf{st})$ takes an arbitrary matrix $\boldsymbol{M} \in \mathbb{F}_q^{r \times c}$ and outputs a commitment com and a commitment state st.
- $\mathsf{Eval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, (i, j)) \to (y, \pi)$ takes $(i, j) \in \mathcal{D}$ and outputs $y = \mathsf{val}_M(i, j)$ together with a proof π.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i, j), y, \pi) \to \{0, 1\}$ verifies a claimed value and opening proof for $(i, j) \in \mathcal{D}$.
- $\mathsf{GetProofs}(\mathsf{crs}, \mathsf{com}, \mathsf{dir}, \zeta, \mathcal{P}, V)$ returns target openings or $\perp$, in one of the following directions:
  - If $\mathsf{dir} = \mathsf{row}$, then $\zeta = i \in [k]$ and

$$\mathcal{P} = \{(j, y_j, \pi_j) : j \in J\}, \qquad J \subseteq [k], \quad |J| \ge t_c^H.$$

The source tuples refer to points (i, j). For a target set $V \subseteq \mathcal{J}$, the algorithm returns

$$\{(j', y_{j'}, \pi_{j'}) : j' \in V\},$$

where the tuple indexed by j' refers to (i, j'). Thus, target columns may include packing indices $1 - \ell, \ldots, 0$.

- If $\mathsf{dir} = \mathsf{col}$, then $\zeta = j \in [k]$ and

$$\mathcal{P} = \{(i, y_i, \pi_i) : i \in I\}, \qquad I \subseteq [k], \quad |I| \geq t_c^G.$$

For a target set $V \subseteq \mathcal{I}$, the algorithm returns

$$\{(i', y_{i'}, \pi_{i'}) : i' \in V\},$$

where the tuple indexed by i' refers to (i', j). Thus the target rows may include index 0.

In both directions, source positions must be distinct and every source point must lie in $[k]^2$.

For compact notation, we define $\mathsf{pt}_{\mathsf{row}}(i, \gamma) = (i, \gamma)$ and $\mathsf{pt}_{\mathsf{col}}(j, \gamma) = (\gamma, j)$. and $t_{\mathsf{row}} = t_c^H, t_{\mathsf{col}} = t_c^G$. An input $(\mathsf{dir}, \zeta, \mathcal{P}, V)$ is *admissible* if it satisfies the corresponding direction, source-domain, source-count, and target-domain requirements above.

Full-opening relation. The scheme specifies an efficiently decidable full-opening relation

$$\mathcal{R}_{\mathsf{open}}(\mathsf{crs}, \mathsf{com}, \boldsymbol{M}, \mathsf{st}) \in \{0, 1\}.$$

A tuple $(\boldsymbol{M}, \mathsf{st})$ is a valid full opening of com only if it satisfies the relation above. Honest outputs of $\mathsf{Commit}(\mathsf{crs}, \boldsymbol{M})$ satisfy this relation, except with negligible probability. We write

$$\mathsf{com} \equiv \mathsf{Commit}(\mathsf{crs}, \boldsymbol{M}; \mathsf{st}) \quad \Longleftrightarrow \quad \mathcal{R}_{\mathsf{open}}(\mathsf{crs}, \mathsf{com}, \boldsymbol{M}, \mathsf{st}) = 1.$$

A 2D bounded-encoder linear code commitment scheme 2DECCom must satisfy the following properties.

Correctness. Honest evaluation proofs and proofs interpolated from honest source opening proofs must pass verification.

Definition 8 (Correctness). *For every* $\mathsf{crs} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, k, r, c, \ell, f)$*, every* $\boldsymbol{M} \in \mathbb{F}_q^{r \times c}$*, consider* $(\mathsf{com}, \mathsf{st}) \leftarrow\!\$\ \mathsf{Commit}(\mathsf{crs}, M)$*. For every* $(i, j) \in \mathcal{D}$*, if* $(y, \pi) \leftarrow\!\$\ \mathsf{Eval}(\mathsf{crs}, M, \mathsf{st}, (i, j))$*, then*

$$y = \mathsf{val}_M(i, j) \ \wedge \ \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i, j), y, \pi) = 1 \ \wedge \mathcal{R}_{\mathsf{open}}(\mathsf{crs}, \mathsf{com}, \boldsymbol{M}, \mathsf{st}) = 1$$

except with negligible probability.

Moreover, let $(\mathsf{dir}, \zeta, \mathcal{P}, V)$ *be any admissible input whose source tuples are generated by* Eval *using the same* $(\boldsymbol{M}, \mathsf{st})$*. Except with negligible probability,* $\mathsf{GetProofs}(\mathsf{crs}, \mathsf{com}, \mathsf{dir}, \zeta, \mathcal{P}, V)$ *returns exactly one tuple* $(\gamma, y_\gamma, \pi_\gamma)$ *for every* $\gamma \in V$*, satisfying* $y_\gamma = \mathsf{val}_M\big(\mathsf{pt}_{\mathsf{dir}}(\zeta, \gamma)\big)$ *and*

$$\mathsf{Verify}\big(\mathsf{crs}, \mathsf{com}, \mathsf{pt}_{\mathsf{dir}}(\zeta, \gamma), y_\gamma, \pi_\gamma\big) = 1.$$

Probabilities are over the randomness of setup, commitment generation, evaluation, and proof reconstruction.

Computational Proof Canonicality. We require that it is hard for any PPT adversary to come up with two valid opening proofs for the same point and value.

Definition 9 (Computational proof canonicality). *A two-dimensional commitment scheme satisfies* computational proof canonicality *if, for every admissible polynomially bounded choice of parameters (k, r, c, ℓ, f) and every* PPT *adversary $\mathcal{A}$, consider the experiment*

$$\mathsf{crs} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, k, r, c, \ell, f), \qquad (\mathsf{com}, (i,j), y, \boldsymbol{\pi}, \boldsymbol{\pi}') \leftarrow\!\$\ \mathcal{A}(\mathsf{crs}).$$

The following probability, over this experiment, is negligible in λ:

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{can}}_{\mathsf{2DECCom},\mathcal{A}}(\lambda)\\ &:= \Pr\begin{bmatrix}(i,j) \in \mathcal{D}, \quad y \in \mathbb{F}_q, \quad \boldsymbol{\pi} \neq \boldsymbol{\pi}',\\ \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i,j), y, \boldsymbol{\pi}) = 1,\\ \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i,j), y, \boldsymbol{\pi}') = 1\end{bmatrix}.\end{aligned}$$

Evaluation binding. It must be computationally infeasible to open the same commitment to two different values at the same point.

Definition 10 (Evaluation binding). *A commitment scheme* 2DECCom *satisfies evaluation binding if for every* PPT *adversary $\mathcal{A}$, the following advantage is negligible in λ:*

$$\Pr\begin{bmatrix}(i,j) \in \mathcal{D} \wedge y \neq y' \wedge & (k, r, c, \ell, f) \leftarrow\!\$\ \mathcal{A}(1^\lambda)\\ \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i,j), y, \pi) = 1 \wedge : & \mathsf{crs} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, k, r, c, \ell, f)\\ \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i,j), y', \pi') = 1 & (\mathsf{com}, (i,j), (y, \pi), (y', \pi')) \leftarrow\!\$\ \mathcal{A}(\mathsf{crs})\end{bmatrix}$$

Proof interpolation. We require that proof reconstruction via GetProofs must succeed even when the source openings are adversarially generated.

Definition 11 (Proof interpolation). *A commitment scheme* 2DECCom *satisfies proof interpolation, if for every* PPT *adversary $\mathcal{A}$, sample*

$$\mathsf{crs} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, k, r, c, \ell, f)$$

and let

$$(\mathsf{com}, \mathsf{dir}, \zeta, \mathcal{P}, V) \leftarrow\!\$\ \mathcal{A}(\mathsf{crs}).$$

Write

$$\mathcal{P} = \{(\gamma, y_\gamma, \pi_\gamma) : \gamma \in \Gamma\}.$$

The probability of the following is negligible in λ:

1. *$(\mathsf{dir}, \zeta, \mathcal{P}, V)$ is admissible. In particular, $\zeta \in [k]$, $\Gamma \subseteq [k]$, $|\Gamma| \geq t_{\mathsf{dir}}$, and the source positions are distinct;*
2. *every source opening verifies:*

$$\mathsf{Verify}\big(\mathsf{crs}, \mathsf{com}, \mathsf{pt}_{\mathsf{dir}}(\zeta, \gamma), y_\gamma, \pi_\gamma\big) = 1 \qquad (\gamma \in \Gamma);$$

3. $\mathsf{GetProofs}(\mathsf{crs}, \mathsf{com}, \mathsf{dir}, \zeta, \mathcal{P}, V)$ *returns* $\perp$*, fails to return exactly one tuple at every requested target, returns a target tuple* $(\gamma', y_{\gamma'}, \pi_{\gamma'})$ *such that*

$$\mathsf{Verify}\big(\mathsf{crs}, \mathsf{com}, \mathsf{pt}_{\mathsf{dir}}(\zeta, \gamma'), y_{\gamma'}, \pi_{\gamma'}\big) = 0,$$

or returns a target tuple whose value is not the linear decoding of the source values, that is, $y_{\gamma'} \neq \boldsymbol{h}_{\gamma'}^{\top} \boldsymbol{D}_H(\Gamma)(y_\gamma)_{\gamma\in\Gamma}$ *for* $\mathsf{dir} = \mathsf{row}$ *and* $y_{\gamma'} \neq \boldsymbol{g}_{\gamma'}^{\top} \boldsymbol{D}_G(\Gamma)(y_\gamma)_{\gamma\in\Gamma}$ *for* $\mathsf{dir} = \mathsf{col}$*.*

One-shot adaptive simulation hiding. We require the commitment scheme to satisfy a hiding property that we call one-shot adaptive simulation hiding. In particular, there must exist a simulator that can produce a commitment without knowing anything about the matrix. Moreover, it must be able to produce valid opening proofs for an adversarially-chosen subset of points.

We define some notation before presenting the formal definition. For $\boldsymbol{s} = (s_1, \ldots, s_\ell) \in \mathbb{F}_q^\ell$, define

$$\mathsf{h}(\boldsymbol{M}) = \big(\mathsf{val}_M(0, 1-t)\big)_{t\in[\ell]} = (\boldsymbol{b}_0^{\top} \boldsymbol{M} \boldsymbol{a}_t)_{t\in[\ell]}, \qquad \boldsymbol{z} = \boldsymbol{s} + \mathsf{h}(\boldsymbol{M}).$$

Note that the pads belong to the sharing layer and are not part of the commitment.

For any $J \subseteq [k]$ and $T \subseteq [\ell]$, let $\bar{J} = [k] \setminus J$ and $\boldsymbol{A}_T = [\boldsymbol{a}_t]_{t\in T}$. We define a linear map $\mathcal{L}_{J,T}$ as follows:

$$\mathcal{L}_{J,T}(\boldsymbol{M}) = \big(\boldsymbol{G}_J \boldsymbol{M},\ \boldsymbol{M} \boldsymbol{H}_J^{\top},\ \boldsymbol{G}_{\bar{J}} \boldsymbol{M} \boldsymbol{A}_T\big). \tag{16}$$

The adversary is allowed to query the above linear map for any J, T. Looking ahead, the set J will denote the set of parties the adversary corrupts in the AVSS scheme, and T will denote the secret values that the adversary will query reconstruction for. Hence, the above linear map captures all the scalar information that the adversary will receive in response to its query. We refer to this map as the leakage map corresponding to J, T. We also define:

$$\mathcal{Q}_{J,T} = \big((i,j)\big)_{i\in J,\ j\in[k]} \parallel \big((i,j)\big)_{i\in[k],\ j\in J} \parallel \big((i, 1-t)\big)_{i\in\bar{J},\ t\in T},$$

where $\parallel$ denotes concatenation. Define $\mathsf{BatchEval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, J, T)$ as the algorithm that, for each (i,j) in $\mathcal{Q}_{J,T}$, runs

$$(y_{i,j}, \pi_{i,j}) \leftarrow\!\!\$\ \mathsf{Eval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, (i,j))$$

and returns the list Π of resulting proofs. The scalar values authenticated by these proofs are equal to $\mathcal{L}_{J,T}(\boldsymbol{M})$.

Definition 12 (One-shot adaptive simulation hiding). *A 2D bounded-encoder linear code commitment scheme is one-shot adaptively simulation hiding for* (f, ℓ) *if there exist* PPT *algorithms*

$$\begin{aligned}
\mathsf{SimSetup}(1^\lambda, k, r, c, \ell, f) &\to (\mathsf{crs}, \tau_{\mathsf{sim}}),\\
\mathsf{SimCommit}(\mathsf{crs}, \tau_{\mathsf{sim}}) &\to (\mathsf{com}, \omega),\\
\mathsf{SimFull}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{M}) &\to \mathsf{st}' \text{ or } \perp,\\
\mathsf{SimBatch}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{z}, J, T, E) &\to \Pi' \text{ or } \perp
\end{aligned}$$

such that, for every PPT *adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$*,*

$$\Big|\Pr\Big[\mathbf{G}_{\mathcal{A},1}^{\mathsf{wa\text{-}sh}}(1^\lambda) = 1\Big] - \Pr\Big[\mathbf{G}_{\mathcal{A},0}^{\mathsf{wa\text{-}sh}}(1^\lambda) = 1\Big]\Big| \le \mathrm{negl}(\lambda),$$

where the experiments are defined in Figure 2.

Game $\mathbf{G}^{\mathsf{wa\text{-}sh}}_{\mathcal{A},\chi}(1^\lambda)$, $\chi \in \{0,1\}$

$\mathbf{if}\ \chi = 1\ \ \mathbf{then}\ \mathsf{crs} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, k, r, c, \ell, f)$
$\mathbf{else}\quad (\mathsf{crs}, \tau_{\mathsf{sim}}) \leftarrow\!\$\ \mathsf{SimSetup}(1^\lambda, k, r, c, \ell, f)$
$(\boldsymbol{s}, \mathsf{aux}) \leftarrow\!\$\ \mathcal{A}_0(\mathsf{crs})$
$\mathbf{if}\ \boldsymbol{s} \notin \mathbb{F}_q^\ell\ \ \mathbf{then\ return}\ 0$
$\boldsymbol{M} \leftarrow\!\$\ \mathbb{F}_q^{r\times c}$
$\boldsymbol{z} \leftarrow \boldsymbol{s} + \mathsf{h}(\boldsymbol{M})$
$\mathbf{if}\ \chi = 1\ \ \mathbf{then}\ (\mathsf{com}, \mathsf{st}) \leftarrow\!\$\ \mathsf{Commit}(\mathsf{crs}, \boldsymbol{M})$
$\mathbf{else}\quad (\mathsf{com}, \omega) \leftarrow\!\$\ \mathsf{SimCommit}(\mathsf{crs}, \tau_{\mathsf{sim}})$
$(Q, \mathsf{aux}) \leftarrow\!\$\ \mathcal{A}_1(\mathsf{crs}, \mathsf{com}, \boldsymbol{z}, \mathsf{aux})$
$\mathbf{if}\ Q \notin \mathcal{Q}_{\mathsf{adm}}\ \ \mathbf{then\ return}\ 0$
$\mathbf{if}\ Q = \mathsf{Full}\ \ \mathbf{then}$
  $E \leftarrow \boldsymbol{M}$
  $\mathbf{if}\ \chi = 1\ \ \mathbf{then}\ \Pi \leftarrow \mathsf{st}$
  $\mathbf{else}\quad \Pi \leftarrow\!\$\ \mathsf{SimFull}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{M})$
$\mathbf{else}$
  parse $Q = \mathsf{Batch}(J, T)$
  $E \leftarrow \mathcal{L}_{J,T}(\boldsymbol{M})$
  $\mathbf{if}\ \chi = 1\ \ \mathbf{then}\ \Pi \leftarrow\!\$\ \mathsf{BatchEval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, J, T)$
  $\mathbf{else}\quad \Pi \leftarrow\!\$\ \mathsf{SimBatch}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{z}, J, T, E)$
$\mathbf{return}\ \mathcal{A}_2(\mathsf{crs}, \mathsf{com}, \boldsymbol{z}, Q, E, \Pi, \mathsf{aux})$

Fig. 2. The one-query one-shot adaptive simulation-hiding experiment for a 2D bounded-encoder linear code commitment scheme. The case $\chi = 1$ is real and $\chi = 0$ is simulated.

The adversary is allowed to make exactly one query, and it must belong to the admissible set defined below:

$$\mathcal{Q}_{\mathsf{adm}} = \{\mathsf{Full}\} \cup \{\mathsf{Batch}(J,T) : J \subseteq [k],\ |J| \le f,\ T \subseteq [\ell],\ |T| \le \ell - 1\}\,.$$

For a full query, SimFull *receives the actual matrix* $\boldsymbol{M}$ *and simulates the commitment state. For a batch query,* SimBatch *receives the public pads, the chosen index sets, and the leaked values corresponding to* J, T *i.e.* $E = \mathcal{L}_{J,T}(\boldsymbol{M})$*, together with its setup and commitment state. It simulates the entire joint proof response.*

Remark 1. Note that we call the above primitive a 2D bounded-encoder linear code commitment scheme since we will only need to commit with respect to 2D bounded-encoder linear codes. But, our definition applies more generally to a commitment scheme for any two dimensional linear code.

5.2 Our Construction

5.2.1 Component codes and notation We instantiate the definition of 2DECCom over $\mathbb{F}_q$, for an odd prime q, using the integer-lattice version of PowerBASIS from the preliminaries. Thus $\boldsymbol{M} \in \mathbb{F}_q^{r\times c}$, whereas the coefficient openings belong to $\mathbb{Z}^m$.

Write

$$\begin{aligned} r &= R_G k + O(1), \qquad & c &= R_H k + O(1), \\ t_c^G &= \lceil \tau_c^G k \rceil, & t_c^H &= \lceil \tau_c^H k \rceil. \end{aligned} \tag{17}$$

Here R_G, R_H are the two constant rates, and τ_c^G, τ_c^H are the reconstruction fractions for the two component codes. The last line holds over $\mathbb{F}_q$ whenever $|I| \geq t_c^G$ and $|J| \geq t_c^H$. Let $\varepsilon_{\mathsf{code}}$ bound failure of either instance.

We number coefficients by $0 \leq a < r$, $0 \leq b < c$, and parties by $i \in [k]$. Recall that $\boldsymbol{g}_0 = \boldsymbol{b}_0$ and $\boldsymbol{h}_0 = \boldsymbol{a}_1, \ldots, \boldsymbol{h}_{1-\ell} = \boldsymbol{a}_\ell$ are the vectors that will be used to mask the secrets. In particular, the t-th mask is $\mathsf{h}_t(\boldsymbol{M}) = \mathsf{val}_{\boldsymbol{M}}(0, 1-t) = \boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_t$. As in the definition, the sharing layer publishes $\boldsymbol{z} = \boldsymbol{s} + (\mathsf{h}_t(\boldsymbol{M}))_{t \in [\ell]}$; these pads are not part of the commitment.

Every code or mask coefficient is used as an integer on opening vectors and is reduced modulo q on values. For a vector $\boldsymbol{x} \in \mathbb{F}_q^m$, we write $\mathsf{lift}(\boldsymbol{x}) \in \mathbb{Z}^m$ for its entrywise centered integer representative in $(-q/2, q/2]^m$; the decoder matrices act over $\mathbb{F}_q$, and lift returns the resulting proof vectors to $\mathbb{Z}^m$. Define

$$\begin{aligned} L_G &= \max\{1, \max_{i \in [k]} \|\boldsymbol{g}_i\|_1\}, \qquad & L_H &= \max\{1, \max_{j \in [k]} \|\boldsymbol{h}_j\|_1\}, \\ L_{\mathcal{I}} &= \max\{1, \max_{i \in \mathcal{I}} \|\boldsymbol{g}_i\|_1\}, \qquad & L_{\mathcal{J}} &= \max\{1, \max_{j \in \mathcal{J}} \|\boldsymbol{h}_j\|_1\}, \\ B(i,j) &= \|\boldsymbol{g}_i\|_1 \|\boldsymbol{h}_j\|_1 \beta_s, & B_{\max} &= L_{\mathcal{I}} L_{\mathcal{J}} \beta_s. \end{aligned} \tag{18}$$

By Definition 7, L_G, L_H are polynomial in k.

5.2.2 Construction Let $n \geq 2$ be the syndrome dimension, let m be the opening dimension, and write $d_g = n\lceil \log_2 q \rceil$ for the number of columns of $\boldsymbol{G} \in \mathbb{F}_q^{n \times d_g}$. Set

$$\begin{aligned} \boldsymbol{K}_a &= \boldsymbol{W}^{-a} \quad (0 \leq a < r), \\ \boldsymbol{B}_r &= \left[\mathrm{diag}(\boldsymbol{A}, \boldsymbol{W}\boldsymbol{A}, \ldots, \boldsymbol{W}^{r-1}\boldsymbol{A}) \;\middle|\; -\boldsymbol{1}_r \otimes \boldsymbol{G} \right], \\ \boldsymbol{G}_r &= \boldsymbol{I}_r \otimes \boldsymbol{G}. \end{aligned} \tag{19}$$

Here $\boldsymbol{B}_r$ has rn rows and $rm + d_g$ columns. We use one PowerBASIS commitment $\boldsymbol{c}_b$ for each column of $\boldsymbol{M}$. We will use a NIZK argument of knowledge $\Pi_{\mathsf{wf}}^{\mathsf{bi}}$ for the following relation:

$$\begin{aligned} \mathcal{R}_{\mathsf{wf}}^{\mathsf{2d}} := \Big\{ \big((\boldsymbol{A}_\star, \boldsymbol{W}, (\boldsymbol{c})_{b=0}^{c-1}); (\boldsymbol{s}_{a,b})_{\substack{0 \leq a < r \\ 0 \leq b < c}} \big) : \|\boldsymbol{s}_{a,b}\| \leq \beta_s \quad \text{and} \\ \boldsymbol{A}_\star \boldsymbol{s}_{a,b} = \big(\boldsymbol{W}^{-a} \boldsymbol{c}_b\big)_\star \quad \text{for every } a, b \Big\}. \end{aligned} \tag{20}$$

The commitment additionally contains a NIZK proof of knowledge for the above relation. We denote the public statement by $\mathsf{x}_{\mathsf{wf}} = (\boldsymbol{A}_\star, \boldsymbol{W}, (\boldsymbol{c}_b)_{b<c})$, where $\boldsymbol{A}_\star$ consists of the last $n-1$ rows of $\boldsymbol{A}$. We also define the public linear combinations

$$\boldsymbol{K}(\boldsymbol{u}) = \sum_{a<r} u_a \boldsymbol{K}_a, \qquad \boldsymbol{c}(\boldsymbol{v}) = \sum_{b<c} v_b \boldsymbol{c}_b. \tag{21}$$

The coefficient bound and sampling widths are specified in Section 6.2.2; in particular, $2B_{\max} < q$ and $\beta_{\mathsf{PB}} \geq 2B_{\max}$ where β_{PB} is the norm bound for our PowerBASIS assumption.

$\mathsf{Setup}(1^\lambda, k, r, c, \ell, f)$:

1. Sample
$$\boldsymbol{G}_{\mathsf{code}} \leftarrow\!\$\ \mathsf{LC}_G.\mathsf{Setup}(1^k, r), \qquad \boldsymbol{H}_{\mathsf{code}} \leftarrow\!\$\ \mathsf{LC}_H.\mathsf{Setup}(1^k, c).$$
2. Sample $\boldsymbol{b}_0 \leftarrow\!\$\ \{0,\dots,D-1\}^r$ and for all $i \in [\ell]$, sample $\boldsymbol{a}_i \leftarrow\!\$\ \{0,\dots,D-1\}^c$. Set $\mathsf{crs}_{\mathsf{code}} = (\boldsymbol{G}_{\mathsf{code}}, \boldsymbol{H}_{\mathsf{code}}, \boldsymbol{b}_0, \{\boldsymbol{a}_i\}_{[\ell]})$.
3. Sample $(\boldsymbol{A}, \boldsymbol{R}) \leftarrow\!\$\ \mathsf{TrapGen}(1^\lambda, n, m, q)$ and $\boldsymbol{W} \leftarrow\!\$\ \mathsf{GL}_n(\mathbb{F}_q)$, with $\boldsymbol{AR} = \boldsymbol{G}$. Form $\boldsymbol{B}_r, \boldsymbol{G}_r$ as in (19). For $a < r$, set
$$\boldsymbol{R}_a = \boldsymbol{R}\boldsymbol{G}^{-1}(\boldsymbol{K}_a\boldsymbol{G}), \qquad \widetilde{\boldsymbol{R}} = \begin{bmatrix}\mathrm{diag}(\boldsymbol{R}_0,\dots,\boldsymbol{R}_{r-1}) \\ \boldsymbol{0}_{d_g \times r d_g}\end{bmatrix}.$$
Re-randomize the trapdoor by sampling: $\boldsymbol{T}_r \leftarrow\!\$\ \mathsf{SamplePre}(\boldsymbol{B}_r, \widetilde{\boldsymbol{R}}, \boldsymbol{G}_r, \sigma_0)$.
4. Sample $\mathsf{crs}_{\mathsf{wf}} \leftarrow\!\$\ \Pi_{\mathsf{wf}}.\mathsf{Setup}(1^\lambda)$. Publish
$$\mathsf{crs} = (\mathsf{crs}_{\mathsf{code}}, \boldsymbol{A}, \boldsymbol{W}, \boldsymbol{T}_r, \mathsf{crs}_{\mathsf{wf}}).$$

Fig. 3. The Setup algorithm for our 2D bounded-encoder linear code commitment scheme Basis2DLCCom.

The full-opening relation. The state is the coefficient array $\mathsf{st} = (\boldsymbol{s}_{ab})_{a<r,b<c}$. We define the opening relation as follows:

$$\begin{aligned}&\mathcal{R}_{\mathsf{open}}(\mathsf{crs}, \mathsf{com}, \boldsymbol{M}, \mathsf{st}) = 1 \\ &\quad\iff \quad \Pi_{\mathsf{wf}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{wf}}, \mathsf{x}_{\mathsf{wf}}, \pi_{\mathsf{wf}}) = 1 \quad \text{and} \\ &\qquad\qquad \begin{cases} \boldsymbol{A}\boldsymbol{s}_{ab} + M_{ab}\boldsymbol{e}_1 = \boldsymbol{K}_a\boldsymbol{c}_b \pmod q, \\ \|\boldsymbol{s}_{ab}\|_2 \le \beta_s \end{cases} \quad \text{for every } a < r, b < c.\end{aligned} \tag{22}$$

5.2.3 Correctness and security

Remark 2 (Instantiating the well-formedness proof). The relation is a batch of $O(k^2)$ SIS-type relations as described in (22): the witness consists of rc short vectors $\boldsymbol{s}_{ab} \in \mathbb{Z}^m$ with $\boldsymbol{A}_\star \boldsymbol{s}_{ab} = (\boldsymbol{K}_a\boldsymbol{c}_b)_\star \pmod q$, where $(\cdot)_\star$ drops the first coordinate. This is the standard relation targeted by many lattice-based zero-knowledge proofs of knowledge of short preimages [BLNS20, ACK21, BCS21, NS22, ACL+22, BCS23, BS23]. In more detail, we need the following three properties for the NIZK: (i) knowledge soundness for some norm bound $\beta'_s \ge \beta_s$ on the extracted vectors (we discuss this below); (ii) zero-knowledge, which SimCommit uses to simulate π_{wf} for uniformly random commitment vectors; and (iii) a proof length $L_{\mathsf{wf}} = O(k)$, up to factors polynomial in λ, so that sharing costs $O(k^2)$ words. A rewinding extractor suffices, and the argument may be obtained from an interactive protocol by the Fiat–Shamir transform in the random oracle model. A natural candidate satisfying these requirements is LaBRADOR [BS23]. Its principal relation is knowledge of short vectors over $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N+1)$ satisfying dot-product constraints, its proof size is polylogarithmic in the witness size (concretely tens of kilobytes) and independent of rc, and its prover runs in time linear in the witness size, which is $O(k^2 m \log q)$ here. This gives $L_{\mathsf{wf}} = \mathsf{poly}(\lambda)\mathsf{polylog}(k)$, well within the budget. To use it, one could instantiate PowerBASIS over a module lattice, as in the formulation of [FMN24]: $\boldsymbol{A}$ then has entries in $\mathcal{R}_q$, every equation $\boldsymbol{A}_\star \boldsymbol{s}_{ab} = (\boldsymbol{K}_a\boldsymbol{c}_b)_\star$ is a module relation, and the requirement that M_{ab} is a scalar becomes the linear constraint that the first coordinate of $\boldsymbol{K}_a\boldsymbol{c}_b - \boldsymbol{A}\boldsymbol{s}_{ab}$ is a constant polynomial, which LaBRADOR supports as a constraint on constant coefficients. LaBRADOR itself is not zero-knowledge; as its

$\underline{\mathsf{Commit}(\mathsf{crs}, \boldsymbol{M})}$:

1. Require $\boldsymbol{M} \in \mathbb{F}_q^{r \times c}$. For each $b < c$, sample

$$\begin{pmatrix} (\boldsymbol{s}_{ab})_{a<r} \\ \widehat{\boldsymbol{c}}_b \end{pmatrix} \leftarrow\!\!\$\ \mathsf{SamplePre}\left(\boldsymbol{B}_r, \boldsymbol{T}_r, (-M_{ab}\boldsymbol{W}^a\boldsymbol{e}_1)_{a<r}, \sigma\right), \qquad \boldsymbol{c}_b = \boldsymbol{G}\widehat{\boldsymbol{c}}_b.$$

   Abort if some $\|\boldsymbol{s}_{ab}\|_2 > \beta_s$.
2. Set $\mathsf{st} = (\boldsymbol{s}_{ab})_{a<r,b<c}$ and sample

$$\pi_{\mathsf{wf}} \leftarrow\!\!\$\ \Pi_{\mathsf{wf}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{wf}}, \mathsf{x}_{\mathsf{wf}}, \mathsf{st}).$$

   Return $\mathsf{com} = ((\boldsymbol{c}_b)_{b<c}, \pi_{\mathsf{wf}})$ and st.

$\underline{\mathsf{Eval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, (i,j))}$: Require $(i,j) \in \mathcal{D}$, and return

$$y = \boldsymbol{g}_i^\top \boldsymbol{M}\boldsymbol{h}_j \in \mathbb{F}_q, \qquad \boldsymbol{\pi} = \sum_{a<r}\sum_{b<c} g_{ia}h_{jb}\boldsymbol{s}_{ab} \in \mathbb{Z}^m.$$

$\underline{\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i,j), y, \boldsymbol{\pi})}$:

1. Output 0 if $(i,j) \notin \mathcal{D}$, $y \notin \mathbb{F}_q$, or $\boldsymbol{\pi} \notin \mathbb{Z}^m$. Parse $\mathsf{com} = ((\boldsymbol{c}_b)_{b<c}, \pi_{\mathsf{wf}})$.
2. If $\Pi_{\mathsf{wf}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{wf}}, \mathsf{x}_{\mathsf{wf}}, \pi_{\mathsf{wf}}) = 0$, return 0.
3. Return 1 iff

$$\|\boldsymbol{\pi}\|_2 \le B(i,j), \qquad \boldsymbol{A}\boldsymbol{\pi} + y\boldsymbol{e}_1 = \boldsymbol{K}(\boldsymbol{g}_i)\boldsymbol{c}(\boldsymbol{h}_j) \pmod q.$$

Fig. 4. The $\mathsf{Commit}, \mathsf{Eval}$ and Verify algorithms for our commitment scheme $\mathsf{Basis2DLCCom}$.

authors note, zero-knowledge is obtained by composing it with a zero-knowledge inner proof such as the framework of [LNP22], where LaBRADOR proves knowledge of an accepting inner transcript, retaining the polylogarithmic proof size.

Regarding the size of the public parameters, both LNP22 [LNP22] and LaBRADOR [BS23] admit transparent setup with an $O(1)$-size CRS in k, representing public randomness by short seeds and suppressing security-parameter factors. Thus the composed well-formedness proof also has an $O(1)$-size CRS.

Lastly, note that many of the proof systems mentioned above only support a relaxed notion of knowledge soundness, wherein the extracted witnesses may have a larger norm than the original bound (this is denoted by β'_s above). But this can be easily handled by simply replacing β_s with β'_s in the norm bounds of the verification check, and scaling the modulus q and the norm-bound for our PowerBASIS assumption accordingly. Essentially, the reductions of Theorems 3 and 4 will now use the norm bound of the extracted witnesses instead of the original bound.

Theorem 1 (Correctness). *Under the parameter conditions in Section 6.2.2 and completeness of Π_{wf}, the construction satisfies Definition 8.*

Proof. Condition on successful code and lattice setup (note that this fails only with negligible probability). First observe that

$$\boldsymbol{A}\boldsymbol{R}_a = \boldsymbol{G}\boldsymbol{G}^{-1}(\boldsymbol{K}_a\boldsymbol{G}) = \boldsymbol{K}_a\boldsymbol{G}.$$

Therefore $\boldsymbol{B}_r\widetilde{\boldsymbol{R}} = \boldsymbol{G}_r$, so the public preimage trapdoor has the required target. As per the commitment algorithm, for each a, b,

$$\boldsymbol{W}^a\boldsymbol{A}\boldsymbol{s}_{ab} - \boldsymbol{G}\widehat{\boldsymbol{c}}_b = -M_{ab}\boldsymbol{W}^a\boldsymbol{e}_1.$$

$\mathsf{GetProofs}(\mathsf{crs}, \mathsf{com}, \mathsf{dir}, \zeta, \mathcal{P}, V)$:

1. If $(\mathsf{dir}, \zeta, \mathcal{P}, V)$ is not admissible as in the definition, return $\perp$. In particular the source positions are distinct and every source point is in $[k]^2$. For every source tuple $((i,j), y_{ij}, \boldsymbol{\pi}_{ij})$ determined by $\mathcal{P}$ and the selected direction, abort and return $\perp$ if
$$\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i,j), y_{ij}, \boldsymbol{\pi}_{ij}) = 0.$$
2. If $\mathsf{dir} = \mathsf{row}$, write $\zeta = i \in [k]$ and $\mathcal{P} = \{(j, y_j, \boldsymbol{\pi}_j) : j \in J\}$, where $|J| \geq t_c^H$ and $V \subseteq \mathcal{J}$. Compute, over $\mathbb{F}_q$,
$$\widehat{\boldsymbol{u}} = \boldsymbol{D}_H(J)(y_j)_{j\in J}, \qquad \widehat{\boldsymbol{S}} = \boldsymbol{D}_H(J)(\boldsymbol{\pi}_j^\top \bmod q)_{j\in J}.$$
Output
$$\left\{\left(j', \boldsymbol{h}_{j'}^\top \widehat{\boldsymbol{u}}, \mathsf{lift}(\widehat{\boldsymbol{S}}^\top \boldsymbol{h}_{j'})\right) : j' \in V\right\}.$$
3. If $\mathsf{dir} = \mathsf{col}$, write $\zeta = j \in [k]$ and $\mathcal{P} = \{(i, y_i, \boldsymbol{\pi}_i) : i \in I\}$, where $|I| \geq t_c^G$ and $V \subseteq \mathcal{I}$. Compute, over $\mathbb{F}_q$,
$$\widehat{\boldsymbol{v}} = \boldsymbol{D}_G(I)(y_i)_{i\in I}, \qquad \widehat{\boldsymbol{T}} = \boldsymbol{D}_G(I)(\boldsymbol{\pi}_i^\top \bmod q)_{i\in I}.$$
Output
$$\left\{\left(i', \boldsymbol{g}_{i'}^\top \widehat{\boldsymbol{v}}, \mathsf{lift}(\widehat{\boldsymbol{T}}^\top \boldsymbol{g}_{i'})\right) : i' \in V\right\}.$$

Fig. 5. The GetProofs algorithm for our commitment scheme Basis2DLCCom.

Note that Commit aborts unless every sampled $\boldsymbol{s}_{ab}$ has norm at most β_s. By the Gaussian tail bound (7), applied as in (52), this abort occurs with probability at most τ, which is negligible for the parameters of Section 6.2.2. Hence, conditioned on Commit not aborting, all $\boldsymbol{s}_{ab}$ have norm at most β_s. Next, multiplying the above equation by $\boldsymbol{K}_a$ gives the coefficient equation in (22). Hence, st is a valid witness for the relation $\mathcal{R}_{\mathsf{wf}}^{\mathsf{2d}}$. By completeness of Π_{wf}, π_{wf} will pass verification. Thus the full-opening relation holds.

Fix any $(i,j) \in \mathcal{D}$. Summing the coefficient equations with their integer weights $g_{ia}h_{jb}$ gives

$$\begin{aligned} \boldsymbol{A}\boldsymbol{\pi} + (\boldsymbol{g}_i^\top \boldsymbol{M}\boldsymbol{h}_j)\boldsymbol{e}_1 &= \sum_{a,b} g_{ia}h_{jb}\boldsymbol{K}_a\boldsymbol{c}_b \\ &= \left(\sum_a g_{ia}\boldsymbol{K}_a\right)\left(\sum_b h_{jb}\boldsymbol{c}_b\right) = \boldsymbol{K}(\boldsymbol{g}_i)\boldsymbol{c}(\boldsymbol{h}_j). \end{aligned} \tag{23}$$

Furthermore
$$\|\boldsymbol{\pi}\|_2 \leq \sum_{a,b} |g_{ia}|\,|h_{jb}|\,\|\boldsymbol{s}_{ab}\|_2 \leq \|\boldsymbol{g}_i\|_1\|\boldsymbol{h}_j\|_1\beta_s = B(i,j).$$
Hence, an honestly produced opening via the Eval algorithm will pass verification.

For an honest row input for GetProofs, define $\boldsymbol{S}_i \in \mathbb{Z}^{c\times m}$ by $(\boldsymbol{S}_i)_{b,*} = (\sum_a g_{ia}\boldsymbol{s}_{ab})^\top$. The source value column and proof matrix, reduced modulo q, are respectively $(\boldsymbol{H}_{\mathsf{code}})_J\boldsymbol{M}^\top\boldsymbol{g}_i$ and $(\boldsymbol{H}_{\mathsf{code}})_J(\boldsymbol{S}_i \bmod q)$. Applying (15) yields $\widehat{\boldsymbol{u}} = \boldsymbol{M}^\top\boldsymbol{g}_i$ and $\widehat{\boldsymbol{S}} = \boldsymbol{S}_i \bmod q$ over $\mathbb{F}_q$. For every $j' \in V$, the target proof satisfies $\|\boldsymbol{S}_i^\top\boldsymbol{h}_{j'}\|_2 \leq B(i,j') < q/2$, by the norm bound above and the parameter condition $q > 2B_{\max}$ of Section 6.2.2. Hence $\mathsf{lift}(\widehat{\boldsymbol{S}}^\top\boldsymbol{h}_{j'}) = \boldsymbol{S}_i^\top\boldsymbol{h}_{j'}$ exactly over $\mathbb{Z}$. Consequently, the resulting values and proofs at every $j' \in V$ are exactly the output of $\mathsf{Eval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, (i,j'))$ (and hence will pass verification).

Lastly, for a column input to GetProofs, define $\boldsymbol{T}_j \in \mathbb{Z}^{r\times m}$ by $(\boldsymbol{T}_j)_{a,*} = (\sum_b h_{jb}\boldsymbol{s}_{ab})^\top$. The source lists are $(\boldsymbol{G}_{\mathsf{code}})_I\boldsymbol{M}\boldsymbol{h}_j$ and $(\boldsymbol{G}_{\mathsf{code}})_I(\boldsymbol{T}_j \bmod q)$. The decoder recovers $\widehat{\boldsymbol{v}} = \boldsymbol{M}\boldsymbol{h}_j$ and $\widehat{\boldsymbol{T}} = \boldsymbol{T}_j \bmod q$

over $\mathbb{F}_q$, and the same norm bound $\|\boldsymbol{T}_j^\top \boldsymbol{g}_{i'}\|_2 \le B(i',j) < q/2$ gives $\mathsf{lift}(\widehat{\boldsymbol{T}}^\top \boldsymbol{g}_{i'}) = \boldsymbol{T}_j^\top \boldsymbol{g}_{i'}$ exactly, so the output at every $i' \in V$ is exactly equal to its honest evaluation. □

Theorem 2 (Evaluation binding). *Under the PowerBASIS assumption at bound $\beta_{\mathsf{PB}} \ge 2B_{\max}$, the construction satisfies Definition 10.*

Proof. We will construct an adversary $\mathcal{B}$ that breaks PowerBASIS using an adversary $\mathcal{A}$ that breaks evaluation binding. Given the complete correlated PowerBASIS public tuple $(\boldsymbol{A}, \boldsymbol{W}, \boldsymbol{T}_r)$ from the PowerBASIS challenger, the reduction $\mathcal{B}$ samples the code parameters and honest Π_{wf} CRS, and runs the binding adversary. This exactly matches the distribution of the public setup of the construction. Say $\mathcal{A}$ outputs (i,j) with two accepting pairs $(y, \boldsymbol{\pi})$ and $(y', \boldsymbol{\pi}')$. We subtract their verification equations:

$$\boldsymbol{A}(\boldsymbol{\pi} - \boldsymbol{\pi}') = (y' - y)\boldsymbol{e}_1 \pmod q, \qquad \boldsymbol{A}_\star(\boldsymbol{\pi} - \boldsymbol{\pi}') = \boldsymbol{0} \pmod q.$$

Since $y \ne y'$, the vector $\boldsymbol{x} = \boldsymbol{\pi} - \boldsymbol{\pi}'$ is nonzero modulo q. Also $\|\boldsymbol{x}\|_2 \le \|\boldsymbol{\pi}\|_2 + \|\boldsymbol{\pi}'\|_2 \le 2B(i,j) \le 2B_{\max} \le \beta_{\mathsf{PB}}$. Thus, every winning output gives a valid PowerBASIS solution, and $\mathcal{B}$'s success probability equals the adversary's winning probability. □

Theorem 3 (Computational proof canonicality of Basis2DLCCom). *Suppose*

$$2B_{\max} < q, \qquad \beta_{\mathsf{PB}} \ge 2B_{\max},$$

and assume the PowerBASIS assumption for bound β_{PB}. Then, Basis2DLCCom *satisfies Definition 9. More precisely, for every* PPT *adversary $\mathcal{A}$ there is a* PPT *PowerBASIS adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{can}}_{\mathsf{Basis2DLCCom},\mathcal{A}}(\lambda) \le \mathsf{Adv}^{\mathrm{powerbasis}}_{\mathcal{B},q,n-1,m}(\lambda).$$

Proof. The reduction $\mathcal{B}$ receives the complete PowerBASIS tuple, which consists of $\boldsymbol{A}$, $\boldsymbol{W}$, $\boldsymbol{T}_r$, and the associated structured matrix. It independently samples the component codes, packing vectors, and the CRS for Π_{wf}, and gives the resulting Basis2DLCCom CRS to $\mathcal{A}$. This is statistically close to a real CRS.

Suppose $\mathcal{A}$ outputs a winning tuple $(\mathsf{com}, (i,j), y, \boldsymbol{\pi}, \boldsymbol{\pi}')$. Write $C = (\boldsymbol{c}_b)_{b<c}$ for the vector part of the commitment, and recall $\boldsymbol{c}(\boldsymbol{h}_j) = \sum_{b<c} h_{j,b}\boldsymbol{c}_b$. The two accepting verification equations are

$$\boldsymbol{A}\boldsymbol{\pi} + y\boldsymbol{e}_1 = \boldsymbol{K}(\boldsymbol{g}_i)\boldsymbol{c}(\boldsymbol{h}_j) \pmod q, \qquad \boldsymbol{A}\boldsymbol{\pi}' + y\boldsymbol{e}_1 = \boldsymbol{K}(\boldsymbol{g}_i)\boldsymbol{c}(\boldsymbol{h}_j) \pmod q.$$

Subtracting gives

$$\boldsymbol{A}(\boldsymbol{\pi} - \boldsymbol{\pi}') = 0 \pmod q.$$

Consequently, for $\boldsymbol{x} := \boldsymbol{\pi} - \boldsymbol{\pi}'$,

$$\boldsymbol{A}_\star \boldsymbol{x} = 0 \pmod q, \qquad \|\boldsymbol{x}\|_2 \le \|\boldsymbol{\pi}\|_2 + \|\boldsymbol{\pi}'\|_2 \le 2B(i,j) \le 2B_{\max} \le \beta_{\mathsf{PB}},$$

where $\boldsymbol{A}_\star$ denotes the last $n-1$ rows of $\boldsymbol{A}$, which form the underlying PowerBASIS instance.

Since the two proof vectors are different, $\boldsymbol{x} \ne 0$ as an integer vector. Moreover, $\|\boldsymbol{x}\|_2 \le 2B_{\max} < q$. If $\boldsymbol{x}$ were zero modulo q, some nonzero coordinate would have absolute value at least q, contradicting this norm bound. Thus $\boldsymbol{x}$ is also nonzero modulo q. The reduction simply outputs $\boldsymbol{x}$, which solves its PowerBASIS instance. Hence, every winning output of $\mathcal{A}$ therefore gives a valid PowerBASIS solution. □

Theorem 4 (Proof interpolation). *Suppose that Π_{wf} is knowledge sound for the well-formedness relation of the construction and admits a* PPT *extractor. Then, for every* PPT *adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\Pr[\mathsf{Bad}_{\mathsf{proof\text{-}int}}(\mathcal{A})] \le \mathsf{Adv}^{\mathrm{powerbasis}}_{\mathcal{B},q,n-1,m,r,2B_{\max},\sigma_0}(\lambda) + \mathsf{negl}(\lambda), \tag{24}$$

where $\mathsf{Bad}_{\mathsf{proof\text{-}int}}(\mathcal{A})$ is the event that $\mathcal{A}$ outputs a commitment and source openings satisfying all preconditions of GetProofs*, every source opening verifies, and some opening returned by* GetProofs *does not verify or carries a value other than the linear decoding of the source values.*

Consequently, the 2D bounded-encoder linear code commitment satisfies proof interpolation in both directions if $\mathsf{PowerBASIS}_{q,n-1,m,r,2B_{\max},\sigma_0}$ holds.

Proof. The reduction $\mathcal{B}$ receives a PowerBASIS instance

$$\big(\boldsymbol{A}_\star, \boldsymbol{A}^\star, \boldsymbol{W}, \boldsymbol{B}_r, \boldsymbol{T}_r\big), \qquad \boldsymbol{A}^\star = \begin{bmatrix} \boldsymbol{a}^\top \\ \boldsymbol{A}_\star \end{bmatrix} \in \mathbb{Z}_q^{n\times m}.$$

It sets $\boldsymbol{A} \leftarrow \boldsymbol{A}^\star$ and uses the supplied structured matrix $\boldsymbol{B}_r$ and public trapdoor $\boldsymbol{T}_r$ in the commitment setup. It computes the coefficient matrices $\boldsymbol{K}_a$, $a \in [r]$ from $\boldsymbol{W}$ exactly as in the construction. It generates the code parameters according to the honest setup algorithm and runs

$$(\mathsf{crs}_{\mathsf{wf}}, \tau_{\mathsf{ext}}) \leftarrow\!\$\ \Pi_{\mathsf{wf}}.\mathsf{ExtSetup}(1^\lambda).$$

It gives the resulting crs to $\mathcal{A}$. By the extraction-setup indistinguishability guarantee and the distribution of the PowerBASIS instance, this CRS is computationally indistinguishable from an honest CRS.

The adversary outputs a commitment com and the data for a GetProofs invocation: its direction, source indices, source values and proofs, and target indices. The reduction checks all preconditions and runs GetProofs. It aborts if a precondition fails, if any source opening fails verification, or if every returned opening verifies. Thus, every remaining execution has at least one invalid target opening and a nonempty collection of accepting source openings.

Parse

$$\mathsf{com} = \big((\boldsymbol{c}_b)_{b\in[c]}, \pi_{\mathsf{wf}}\big),$$

and let x_{wf} denote the well-formedness statement checked by Verify for this commitment and these public parameters. Since every source opening verifies and the source collection is nonempty,

$$\Pi_{\mathsf{wf}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{wf}}, x_{\mathsf{wf}}, \pi_{\mathsf{wf}}) = 1.$$

The reduction invokes the well-formedness extractor directly:

$$(\boldsymbol{s}^\star_{a,b})_{a\in[r],\,b\in[c]} \leftarrow\!\$\ \Pi_{\mathsf{wf}}.\mathsf{Ext}(\tau_{\mathsf{ext}}, x_{\mathsf{wf}}, \pi_{\mathsf{wf}}).$$

By knowledge soundness for the coefficient-opening relation, except with negligible probability, the extracted vectors satisfy

$$\boldsymbol{s}^\star_{a,b} \in \mathbb{Z}^m, \qquad \|\boldsymbol{s}^\star_{a,b}\| \le \beta_s, \qquad \boldsymbol{A}_\star \boldsymbol{s}^\star_{a,b} = (\boldsymbol{K}_a \boldsymbol{c}_b)_\star \quad (a \in [r],\ b \in [c]). \tag{25}$$

Here, $(\cdot)_\star$ denotes the final $n-1$ coordinates, and matrix equations involving $\boldsymbol{A}$ or $\boldsymbol{A}_\star$ are over $\mathbb{Z}_q$. The reduction checks these conditions and aborts if they fail. In what follows, condition on the fact that they hold.

Define a coefficient matrix $\boldsymbol{M}^\star \in \mathbb{Z}_q^{r\times c}$ by

$$M^\star_{a,b} := \big(\boldsymbol{K}_a \boldsymbol{c}_b - \boldsymbol{A}\boldsymbol{s}^\star_{a,b}\big)_1.$$

Equation (25) says that all remaining coordinates of this difference vanish. Therefore

$$\boldsymbol{A}\boldsymbol{s}^\star_{a,b} + M^\star_{a,b}\boldsymbol{e}_1 = \boldsymbol{K}_a\boldsymbol{c}_b \quad (a\in[r],\ b\in[c]). \tag{26}$$

Together with the norm bounds, these equations show that $\mathsf{st}^\star = (\boldsymbol{s}^\star_{a,b})_{a,b}$ satisfies $\mathcal{R}_{\mathsf{open}}$ for $(\mathsf{com}, \boldsymbol{M}^\star)$.

Next, for every $(i,j)\in\mathcal{D}$, define the canonical value and proof

$$F^\star(i,j) := \boldsymbol{g}_i^\top \boldsymbol{M}^\star \boldsymbol{h}_j, \qquad \boldsymbol{\pi}^\star(i,j) := \sum_{a\in[r]}\sum_{b\in[c]} g_{i,a}h_{j,b}\boldsymbol{s}^\star_{a,b}. \tag{27}$$

Multiplying (26) by $g_{i,a}h_{j,b}$ and summing over a,b gives

$$\boldsymbol{A}\boldsymbol{\pi}^\star(i,j) + F^\star(i,j)\boldsymbol{e}_1 = \boldsymbol{K}(\boldsymbol{g}_i)\widehat{\boldsymbol{c}}(\boldsymbol{h}_j). \tag{28}$$

Moreover, the vectors $\boldsymbol{g}_i$ and $\boldsymbol{h}_j$ have integer entries, so

$$\begin{aligned}\|\boldsymbol{\pi}^\star(i,j)\| &\le \sum_{a\in[r]}\sum_{b\in[c]} |g_{i,a}|\,|h_{j,b}|\,\|\boldsymbol{s}^\star_{a,b}\| \\ &\le \|\boldsymbol{g}_i\|_1\|\boldsymbol{h}_j\|_1\beta_{\boldsymbol{s}} \le B(i,j) \le B_{\max}.\end{aligned} \tag{29}$$

Thus these canonical pairs satisfy the verification equation and the norm check at every permitted index pair.

We next show that any accepting source opening that differs from its canonical pair yields a solution to the PowerBASIS instance. Consider a source tuple $(i,j,y,\boldsymbol{\pi})$ with

$$\mathsf{Verify}(\mathsf{crs},\mathsf{com},(i,j),y,\boldsymbol{\pi}) = 1.$$

Acceptance implies

$$\boldsymbol{A}\boldsymbol{\pi} + y\boldsymbol{e}_1 = \boldsymbol{K}(\boldsymbol{g}_i)\widehat{\boldsymbol{c}}(\boldsymbol{h}_j), \qquad \|\boldsymbol{\pi}\| \le B(i,j). \tag{30}$$

Set

$$\boldsymbol{z} := \boldsymbol{\pi} - \boldsymbol{\pi}^\star(i,j) \in \mathbb{Z}^m.$$

Subtracting (28) from (30) yields

$$\boldsymbol{A}\boldsymbol{z} = (F^\star(i,j) - y)\boldsymbol{e}_1, \qquad \boldsymbol{A}_\star\boldsymbol{z} = \boldsymbol{0}. \tag{31}$$

By (29),

$$\|\boldsymbol{z}\| \le \|\boldsymbol{\pi}\| + \|\boldsymbol{\pi}^\star(i,j)\| \le 2B(i,j) \le 2B_{\max}.$$

If $(y,\boldsymbol{\pi}) \ne (F^\star(i,j),\boldsymbol{\pi}^\star(i,j))$, then $\boldsymbol{z}\ne\boldsymbol{0}$. Indeed, $\boldsymbol{z}=\boldsymbol{0}$ would imply equality of the two proofs and, by the first equation in (31), also $y = F^\star(i,j)$. Consequently, from any noncanonical accepting source, the reduction outputs a vector satisfying

$$\boldsymbol{0}\ne\boldsymbol{z}\in\mathbb{Z}^m, \qquad \|\boldsymbol{z}\| \le 2B_{\max}, \qquad \boldsymbol{A}_\star\boldsymbol{z} = \boldsymbol{0} \pmod q,$$

which solves its PowerBASIS instance. Hence, we condition the rest of the analysis on all accepting source openings being equal to the canonical pair obtained from the extraction.

The values returned by GetProofs are the linear decodings of the source values by construction (Figure 5), so the value clause of Definition 11 never fails. It remains to prove that if every source opening is canonical, then every opening returned by GetProofs verifies. We consider the two directions separately.

Case 1: interpolation in the first (G, or X) direction. Consider an index $j \in \mathcal{I}_H$ for which GetProofs is called. Let $I \subseteq [k]$ be its source set, and write the accepting source pairs as $(y_i, \boldsymbol{\pi}_i)_{i\in I}$. By the above analysis,

$$y_i = \boldsymbol{g}_i^\top \boldsymbol{M}^\star \boldsymbol{h}_j, \qquad \boldsymbol{\pi}_i = \sum_{a\in[r]} \sum_{b\in[c]} g_{i,a} h_{j,b} \boldsymbol{s}^\star_{a,b} \quad (i \in I). \tag{32}$$

For a requested target $i' \in \mathcal{I}_G$, define the row vector

$$(\lambda_i)_{i\in I} := \boldsymbol{g}_{i'}^\top \boldsymbol{D}_G(I).$$

Equation (15) gives the identity

$$\sum_{i\in I} \lambda_i \boldsymbol{g}_i = \boldsymbol{g}_{i'} \pmod q. \tag{33}$$

Since GetProofs decodes with $\boldsymbol{D}_G(I)$ and evaluates the decoded vectors at $\boldsymbol{g}_{i'}$, its returned value and proof are

$$y' := \sum_{i\in I} \lambda_i y_i, \qquad \boldsymbol{\pi}' := \mathsf{lift}\Big(\sum_{i\in I} \lambda_i \boldsymbol{\pi}_i \bmod q\Big).$$

Using (32) and (33), we obtain

$$\begin{aligned} y' &= \left(\sum_{i\in I} \lambda_i \boldsymbol{g}_i\right)^\top \boldsymbol{M}^\star \boldsymbol{h}_j = \boldsymbol{g}_{i'}^\top \boldsymbol{M}^\star \boldsymbol{h}_j = F^\star(i', j), \\ \sum_{i\in I} \lambda_i \boldsymbol{\pi}_i &= \sum_{a\in[r]} \sum_{b\in[c]} \left(\sum_{i\in I} \lambda_i g_{i,a}\right) h_{j,b} \boldsymbol{s}^\star_{a,b} \\ &\equiv \sum_{a\in[r]} \sum_{b\in[c]} g_{i',a} h_{j,b} \boldsymbol{s}^\star_{a,b} = \boldsymbol{\pi}^\star(i', j) \pmod q. \end{aligned} \tag{34}$$

The congruence holds modulo q because the decoder identity (33) holds over $\mathbb{F}_q$. Since $\|\boldsymbol{\pi}^\star(i', j)\| \le B(i', j) < q/2$ by (29) and the parameter condition $q > 2B_{\max}$ of Section 6.2.2, the centered integer representative of the right-hand side is $\boldsymbol{\pi}^\star(i', j)$ itself, and therefore $\boldsymbol{\pi}' = \boldsymbol{\pi}^\star(i', j)$ over $\mathbb{Z}^m$. Hence (28) and (29) give

$$\boldsymbol{A}\boldsymbol{\pi}' + y'\boldsymbol{e}_1 = \boldsymbol{K}(\boldsymbol{g}_{i'})\widehat{\boldsymbol{c}}(\boldsymbol{h}_j), \qquad \|\boldsymbol{\pi}'\| \le B(i', j).$$

The commitment contains the same accepting well-formedness proof. Therefore

$$\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i', j), y', \boldsymbol{\pi}') = 1.$$

This holds for every requested i' and every fixed second index in the batch.

Case 2: interpolation in the second (H, or Y) direction. Fix any first index $i \in \mathcal{I}_G$ for the GetProofs invocation. Let $J \subseteq [k]$ be its source set, and write the accepting source pairs as $(y_j, \boldsymbol{\pi}_j)_{j\in J}$. By the above analysis,

$$y_j = \boldsymbol{g}_i^\top \boldsymbol{M}^\star \boldsymbol{h}_j, \qquad \boldsymbol{\pi}_j = \sum_{a\in[r]} \sum_{b\in[c]} g_{i,a} h_{j,b} \boldsymbol{s}^\star_{a,b} \quad (j \in J). \tag{35}$$

For a requested target $j' \in \mathcal{I}_H$, define

$$(\mu_j)_{j\in J} := \boldsymbol{h}_{j'}^\top \boldsymbol{D}_H(J).$$

Equation (15) implies, over $\mathbb{F}_q$,

$$\sum_{j\in J} \mu_j \boldsymbol{h}_j = \boldsymbol{h}_{j'} \pmod q. \tag{36}$$

The value and proof returned by GetProofs are

$$y' := \sum_{j\in J} \mu_j y_j, \qquad \boldsymbol{\pi}' := \mathsf{lift}\Big(\sum_{j\in J} \mu_j \boldsymbol{\pi}_j \bmod q\Big).$$

Consequently,

$$\begin{aligned} y' &= \boldsymbol{g}_i^\top \boldsymbol{M}^\star \left(\sum_{j\in J} \mu_j \boldsymbol{h}_j\right) = \boldsymbol{g}_i^\top \boldsymbol{M}^\star \boldsymbol{h}_{j'} = F^\star(i, j'), \\ \sum_{j\in J} \mu_j \boldsymbol{\pi}_j &= \sum_{a\in[r]} \sum_{b\in[c]} g_{i,a} \left(\sum_{j\in J} \mu_j h_{j,b}\right) \boldsymbol{s}^\star_{a,b} \\ &\equiv \sum_{a\in[r]} \sum_{b\in[c]} g_{i,a} h_{j',b} \boldsymbol{s}^\star_{a,b} = \boldsymbol{\pi}^\star(i, j') \pmod q. \end{aligned} \tag{37}$$

Again, the congruence holds modulo q, and since $\|\boldsymbol{\pi}^\star(i, j')\| \le B(i, j') < q/2$, the centered integer representative of the right-hand side is $\boldsymbol{\pi}^\star(i, j')$, so $\boldsymbol{\pi}' = \boldsymbol{\pi}^\star(i, j')$ over $\mathbb{Z}^m$. The canonical equation, the bound $\|\boldsymbol{\pi}'\| \le B(i, j')$, and the same accepting well-formedness proof imply

$$\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, (i, j'), y', \boldsymbol{\pi}') = 1.$$

This holds for every requested j' and every fixed first index in the batch.

Thus, in either direction, successful extraction from the well-formedness proof together with canonical source openings (based on the PowerBASIS assumption) force every target opening to verify. In particular, on an execution with an invalid target proof and successful extraction, at least one source must be noncanonical, and the reduction outputs a PowerBASIS solution as above. Therefore, in the experiment using the extraction CRS,

$$\begin{aligned} \Pr[\mathsf{Bad}_{\mathsf{proof\text{-}int}}(\mathcal{A})] \le{}& \Pr[\mathcal{B} \text{ solves its PowerBASIS instance}] \\ &+ \Pr[\text{an accepting well-formedness proof fails extraction}]. \end{aligned}$$

Knowledge soundness makes the second term negligible. This proves the theorem.

□

Theorem 5 (One-shot adaptive simulation hiding). *Under the parameters specified in Section 6.2.2, and in particular $0 < \varepsilon \le 1/4$ and $\sigma \ge K_{\mathsf{hid}}\eta_\varepsilon(\Lambda)$ (where $K_{\mathsf{hid}} = B_G B_H \sqrt{rc}$), if Π_{wf} is zero-knowledge, then the commitment scheme* Basis2DLCCom *satisfies Definition 12. Namely, for any adversary, its distinguishing advantage is at most*

$$\varepsilon_{\mathsf{zk}} + \varepsilon_{\mathsf{code}} + \varepsilon_{\mathsf{mask}} + \frac{2(c+\ell)\varepsilon}{1-\varepsilon} + 2\tau + (c+rc)\,\varepsilon_{\mathsf{samp}}. \tag{38}$$

Here $\varepsilon_{\mathsf{zk}}$ is the advantage for replacing the argument setup and proof by their simulated versions, and

$$\tau = rc\,\frac{1+\varepsilon}{1-\varepsilon}\,2^{-m}$$

bounds the probability that one of the rc coefficient openings sampled by Commit*, or by the simulator, exceeds its norm bound β_s. Moreover, $\varepsilon_{\mathsf{code}}$ and $\varepsilon_{\mathsf{mask}}$ bound the failure of the code and mask setup (Lemma 1), and $\varepsilon_{\mathsf{samp}}$ is the statistical error of one* SamplePre *call;* Commit *makes c such calls and the simulator makes rc. With the parameters in Section 6.2.2, this advantage is negligible in λ.*

Before proving the theorem, we state and prove some helper lemmas.

The following lemma generalizes [ANP23, Claim 4.1] to our two-dimensional setting.

Lemma 1 (Short packing directions). *Fix the two integer encoder matrices $(\boldsymbol{G}_{\mathsf{code}}, \boldsymbol{H}_{\mathsf{code}})$. Sample every coordinate of $\boldsymbol{b}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_\ell$ independently and uniformly from $\{0, \ldots, D-1\}$, with $2 \le D \le q$. Let $B_G, B_H \ge 1$ be integers with $q > 2\max\{B_G, B_H\}$, and set*

$$\begin{aligned} N_G &= \frac{(B_G+1)^r}{(2B_G L_G + 1)^f}, \\ N_H &= \frac{(B_H+1)^c}{(2B_H L_H + 1)^f \big(2B_H c(D-1)+1\big)^{\ell-1}}, \\ N_f &= \sum_{j=0}^{f} \binom{k}{j} \le 2^k. \end{aligned} \tag{39}$$

Here L_G, L_H denote the max row norm for $\boldsymbol{G}_{\mathsf{code}}, \boldsymbol{H}_{\mathsf{code}}$ respectively (as defined in (18)). Assume $N_G, N_H > 1$ and $0 \le f \le k$. Except with probability at most

$$\varepsilon_{\mathsf{mask}} = \min\left\{1,\ N_f\left(D^{-\log_q N_G} + \ell D^{-\log_q N_H}\right)\right\}, \tag{40}$$

the following statements hold simultaneously for every $J \subseteq [k]$ with $|J| \le f$ and every $t \in [\ell]$: there exist $\boldsymbol{u}_J \in \mathbb{Z}^r$ and $\boldsymbol{v}_{J,t} \in \mathbb{Z}^c$ with

$$\begin{aligned} (\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{u}_J &= \boldsymbol{0}, & \boldsymbol{b}_0^\top \boldsymbol{u}_J &\neq 0 \pmod q, & \|\boldsymbol{u}_J\|_\infty &\le B_G, \\ (\boldsymbol{H}_{\mathsf{code}})_J \boldsymbol{v}_{J,t} &= \boldsymbol{0}, & \boldsymbol{a}_t^\top \boldsymbol{v}_{J,t} &\neq 0 \pmod q, & \|\boldsymbol{v}_{J,t}\|_\infty &\le B_H, \\ \boldsymbol{a}_u^\top \boldsymbol{v}_{J,t} &= 0 \quad (\forall\ u \neq t). \end{aligned} \tag{41}$$

All zero equalities hold over $\mathbb{Z}$. In particular, the matrix $\boldsymbol{U}_{J,t} = \boldsymbol{u}_J \boldsymbol{v}_{J,t}^\top$ satisfies $\|\operatorname{vec}(\boldsymbol{U}_{J,t})\|_2 \le K_{\mathsf{hid}} := B_G B_H \sqrt{rc}$.

Proof. **Count short kernel vectors.** Consider any integer matrix $\boldsymbol{P} \in \mathbb{Z}^{t_0 \times d}$ whose i-th row has ℓ_1-norm at most e_i. For an integer $B \geq 1$, the i-th coordinate of $\boldsymbol{P}\boldsymbol{x}$, for $\boldsymbol{x} \in \{0, \ldots, B\}^d$, lies in $[-Be_i, Be_i]$. Thus, this map has at most $\prod_{i=1}^{t_0}(2Be_i + 1)$ outputs. Some output has at least

$$N(\boldsymbol{P}, B) = \frac{(B+1)^d}{\prod_{i=1}^{t_0}(2Be_i + 1)}$$

preimages. Subtracting a fixed preimage from every other one produces at least $N(\boldsymbol{P}, B) - 1$ distinct nonzero vectors in

$$\mathcal{V} = \{\boldsymbol{v} \in \mathbb{Z}^d : \|\boldsymbol{v}\|_\infty \leq B,\ \boldsymbol{P}\boldsymbol{v} = \boldsymbol{0},\ \boldsymbol{v} \neq \boldsymbol{0}\}.$$

This generalizes the pigeonhole step of [ANP23]'s Claim 4.1.

Choose independent directions. Suppose $q > 2B$. Reduction modulo q is injective on $[-B, B]^d$, and no nonzero vector in this box reduces to zero. Choose a maximal set $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_h \in \mathcal{V}$ independent over $\mathbb{F}_q$. Its span contains every vector of $\mathcal{V}$ modulo q, so

$$q^h - 1 \geq |\mathcal{V}| \geq N(\boldsymbol{P}, B) - 1 \quad \Longrightarrow \quad h \geq \log_q N(\boldsymbol{P}, B).$$

For an independent $\boldsymbol{a} \leftarrow\!\$\ \{0, \ldots, D-1\}^d$, failure to find a direction with $\boldsymbol{a}^\top \boldsymbol{v} \neq 0 \pmod q$ implies $\boldsymbol{a}^\top \boldsymbol{v}_j = 0 \pmod q$ for every $j \in [h]$. These are h independent equations. Fix all nonpivot coordinates of $\boldsymbol{a}$. The equations uniquely determine the remaining h coordinates over $\mathbb{F}_q$. Each required value has probability at most $1/D$, and those coordinates are independent. Hence

$$\Pr[\boldsymbol{a}^\top \boldsymbol{v} = 0 \pmod q \text{ for all } \boldsymbol{v} \in \mathcal{V}] \leq D^{-h} \leq D^{-\log_q N(\boldsymbol{P}, B)}. \tag{42}$$

(For $D = q$ this becomes $1/N(\boldsymbol{P}, B)$, as in [ANP23].)

Apply the count to the two packing directions. Fix J. For the outer direction, take $\boldsymbol{P} = (\boldsymbol{G}_{\mathsf{code}})_J$, $B = B_G$, and $\boldsymbol{a} = \boldsymbol{b}_0$. Since $|J| \leq f$ and each row norm is at most L_G, $N(\boldsymbol{P}, B_G) \geq N_G$. Equation (42) then bounds failure by $D^{-\log_q N_G}$.

For the inner direction of a fixed t, first fix all masks except $\boldsymbol{a}_t$, and take

$$\boldsymbol{P} = \begin{pmatrix} (\boldsymbol{H}_{\mathsf{code}})_J \\ (\boldsymbol{a}_u^\top)_{u \in [\ell] \setminus \{t\}} \end{pmatrix}, \qquad B = B_H.$$

The added rows have norm at most $c(D-1)$. Thus $N(\boldsymbol{P}, B_H) \geq N_H$, while $\boldsymbol{a}_t$ is still independently uniform. Hence the failure probability is at most $D^{-\log_q N_H}$. The integer kernel conditions are exactly those in (41).

Lastly, we union bound over the N_f possible sets and the ℓ inner targets to obtain (40). Finally,

$$\|\operatorname{vec}(\boldsymbol{u}_J \boldsymbol{v}_{J,t}^\top)\|_2 = \|\boldsymbol{u}_J\|_2 \|\boldsymbol{v}_{J,t}\|_2 \leq B_G\sqrt{r}\, B_H\sqrt{c}.$$

Note that this lemma only proves existence of such vectors, and this will be enough for our proofs to go through. □

For the rest of the analysis, a *good mask setup* means that all the conclusions of Lemma 1 hold.

Lemma 2 (Scalar packing privacy). *Fix good codes and masks, a subset $J \subseteq [k]$ of size $\leq f$, and $T \subseteq [\ell]$ of size $< \ell$. Let $\mathcal{L}_{J,T}$ be the scalar leakage map in (16). If $\boldsymbol{s}, \boldsymbol{s}' \in \mathbb{F}_q^\ell$ agree on T and $\boldsymbol{M} \leftarrow\!\$\ \mathbb{F}_q^{r \times c}$, then the following distributions are identical:*

$$\big(\boldsymbol{s} + \mathsf{h}(\boldsymbol{M}), \mathcal{L}_{J,T}(\boldsymbol{M})\big) \ \stackrel{d}{=}\ \big(\boldsymbol{s}' + \mathsf{h}(\boldsymbol{M}), \mathcal{L}_{J,T}(\boldsymbol{M})\big). \tag{43}$$

Moreover, over $\mathbb{F}_q$,

$$\begin{aligned} \boldsymbol{b}_0^\top &\notin \mathrm{rowspan}((\boldsymbol{G}_{\mathsf{code}})_J), \\ \mathrm{rank}\begin{pmatrix}(\boldsymbol{H}_{\mathsf{code}})_J \\ (\boldsymbol{a}_t^\top)_{t\in[\ell]}\end{pmatrix} &= \mathrm{rank}((\boldsymbol{H}_{\mathsf{code}})_J) + \ell. \end{aligned} \tag{44}$$

Proof. We prove privacy by fixing the leakage and showing that the remaining masks are jointly uniform. This is the packed-matrix analogue of the linear-independence argument in [ANP23, Claim 3.3].

Fix J, T. For each $t \notin T$, let

$$\boldsymbol{U}_t := \boldsymbol{u}_J \boldsymbol{v}_{J,t}^\top, \qquad \kappa_t := \mathsf{h}_t(\boldsymbol{U}_t) = (\boldsymbol{b}_0^\top \boldsymbol{u}_J)(\boldsymbol{a}_t^\top \boldsymbol{v}_{J,t}).$$

By Lemma 1, $\kappa_t \neq 0 \pmod q$, and

$$\begin{aligned} &(\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{U}_t = \boldsymbol{0}, \qquad \boldsymbol{U}_t (\boldsymbol{H}_{\mathsf{code}})_J^\top = \boldsymbol{0}, \\ &\boldsymbol{U}_t \boldsymbol{a}_u = \boldsymbol{0} \quad (u \neq t), \qquad \mathcal{L}_{J,T}(\boldsymbol{U}_t) = 0. \end{aligned} \tag{45}$$

These vanishing identities hold over the integers. Thus $\boldsymbol{U}_t$ preserves the leakage $\mathcal{L}_{J,T}$ and changes only mask t.

More precisely, for any choice of increments $\delta = (\delta_t)_{t\notin T}$, define

$$\Phi_\delta(\boldsymbol{X}) := \boldsymbol{X} + \sum_{t\notin T} \delta_t \kappa_t^{-1} \boldsymbol{U}_t.$$

Each κ_t is a nonzero scalar modulo q, so its inverse is well defined. By linearity,

$$\mathcal{L}_{J,T}(\Phi_\delta(\boldsymbol{X})) = \mathcal{L}_{J,T}(\boldsymbol{X}), \qquad \mathsf{h}_t(\Phi_\delta(\boldsymbol{X})) = \begin{cases} \mathsf{h}_t(\boldsymbol{X}) + \delta_t, & t \notin T, \\ \mathsf{h}_t(\boldsymbol{X}), & t \in T. \end{cases}$$

Moreover, Φ_δ is a bijection, with inverse $\Phi_{-\delta}$.

Now fix any leakage value E that is attainable, i.e. there exists some matrix $\boldsymbol{M}_0$ s.t. $\mathcal{L}_{J,T}(\boldsymbol{M}_0) = E$. For every $t \in T$, the leakage determines both

$$(\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{M}\boldsymbol{a}_t \qquad \text{and} \qquad (\boldsymbol{G}_{\mathsf{code}})_{\bar{J}} \boldsymbol{M}\boldsymbol{a}_t.$$

Combining these gives $\boldsymbol{G}_{\mathsf{code}} \boldsymbol{M}\boldsymbol{a}_t$. Full-word decoding recovers $\boldsymbol{M}\boldsymbol{a}_t$, and hence determines $\mathsf{h}_t(\boldsymbol{M}) = \boldsymbol{b}_0^\top \boldsymbol{M}\boldsymbol{a}_t$. Therefore, conditioning on leakage E fixes all masks indexed by T.

For the remaining masks, let $\boldsymbol{d} = (d_t)_{t\notin T}$ be any assignment and define

$$\mathcal{F}_{E,\boldsymbol{d}} := \{\boldsymbol{X} : \mathcal{L}_{J,T}(\boldsymbol{X}) = E, \quad \mathsf{h}_t(\boldsymbol{X}) = d_t \text{ for every } t \notin T\}. \tag{46}$$

Since E is attainable, choose any $\boldsymbol{X}_0$ with $\mathcal{L}_{J,T}(\boldsymbol{X}_0) = E$. Taking

$$\delta_t = d_t - \mathsf{h}_t(\boldsymbol{X}_0) \qquad (t \notin T)$$

gives $\Phi_\delta(\boldsymbol{X}_0) \in \mathcal{F}_{E,\boldsymbol{d}}$. Thus every set in (46) is nonempty.

Furthermore, for any two assignments $\boldsymbol{d}, \boldsymbol{d}'$, the map

$$\Phi_{\boldsymbol{d}'-\boldsymbol{d}} : \mathcal{F}_{E,\boldsymbol{d}} \longrightarrow \mathcal{F}_{E,\boldsymbol{d}'}$$

is a bijection, with inverse $\Phi_{\boldsymbol{d}-\boldsymbol{d}'}$. Hence these sets all have the same cardinality. Since $\boldsymbol{M}$ is uniform, its conditional distribution given $\mathcal{L}_{J,T}(\boldsymbol{M}) = E$ is uniform on the matrices with leakage E. The sets $\mathcal{F}_{E,\boldsymbol{d}}$ partition those matrices into equally sized nonempty sets. Consequently, $(\mathsf{h}_t(\boldsymbol{M}))_{t\notin T}$ is jointly uniform under this conditioning.

Let $\boldsymbol{s}, \boldsymbol{s}'$ agree on T. Conditioned on leakage E, the pads $s_t + \mathsf{h}_t(\boldsymbol{M})$ and $s'_t + \mathsf{h}_t(\boldsymbol{M})$ agree for $t \in T$. For $t \notin T$, the masks are jointly uniform, so adding either secret vector gives the same distribution. Thus the conditional scalar views are identically distributed for every attainable E. The leakage itself depends only on $\boldsymbol{M}$, so averaging over E gives

$$\big(\mathcal{L}_{J,T}(\boldsymbol{M}),\, \boldsymbol{s} + \mathsf{h}(\boldsymbol{M})\big) \overset{d}{=} \big(\mathcal{L}_{J,T}(\boldsymbol{M}),\, \boldsymbol{s}' + \mathsf{h}(\boldsymbol{M})\big).$$

This proves (43).

It remains to prove the rank assertions over $\mathbb{F}_q$. Every vector in the row span of $(\boldsymbol{G}_{\mathsf{code}})_J$ has zero inner product with $\boldsymbol{u}_J$, whereas $\boldsymbol{b}_0^\top \boldsymbol{u}_J \neq 0 \pmod q$. Therefore $\boldsymbol{b}_0^\top$ is outside that row span, so adjoining it increases the rank by one.

For the inner rank assertion, suppose that

$$\sum_{t\in[\ell]} \alpha_t \boldsymbol{a}_t^\top \in \mathrm{rowspan}\big((\boldsymbol{H}_{\mathsf{code}})_J\big), \qquad \alpha_t \in \mathbb{F}_q.$$

For each $u \in [\ell]$, multiply by $\boldsymbol{v}_{J,u}$. The short-direction identities give

$$0 = \left(\sum_{t\in[\ell]} \alpha_t \boldsymbol{a}_t^\top\right) \boldsymbol{v}_{J,u} = \alpha_u \boldsymbol{a}_u^\top \boldsymbol{v}_{J,u}.$$

Since $\boldsymbol{a}_u^\top \boldsymbol{v}_{J,u} \neq 0 \pmod q$, we obtain $\alpha_u = 0$. Thus all coefficients vanish. The packing rows are therefore linearly independent modulo the row span of $(\boldsymbol{H}_{\mathsf{code}})_J$, so adjoining them increases the rank by ℓ, as claimed. □

Let $P = rc$, and flatten matrices by stacking their rows. Represent the state by $\boldsymbol{S} \in \mathbb{Z}^{P\times m}$, whose row indexed by (a, b) is $\boldsymbol{s}_{ab}^\top$. Thus applying an integer matrix $\boldsymbol{L}_0 \in \mathbb{Z}^{d_L\times P}$ to the state means $\boldsymbol{S} \mapsto \boldsymbol{L}_0\boldsymbol{S}$, applying the same combinations to all m columns. On the flattened state this is $\boldsymbol{L}_0 \otimes \boldsymbol{I}_m$.

Put $\Lambda = \Lambda_q^\perp(\boldsymbol{A})$. For fixed $C = (\boldsymbol{c}_b)_{b<c}$ and $\boldsymbol{M}$, let

$$\Omega_{C,\boldsymbol{M}} = \left\{\boldsymbol{S} \in \mathbb{Z}^{P\times m} :\ \boldsymbol{A}\boldsymbol{s}_{ab} = \boldsymbol{K}_a\boldsymbol{c}_b - M_{ab}\boldsymbol{e}_1 \pmod q\ (a < r, b < c)\right\}. \tag{47}$$

In the flattened representation this is a coset of $\Lambda^{\oplus P}$. It imposes only the opening equations; the norm restriction is imposed separately below. Gaussian weights use the Frobenius norm on $\boldsymbol{S}$, equivalently the Euclidean norm of its flattened vector.

The next lemma is the crucial step that allows us to simulate opening proofs for any adversarially-chosen subset of points.

Lemma 3 (One short direction hides a joint proof projection). *Assume $\boldsymbol{A}$ is surjective over $\mathbb{F}_q$. Fix $C, \boldsymbol{M}$ and $\boldsymbol{M}' = \boldsymbol{M} + \delta\boldsymbol{U} \pmod q$, where $\delta \in \mathbb{F}_q$, $\boldsymbol{U} \in \mathbb{Z}^{r\times c}$, and $\boldsymbol{w} = \mathrm{vec}(\boldsymbol{U})$ is primitive, meaning that its entries have greatest common divisor 1. Let $\boldsymbol{L}_0 \in \mathbb{Z}^{d_L\times P}$ satisfy $\boldsymbol{L}_0\boldsymbol{w} = \boldsymbol{0}$ over $\mathbb{Z}$. Independently define*

$$\boldsymbol{S}_{\boldsymbol{M}} \leftarrow\!\$\ D_{\Omega_{C,\boldsymbol{M}},\sigma}, \qquad \boldsymbol{S}_{\boldsymbol{M}'} \leftarrow\!\$\ D_{\Omega_{C,\boldsymbol{M}'},\sigma}.$$

If $0 < \varepsilon < 1$ *and* $\sigma \geq \|\boldsymbol{w}\|_2 \eta_\varepsilon(\Lambda)$*, then*

$$\Delta(\boldsymbol{L}_0 \boldsymbol{S}_{\boldsymbol{M}}, \boldsymbol{L}_0 \boldsymbol{S}_{\boldsymbol{M}'}) \leq \frac{2\varepsilon}{1-\varepsilon}. \tag{48}$$

Here, $\boldsymbol{L}_0 \boldsymbol{S}_{\boldsymbol{M}}$ *denotes the opening proofs corresponding to the leakage map* $\boldsymbol{L}_0$*. The two distributions of leakage have the same support, and on that support their probability ratios lie in* $[\kappa^{-1}, \kappa]$*, where* $\kappa = ((1+\varepsilon)/(1-\varepsilon))^2$*. Moreover, let* $\mathcal{B} = \{\boldsymbol{S} : \|\boldsymbol{s}_{ab}\|_2 \leq \beta_s \ \forall a, b\}$*. Then,*

$$\Delta\left(\boldsymbol{L}_0(\boldsymbol{S}_{\boldsymbol{M}} \mid \mathcal{B}), \boldsymbol{L}_0(\boldsymbol{S}_{\boldsymbol{M}'} \mid \mathcal{B})\right) \leq \frac{2\varepsilon}{1-\varepsilon} + 2\tau \tag{49}$$

where

$$\beta_s = 2\sigma\sqrt{m + \lambda + \ln(2P)}, \qquad \tau = P\frac{1+\varepsilon}{1-\varepsilon}\, 2^{-m}.$$

Proof. **The two opening-equation cosets have the same visible support.** Choose any $\boldsymbol{x}_0 \in \mathbb{Z}^m$ with $\boldsymbol{A}\boldsymbol{x}_0 = -\delta \boldsymbol{e}_1 \pmod q$, which exists by surjectivity (note $\boldsymbol{x}_0$ does not need to be low-norm). Then

$$\Omega_{C,\boldsymbol{M}'} = \Omega_{C,\boldsymbol{M}} + \boldsymbol{w}\boldsymbol{x}_0^\top. \tag{50}$$

Indeed, adding $U_{ab}\boldsymbol{x}_0$ to the coefficient opening at position (a, b) changes its syndrome by $-\delta U_{ab}\boldsymbol{e}_1$, as required for $\boldsymbol{M}'$; subtraction gives the reverse inclusion. Moreover, the leakage is unchanged because $\boldsymbol{L}_0(\boldsymbol{w}\boldsymbol{x}_0^\top) = (\boldsymbol{L}_0\boldsymbol{w})\boldsymbol{x}_0^\top = \boldsymbol{0}$. (Note that this translation may not satisfy the norm bound or preserve individual Gaussian probabilities but that is okay.)

Expose the component perpendicular to the hidden direction. Let

$$\mathcal{U}_{\boldsymbol{w}} = \{\boldsymbol{w}\boldsymbol{y}^\top : \boldsymbol{y} \in \mathbb{R}^m\} \subseteq \mathbb{R}^{P \times m}.$$

Let $\boldsymbol{Z}$ be the orthogonal projection of $\boldsymbol{S}$ onto $\mathcal{U}_{\boldsymbol{w}}^\perp$. Explicitly,

$$\boldsymbol{y}^\top = \frac{\boldsymbol{w}^\top \boldsymbol{S}}{\|\boldsymbol{w}\|_2^2}, \qquad \boldsymbol{Z} = \boldsymbol{S} - \boldsymbol{w}\boldsymbol{y}^\top.$$

Observe that $\boldsymbol{L}_0\boldsymbol{S}$ is determined completely by this projection: $\boldsymbol{L}_0\boldsymbol{S} = \boldsymbol{L}_0\boldsymbol{Z}$, since $\boldsymbol{L}_0\boldsymbol{w} = 0$. Hence it suffices to compare the statistical distance between $\boldsymbol{Z}$.

Now observe that the translation in Equation (50) modifies $\boldsymbol{S}$ to $\boldsymbol{S} + \boldsymbol{w}\boldsymbol{x}_0^\top$, hence, their projections onto $\mathcal{U}_{\boldsymbol{w}}^\perp$ are equal. In other words, both sets $\Omega_{C,\boldsymbol{M}}$ and $\Omega_{C,\boldsymbol{M}'}$ have the same set $\mathcal{Z}$ of possible projections. Additionally, write $S = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_P)$ in integer coordinates and set $d = \|\boldsymbol{w}\|_2^2 = \sum_{t=1}^P w_t^2$. Its projection $\boldsymbol{Z}$ has blocks

$$\boldsymbol{Z}_i = \boldsymbol{s}_i - \frac{w_i}{d}\sum_{t=1}^P w_t \boldsymbol{s}_t.$$

Since d is a positive integer, every coordinate of $\boldsymbol{Z}$ is an integer multiple of $1/d$. Thus $\mathcal{Z}$ lies on a fixed grid and is discrete.

Identify all hidden completions of a fixed projection. For any $\boldsymbol{Z} = \boldsymbol{Z}_0 \in \mathcal{Z}$, choose any $\boldsymbol{S}_0 \in \Omega_{C,\boldsymbol{M}}$ with that projection. Then, $\boldsymbol{S}_0 = \boldsymbol{Z}_0 + \boldsymbol{c}_{\boldsymbol{M}}(\boldsymbol{Z}_0)$, with $\boldsymbol{c}_{\boldsymbol{M}}(\boldsymbol{Z}_0) \in \mathcal{U}_{\boldsymbol{w}}$. Next, for any $\boldsymbol{S}$ to have the same projection as $\boldsymbol{S}_0$, we must have:

– $\boldsymbol{A}(\boldsymbol{S}-\boldsymbol{S}_0)=0$, since both $\boldsymbol{S},\boldsymbol{S}_0$ are in $\Omega_{C,\boldsymbol{M}}$. Formally, $\boldsymbol{S}-\boldsymbol{S}_0\in\Lambda^{\oplus P}$, where

$$\Lambda=\{\boldsymbol{x}\in\mathbb{Z}^m:\boldsymbol{A}\boldsymbol{x}=0 \bmod q\}$$

– $\boldsymbol{S}-\boldsymbol{S}_0$ must be in $\mathcal{U}_{\boldsymbol{w}}$ since $\boldsymbol{S}_0-\boldsymbol{w}\boldsymbol{y}_0^\top=\boldsymbol{Z}_0=\boldsymbol{S}-\boldsymbol{w}\boldsymbol{y}^\top$ for $\boldsymbol{y}_0^\top,\boldsymbol{y}^\top$ as defined above.

Hence, we can write the entire set of $\boldsymbol{S}$ values that project to $\boldsymbol{Z}_0$ as follows:

$$\boldsymbol{Z}_0+\boldsymbol{c}_{\boldsymbol{M}}(\boldsymbol{Z}_0)+\Gamma,\qquad \Gamma=\Lambda^{\oplus P}\cap\mathcal{U}_{\boldsymbol{w}}.$$

We claim $\Gamma=\{\boldsymbol{w}\boldsymbol{x}^\top:\boldsymbol{x}\in\Lambda\}$. To see why, every vector in $\mathcal{U}_{\boldsymbol{w}}$ has the form

$$\boldsymbol{w}\otimes\boldsymbol{y}=(w_1\boldsymbol{y},\ldots,w_P\boldsymbol{y}),\qquad \boldsymbol{y}\in\mathbb{R}^m.$$

For this vector to also belong to $\Lambda^{\oplus P}$, we need $w_i\boldsymbol{y}\in\Lambda$ for every $i\in[P]$.

Since $\boldsymbol{w}$ is primitive, its entries have gcd 1 (Note that, even if $\boldsymbol{w}$ were not primitive, we can take the gcd g of all entries in $\boldsymbol{w}$, and replace δ by $\delta\cdot g$). Hence there exist integers $b_1,\ldots,b_P$ such that

$$\sum_{i=1}^{P}b_iw_i=1.$$

Consequently,

$$\sum_{i=1}^{P}b_i(w_i\boldsymbol{y})=\boldsymbol{y}\in\Lambda,$$

because a lattice is closed under integer linear combinations.

Conversely, if $\boldsymbol{y}\in\Lambda$, then $w_i\boldsymbol{y}\in\Lambda$ for every $i\in[P]$, since each w_i is an integer. Therefore,

$$\Gamma=\Lambda^{\oplus P}\cap\mathcal{U}_{\boldsymbol{w}}=\boldsymbol{w}\otimes\Lambda.$$

Crucially, this means that the lattice of allowed hidden differences is the same for both matrices. Changing $\boldsymbol{M}$ to $\boldsymbol{M}'$ only changes the shift $\boldsymbol{c}_{\boldsymbol{M}}(\boldsymbol{Z}_0)$ to $\boldsymbol{c}_{\boldsymbol{M}'}(\boldsymbol{Z}_0)$.

Sum Gaussian weights over the hidden completions. Now, let us try to measure the distribution of $\boldsymbol{Z}$ when $\boldsymbol{S}$ is sampled from a Gaussian distribution over $\Omega_{C,\boldsymbol{M}}$. A state $\boldsymbol{S}$ receives unnormalized Gaussian weight

$$\exp\left(-\frac{\pi\|\boldsymbol{S}\|^2}{\sigma^2}\right).$$

To find the weight associated with observing $\boldsymbol{Z}_0$, we sum over all possible states $\boldsymbol{S}$ that project to $\boldsymbol{Z}_0$:

$$W_{\boldsymbol{M}}(\boldsymbol{Z}_0)=\sum_{\gamma\in\Gamma}\exp\left(-\frac{\pi\|\boldsymbol{Z}_0+\boldsymbol{c}_{\boldsymbol{M}}+\gamma\|^2}{\sigma^2}\right).$$

Since $\boldsymbol{Z}_0\perp\mathcal{U}_{\boldsymbol{w}}$ and $\boldsymbol{c}_{\boldsymbol{M}}+\gamma\in\mathcal{U}_{\boldsymbol{w}}$, Pythagoras gives

$$\|\boldsymbol{Z}_0+\boldsymbol{c}_{\boldsymbol{M}}+\gamma\|^2=\|\boldsymbol{Z}_0\|^2+\|\boldsymbol{c}_{\boldsymbol{M}}+\gamma\|^2.$$

The factor involving $\boldsymbol{Z}_0$ therefore factors out of the sum:

$$W_{\boldsymbol{M}}(\boldsymbol{Z}_0)=\underbrace{\exp\left(-\frac{\pi\|\boldsymbol{Z}_0\|^2}{\sigma^2}\right)}_{\text{weight from the visible component}}\cdot\underbrace{\rho_\sigma(\boldsymbol{c}_{\boldsymbol{M}}+\Gamma)}_{\text{summed weight of hidden terms}}.$$

Here,

$$\rho_\sigma(\boldsymbol{c}+\Gamma) := \sum_{\gamma\in\Gamma} \exp\left(-\frac{\pi\|\boldsymbol{c}+\gamma\|^2}{\sigma^2}\right).$$

For the same projection $\boldsymbol{Z}_0$, the first factor is identical under $\boldsymbol{M}$ and $\boldsymbol{M}'$. Hence, we only need to compare the second factors to compare the probability mass of $\boldsymbol{Z}_0$ under the Gaussian distributions over $\Omega_{C,\boldsymbol{M}}$ and $\Omega_{C,\boldsymbol{M}'}$.

The map $\boldsymbol{x} \mapsto \boldsymbol{w}\boldsymbol{x}^\top$ multiplies lengths by $\|\boldsymbol{w}\|_2$. Thus Γ is a copy of Λ with every length multiplied by $\|\boldsymbol{w}\|_2$. The smoothing parameter scales by the same factor:

$$\eta_\varepsilon(\Gamma) = \|\boldsymbol{w}\|_2\, \eta_\varepsilon(\Lambda).$$

Our width assumption ensures $\sigma \geq \eta_\varepsilon(\Gamma)$. The standard smoothing estimate therefore gives, for every shift $\boldsymbol{c} \in \mathcal{U}_{\boldsymbol{w}}$,

$$(1-\varepsilon)K \leq \rho_\sigma(\boldsymbol{c}+\Gamma) \leq (1+\varepsilon)K, \qquad K := \frac{\sigma^m}{\det\Gamma}.$$

Here, crucially, K depends on neither the matrix $\boldsymbol{M}$ nor the orthogonal projection $\boldsymbol{Z}_0$. Therefore,

$$W_{\boldsymbol{M}}(\boldsymbol{Z}_0) \in (1\pm\varepsilon)K \exp\left(-\frac{\pi\|\boldsymbol{Z}_0\|^2}{\sigma^2}\right).$$

The same bounds hold for $W_{\boldsymbol{M}'}(\boldsymbol{Z}_0)$, even if the corresponding hidden shifts are far apart.

Normalize the weights and recover the actual leakage. Define

$$Q(\boldsymbol{Z}_0) = \frac{g(\boldsymbol{Z}_0)}{\sum_{\boldsymbol{Z}\in\mathcal{Z}} g(\boldsymbol{Z})}, \qquad p_{\boldsymbol{M}}(\boldsymbol{Z}_0) = \frac{W_{\boldsymbol{M}}(\boldsymbol{Z}_0)}{\sum_{\boldsymbol{Z}\in\mathcal{Z}} W_{\boldsymbol{M}}(\boldsymbol{Z})}.$$

where $g(\boldsymbol{Z}) = e^{-\pi||\boldsymbol{Z}||^2/\sigma^2}$. Here, $p_{\boldsymbol{M}}(\boldsymbol{Z}_0)$ is the probability that a state $\boldsymbol{S}$ sampled from the gaussian distribution has projection $\boldsymbol{Z}_0$. Bounding numerator and denominator separately gives

$$\frac{1-\varepsilon}{1+\varepsilon} \leq \frac{p_{\boldsymbol{M}}(\boldsymbol{Z}_0)}{Q(\boldsymbol{Z}_0)} \leq \frac{1+\varepsilon}{1-\varepsilon}.$$

It follows that

$$\Delta(p_{\boldsymbol{M}}, Q) = \frac{1}{2}\sum_{\boldsymbol{Z}_0} |p_{\boldsymbol{M}}(\boldsymbol{Z}_0) - Q(\boldsymbol{Z}_0)| \leq \frac{\varepsilon}{1-\varepsilon}.$$

The same calculation for $\boldsymbol{M}'$ and the triangle inequality give (48) for the statistical distance between the projections. Comparing both ratios through Q gives $\kappa^{-1} \leq p_{\boldsymbol{M}}(\boldsymbol{Z}_0)/p_{\boldsymbol{M}'}(\boldsymbol{Z}_0) \leq \kappa$. Finally $\boldsymbol{L}_0\boldsymbol{S} = \boldsymbol{L}_0\boldsymbol{Z}$ is a deterministic function – Statistical distance cannot increase under it. Hence, this proves (48). Similarly, summing the pointwise inequalities over each preimage preserves the ratio bounds for $\boldsymbol{L}_0\boldsymbol{S}$ as well.

Restoring the norm bounds. Let us now see what happens if we condition the distributions $\boldsymbol{S}_{\boldsymbol{M}}$ and $\boldsymbol{S}_{\boldsymbol{M}'}$ to have small norms (as is done in the real scheme). For fixed $C, \boldsymbol{M}$, each coefficient opening $\boldsymbol{s}_{ab}$ is sampled from a shifted-lattice Gaussian $D_{\Lambda+\boldsymbol{t}_{ab},\sigma}$. Since $\sigma \geq \eta_\varepsilon(\Lambda)$, applying the Gaussian tail bound (7) with $s=\sigma$, and $\epsilon=\varepsilon$ gives, for any $\beta_s \geq \sigma\sqrt{m}$,

$$\Pr[\|\boldsymbol{s}_{ab}\|_2 > \beta_s] \leq \Pr[\|\boldsymbol{s}_{ab}\|_2 \geq \sigma\sqrt{m}] \leq \frac{1+\varepsilon}{1-\varepsilon}\, 2^{-m}. \tag{51}$$

$\underline{\mathsf{SimSetup}}(1^\lambda, k, r, c, \ell, f)$: Run the code, and lattice steps of Setup, retaining the trapdoor $\boldsymbol{R}$. Sample $(\mathsf{crs}_{\mathsf{wf}}, \tau_{\mathsf{zk}}) \leftarrow\!\!\$\ \Pi_{\mathsf{wf}}.\mathsf{SimSetup}(1^\lambda)$. Return the public CRS and $\tau_{\mathsf{sim}} = (\boldsymbol{R}, \tau_{\mathsf{zk}})$.

$\underline{\mathsf{SimCommit}(\mathsf{crs}, \tau_{\mathsf{sim}})}$: Sample $\boldsymbol{c}_b \leftarrow\!\!\$\ \mathbb{F}_q^n$ independently for all $b < c$. For the same public statement x_{wf}, sample

$$\pi_{\mathsf{wf}} \leftarrow\!\!\$\ \Pi_{\mathsf{wf}}.\mathsf{SimProve}(\mathsf{crs}_{\mathsf{wf}}, \tau_{\mathsf{zk}}, \mathsf{x}_{\mathsf{wf}}).$$

Return $\mathsf{com} = ((\boldsymbol{c}_b)_{b<c}, \pi_{\mathsf{wf}})$ and $\omega = (\boldsymbol{c}_b)_{b<c}$.

$\underline{\mathsf{SampleState}(\boldsymbol{M}')}$: For every $a < r, b < c$ independently sample

$$\boldsymbol{s}'_{ab} \leftarrow\!\!\$\ \mathsf{SamplePre}(\boldsymbol{A}, \boldsymbol{R}, \boldsymbol{K}_a \boldsymbol{c}_b - M'_{ab}\boldsymbol{e}_1, \sigma).$$

Abort if the norm of any vector exceeds β_s. Return $\mathsf{st}' = (\boldsymbol{s}'_{ab})_{a,b}$.

$\underline{\mathsf{SimFull}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{M})}$: Return $\mathsf{SampleState}(\boldsymbol{M})$.

$\underline{\mathsf{SimBatch}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{z}, J, T, E)}$:

1. Sample $\boldsymbol{M}'$ uniformly from the affine space
$$\left\{\boldsymbol{X} \in \mathbb{F}_q^{r\times c} :\ \mathcal{L}_{J,T}(\boldsymbol{X}) = E, \quad \mathsf{h}_t(\boldsymbol{X}) = z_t\ (t \notin T)\right\}.$$
2. Set $\mathsf{st}' \leftarrow\!\!\$\ \mathsf{SampleState}(\boldsymbol{M}')$; abort on $\perp$. Return $\mathsf{BatchEval}(\mathsf{crs}, \boldsymbol{M}', \mathsf{st}', J, T)$.

Fig. 6. The simulator for proving one-shot adaptive simulation hiding of our commitment scheme $\mathsf{Basis2DLCCom}$.

This bound is uniform in the shift $\boldsymbol{t}_{ab}$, and hence in $C, \boldsymbol{M}$.

Define

$$\beta_s = 2\sigma\sqrt{m + \lambda + \ln(2P)}, \qquad \tau = P\frac{1+\varepsilon}{1-\varepsilon}\,2^{-m}. \tag{52}$$

In particular, $\beta_s \geq \sigma\sqrt{m}$. For $\varepsilon \leq 1/4$ and $m \geq \lambda + \lceil \log_2 P \rceil$, we have $\tau \leq (5/3)2^{-\lambda}$. A union bound over the P coefficient openings gives

$$\Pr[\boldsymbol{S}_{\boldsymbol{M}} \notin \mathcal{B}] \leq \tau$$

for every $C, \boldsymbol{M}$, and the same bound holds for $\boldsymbol{S}_{\boldsymbol{M}'}$. Conditioning each experiment on $\mathcal{B}$ changes its distribution, and hence its proof projection, by at most τ. Hence, the statistical distance can only increase by at most 2τ. This proves the lemma. □

Proof (of Theorem 5). We describe the simulator in Figure 6. We will prove the theorem via a sequence of hybrids.

Recall that the adversary sends J, T in its batch query. Recall $\mathcal{Q}_{J,T}$ is the ordered query list corresponding to J, T (including repetitions), as defined in (5.1). Let $d_Q = 2k|J| + (k - |J|)|T|$. Define $\boldsymbol{L}_{J,T} \in \mathbb{Z}^{d_Q \times P}$ as the set of vectors $(\boldsymbol{g}_i \otimes \boldsymbol{h}_j)^\top$ for each (i, j) in $\mathcal{Q}_{J,T}$. Let $\boldsymbol{S}$ denote the matrix whose row indexed by (a, b) is the opening vector $\boldsymbol{s}_{ab}^\top$. Then,

$$\mathsf{BatchEval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, J, T) \quad \text{is the ordered list of rows of } \boldsymbol{L}_{J,T}\boldsymbol{S}. \tag{53}$$

Moreover, for every $t \notin T$, there exists $\boldsymbol{U}_{J,t}$ as in (45) such that

$$\boldsymbol{L}_{J,T}\,\mathrm{vec}(\boldsymbol{U}_{J,t}) = \boldsymbol{0} \quad \text{over } \mathbb{Z}. \tag{54}$$

Let H_0 denote the real game with $\chi = 1$.

Hybrid H_1: Replace the well-formedness proof. We switch Π_{wf} to its simulation setup by running $(\mathsf{crs}_{\mathsf{wf}}, \tau_{\mathsf{zk}}) \leftarrow\!\$\ \Pi_{\mathsf{wf}}.\mathsf{SimSetup}(1^\lambda)$. Later on, when committing, we use τ_{zk} to simulate a valid proof for the statement x_{wf}. By zero-knowledge, no adversary can distinguish H_0 and H_1 with advantage more than $\varepsilon_{\mathsf{zk}}$. More formally,

$$|\Pr[H_0 \text{ outputs } 1] - \Pr[H_1 \text{ outputs } 1]| \le \varepsilon_{\mathsf{zk}}.$$

Note that the well-formedness proof now depends only on the public commitment vectors.

Hybrid H_2: Apply the Gaussian splitting property. We now replace how the coefficient opening state $\boldsymbol{S}$ is sampled:

$$\boldsymbol{c}_b \leftarrow\!\$\ \mathbb{F}_q^n \quad (0 \le b < c) \text{ independently}, \qquad \boldsymbol{S} \leftarrow\!\$\ D_{\Omega_{C,\boldsymbol{M}},\sigma}, \qquad C = (\boldsymbol{c}_b)_{0\le b<c}. \tag{55}$$

We still simulate the well-formedness proof as in H_1. We argue the indistinguishability of H_1 and H_2 by applying the splitting property (9). For one commitment column, write

$$\boldsymbol{B}_r = [\boldsymbol{E}_r \mid \boldsymbol{F}_r], \qquad \boldsymbol{E}_r = \mathrm{diag}(\boldsymbol{A}, \boldsymbol{W}\boldsymbol{A}, \ldots, \boldsymbol{W}^{r-1}\boldsymbol{A}), \qquad \boldsymbol{F}_r = -\mathbf{1}_r \otimes \boldsymbol{G}.$$

Since $\boldsymbol{W}$ is invertible, we have

$$\Lambda_q^\perp(\boldsymbol{E}_r) = \Lambda^{\oplus r}. \tag{56}$$

Since $\sigma \ge \eta_{\varepsilon/(2r)}(\Lambda)$, the splitting property says that we can replace the column preimage sample in the real scheme by the following:

$$\begin{aligned} \widehat{\boldsymbol{c}}_b &\leftarrow\!\$\ D_{\mathbb{Z}^{d_g},\sigma}, \qquad \boldsymbol{c}_b = \boldsymbol{G}\widehat{\boldsymbol{c}}_b, \\ \boldsymbol{s}_{ab} &\leftarrow\!\$\ D_{\{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{K}_a\boldsymbol{c}_b - M_{ab}\boldsymbol{e}_1\},\sigma} \quad (0 \le a < r), \quad \text{independently}. \end{aligned} \tag{57}$$

The statistical error is at most $\varepsilon/(1-\varepsilon)$ per column. Further, $\sigma \ge \eta_\varepsilon(\Lambda_q^\perp(\boldsymbol{G}))$ makes $\boldsymbol{G}\widehat{\boldsymbol{c}}_b$ within $\varepsilon/(1-\varepsilon)$ of uniform. Replacing $\boldsymbol{c}_b$ by a uniform vector changes the joint distribution of the coefficient openings also by the same amount. Additionally, by the gaussian tail bound (51), the probability that any one of the above vectors end up having a large norm is bounded by τ. Hence, the overall statistical distance is bounded by $2c\varepsilon/(1-\varepsilon) + \tau$.

Hybrid H_3: Use $\mathsf{SimCommit}$. We now generate the commitment and its simulation state using $\mathsf{SimCommit}$ as defined in Figure 6, and sample the coefficient opening state only after receiving a query from the adversary. The challenger still generates the actual sharing matrix $\boldsymbol{M}$ and pads $\boldsymbol{z} = \boldsymbol{s} + \mathsf{h}(\boldsymbol{M})$ as in H_2.

For a full query, we sample $\boldsymbol{S} \leftarrow\!\$\ D_{\Omega_{C,\boldsymbol{M}},\sigma}$ using the matrix $\boldsymbol{M}$ provided as input and return $\boldsymbol{S}$. For a batch query, we first compute the scalar terms $E = \mathcal{L}_{J,T}(\boldsymbol{M})$, but we still use all of $\boldsymbol{M}$ to sample the coefficient openings, from $D_{\Omega_{C,\boldsymbol{M}},\sigma}$. We then return $\boldsymbol{L}_{J,T}\boldsymbol{S}$. If there is no query, nothing needs to be sampled.

It is easy to see that the view of the adversary in H_3 is identical to that in H_2: all the distributions remain the same.

Hybrid H_4: Resample the matrix used for a batch proof. We change how we respond to the batch query. On input the batch query and the scalar values E, we sample a fresh matrix $\boldsymbol{X}$ uniformly from

$$\mathcal{F}_{\boldsymbol{s}}(\theta) := \{\boldsymbol{X} \in \mathbb{F}_q^{r\times c} : \mathcal{L}_{J,T}(\boldsymbol{X}) = E,\ \mathsf{h}(\boldsymbol{X}) = \boldsymbol{z} - \boldsymbol{s}\}, \tag{58}$$

Here θ denotes the transcript fixed before the proof vectors are exposed, together with the adversary's random coins:

$$\theta = (\mathsf{crs}, \rho_{\mathcal{A}}, \boldsymbol{s}, \mathsf{com}, \boldsymbol{z}, Q = \mathsf{Batch}(J,T), E).$$

We then sample $\boldsymbol{S} \leftarrow\!\!\$\ D_{\Omega_{C,\boldsymbol{X}},\sigma}$ and return $\boldsymbol{L}_{J,T}\boldsymbol{S}$. The Full query is answered as in H_3.

Observe that up until the batch/full query, the adversary only saw crs, which is independent of the matrix, the commitment, which is also sampled independently of the matrix. It then sees $\boldsymbol{z}$, which leaks $\mathsf{h}(\boldsymbol{M})$. Lastly, as input to the batch query, we fixed E as the leakage w.r.t. J,T, i.e. $E = \mathcal{L}_{J,T}(\boldsymbol{M})$. Hence, from the view of the adversary, the matrix $\boldsymbol{M}$ is indeed uniform random conditioned on $\mathcal{L}_{J,T}(\boldsymbol{X}) = E$ and $\mathsf{h}(\boldsymbol{X}) = \boldsymbol{z} - \boldsymbol{s}\}$. Moreover, after this response, the adversary receives no further information depending on the original matrix. Hence, the view of the adversary in this hybrid is actually identically distributed to H_3. Note that the challenger uses the knowledge of all the secret values $\boldsymbol{s}$ in this hybrid.

Hybrids $H_4^{(u)}$ for $u = 1, \ldots, \ell$: Change one secret at a time. We define $H_4^{(0)} = H_4$. On a batch query with sets J,T, we define $\boldsymbol{s}^{(u)}$ by

$$\boldsymbol{s}_t^{(u)} = \begin{cases} 0, & t \notin T \text{ and } t \le u, \\ \boldsymbol{s}_t, & \text{otherwise.} \end{cases}$$

Relative to $H_4^{(u-1)}$, hybrid $H_4^{(u)}$ replaces the matrix sampler for the batch proof by a uniform sample from

$$\mathcal{F}^{(u)}(\theta) := \{\boldsymbol{X} \in \mathbb{F}_q^{r\times c} : \mathcal{L}_{J,T}(\boldsymbol{X}) = E,\ \mathsf{h}(\boldsymbol{X}) = \boldsymbol{z} - \boldsymbol{s}^{(u)}\}. \tag{59}$$

Note that $\mathcal{F}^{(u)}(\theta) = \mathcal{F}^{(u-1)}(\theta)$ if $u \in T$, and otherwise, the only difference is that the constraint on $\mathsf{h}_u(\boldsymbol{X})$ is changed from $\boldsymbol{z}_u - \boldsymbol{s}_u$ to $\boldsymbol{z}_u$.

We then sample $\boldsymbol{S} \leftarrow\!\!\$\ D_{\Omega_{C,\boldsymbol{X}},\sigma}$ and return $\boldsymbol{L}_{J,T}\boldsymbol{S}$. The actual secret input to the game is still $\boldsymbol{s}$; the vectors $\boldsymbol{s}^{(u)}$ only specify constraints used by the proof-response sampler. The public pads, query, and scalar response have the same distribution in all these hybrids. Full queries and the no-query branch are unchanged.

Fix any $u \notin T$ and a transcript θ before the proof response. By Lemma 2, the sets $\mathcal{F}^{(u-1)}(\theta)$ and $\mathcal{F}^{(u)}(\theta)$ are nonempty: their mask constraints agree on T, and any assignment to the remaining masks is possible. We now give the correspondence between these two matrix distributions and bound the resulting change in the proof response.

Let $\boldsymbol{U} = \boldsymbol{u}_J \boldsymbol{v}_{J,u}^\top$ (where $\boldsymbol{u}_J,\ \boldsymbol{v}_{J,u}$ are as defined in Lemma 1). Let g be the gcd of its integer entries, and define $\boldsymbol{U}_0 = \boldsymbol{U}/g$ and $\boldsymbol{w} = \mathrm{vec}(\boldsymbol{U}_0)$. Since $\mathsf{h}_u(\boldsymbol{U}) \neq 0 \pmod q$, the prime q does not divide g. Therefore

$$\mathsf{h}_u(\boldsymbol{U}_0) = g^{-1}\mathsf{h}_u(\boldsymbol{U}) \neq 0 \pmod q.$$

The vector $\boldsymbol{w}$ is primitive and satisfies $\|\boldsymbol{w}\|_2 \le K_{\mathsf{hid}}$. Dividing the integer zero identities for $\boldsymbol{U}$ by g preserves those identities; in particular,

$$\mathcal{L}_{J,T}(\boldsymbol{U}_0) = 0, \qquad \mathsf{h}_t(\boldsymbol{U}_0) = 0 \quad (t \neq u), \qquad \boldsymbol{L}_{J,T}\boldsymbol{w} = \boldsymbol{0} \quad \text{over } \mathbb{Z}. \tag{60}$$

Define

$$\delta_u := \frac{s_u^{(u-1)} - s_u^{(u)}}{\mathsf{h}_u(\boldsymbol{U}_0)}, \qquad F_u(\boldsymbol{X}) := \boldsymbol{X} + \delta_u \boldsymbol{U}_0 \quad \text{over } \mathbb{F}_q.$$

For each $\boldsymbol{X} \in \mathcal{F}^{(u-1)}(\theta)$, linearity and (60) imply

$$\begin{aligned} \mathcal{L}_{J,T}(F_u(\boldsymbol{X})) &= E, \\ \mathsf{h}(F_u(\boldsymbol{X})) &= \mathsf{h}(\boldsymbol{X}) + \boldsymbol{s}^{(u-1)} - \boldsymbol{s}^{(u)} = \boldsymbol{z} - \boldsymbol{s}^{(u)}. \end{aligned} \tag{61}$$

Thus F_u maps $\mathcal{F}^{(u-1)}(\theta)$ onto $\mathcal{F}^{(u)}(\theta)$, with inverse obtained by subtracting $\delta_u \boldsymbol{U}_0$. It is a bijection and takes a uniform matrix in the first set to a uniform matrix in the second. Note that we only rely on the existence of $\boldsymbol{u}_J$ and $\boldsymbol{v}_{J,u}$ – we do not actually need to compute these vectors.

For each pair $\boldsymbol{X}, F_u(\boldsymbol{X})$, we apply Lemma 3 in its unbounded Gaussian form, with $\boldsymbol{M} = \boldsymbol{X}$, $\boldsymbol{M}' = F_u(\boldsymbol{X})$, the primitive direction $\boldsymbol{w}$, and the proof map $\boldsymbol{S} \mapsto \boldsymbol{L}_{J,T}\boldsymbol{S}$ (equivalently, $\boldsymbol{L}_{J,T} \otimes I_m$ on the vector of rows). We can do so because of (60) and $\|\boldsymbol{w}\|_2 \le K_{\mathsf{hid}}$. Consequently, for the distributions $\mathcal{D}_{\boldsymbol{X}}$ of the batch proof responses,

$$\Delta\big(\mathcal{D}_{\boldsymbol{X}}, \mathcal{D}_{F_u(\boldsymbol{X})}\big) \le \delta_{\mathsf{dir}}, \qquad \delta_{\mathsf{dir}} := \frac{2\varepsilon}{1-\varepsilon}. \tag{62}$$

Summing over all possible values of $\boldsymbol{X}$ (and the corresponding $F_u(\boldsymbol{X})$, and observing that the hybrids are identical for the full query case, we get that the statistical distance between hybrids $H_4^{(u-1)}$ and $H_4^{(u)}$ is bounded by the same quantity δ_{dir}.

Observe that in hybrid $H_4^{(\ell)}$, the challenger's behavior is identical to the simulator defined in Figure 6. Adding up all the hybrids, we get the theorem. □

6 Lattice Bingo

We now describe our lattice-based packed AVSS protocol LatBingo. There are k parties $P_1, \ldots, P_k$, at most f of which are faulty. The Setup algorithm simply runs Basis2DLCCom.Setup with parameters (k, r, c, ℓ, f), and we assume that all parties have access to the resulting common reference string crs. The dealer's inputs $s_0, \ldots, s_\nu$ are elements of $\mathbb{F}_q$ where $\nu + 1 \le \ell$. For a matrix $\boldsymbol{M} \in \mathbb{F}_q^{r \times c}$ and code parameters $\boldsymbol{G}_{\mathsf{code}} \in \mathbb{Z}^{k \times r}$ and $\boldsymbol{H}_{\mathsf{code}} \in \mathbb{Z}^{k \times c}$ in the CRS, we write $\boldsymbol{g}_i^\top$ and $\boldsymbol{h}_i^\top$ for their ith rows. We define $\boldsymbol{\alpha}_i = \boldsymbol{g}_i^\top \boldsymbol{M} \in \mathbb{F}_q^c, \boldsymbol{\beta}_i = \boldsymbol{M}\boldsymbol{h}_i \in \mathbb{F}_q^r$. In particular,

$$\boldsymbol{\alpha}_i \boldsymbol{h}_u = \boldsymbol{g}_i^\top \boldsymbol{\beta}_u = \mathsf{val}_{\boldsymbol{M}}(i, u).$$

The full encoded array is $\boldsymbol{G}_{\mathsf{code}} \boldsymbol{M} \boldsymbol{H}_{\mathsf{code}}^\top$. Each $\boldsymbol{\alpha}_i$ is a decoded row of this array, while each $\boldsymbol{\beta}_i$ is a decoded column.

Recall that $\boldsymbol{g}_0 = \boldsymbol{b}_0$ and $\boldsymbol{h}_{1-t} = \boldsymbol{a}_t$ for $t \in [\ell]$. We use $\boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_t$ as the mask for the $(t-1)$-th secret:

$$\boldsymbol{s}^{\mathsf{ext}} = (s_0, \ldots, s_\nu, 0, \ldots, 0) \in \mathbb{F}_q^\ell, \quad \mathsf{h}(\boldsymbol{M}) = \big(\boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_t\big)_{t \in [\ell]}, \quad \boldsymbol{z} = \boldsymbol{s}^{\mathsf{ext}} + \mathsf{h}(\boldsymbol{M}).$$

To reconstruct s_j, we will reconstruct $\mathsf{h}_{j+1}(\boldsymbol{M})$ and subtract it from z_{j+1}. For distinct positions $U \subseteq [k]$ and scalar values $(y_u)_{u \in U}$, write

$$\begin{aligned} \mathsf{Decode}^G\big(\{(u, y_u) : u \in U\}\big) &= \boldsymbol{D}_G(U)(y_u)_{u \in U}, \\ \mathsf{Decode}^H\big(\{(u, y_u) : u \in U\}\big) &= \boldsymbol{D}_H(U)(y_u)_{u \in U}, \end{aligned}$$

where $\boldsymbol{D}_G, \boldsymbol{D}_H$ describe the decoder matrix for codes $\boldsymbol{G}_{\mathsf{code}}$ and $\boldsymbol{H}_{\mathsf{code}}$ respectively, for a set of codeword positions $U \subseteq [k]$.

Algorithms 2 and 3 formally describe sharing, and Algorithm 4 describes reconstruction. The sharing protocol uses LatBingoDeal, specified in Algorithm 1, as a building block. The protocol uses reliable broadcast and authenticated, forward-secure private channels. We can rely on existing post-quantum secure reliable broadcast protocols such as [DXR21]. Here r, c, ν, ℓ and f are parameters of the scheme. We discuss how to set these in Section 6.2.2. We prove the security of LatBingo in Section 6.1 and its efficiency properties in Section 6.2. We additionally show how our protocol also supports opening a batch of secrets and summing different AVSS instances in Section 6.2.4.

Algorithm 1 LatBingoDeal$(s_0, \ldots, s_\nu)$

Require $(s_0, \ldots, s_\nu) \in \mathbb{F}_q^{\nu+1}$.
$\boldsymbol{s}^{\mathsf{ext}} \leftarrow (s_0, \ldots, s_\nu, 0, \ldots, 0) \in \mathbb{F}_q^{\ell}$.
Sample $\boldsymbol{M} \leftarrow\!\$\ \mathbb{F}_q^{r \times c}$.
$(\mathsf{com}, \mathsf{st}) \leftarrow\!\$\ \mathsf{Basis2DLCCom.Commit}(\mathsf{crs}, \boldsymbol{M})$.
$\boldsymbol{z} \leftarrow \boldsymbol{s}^{\mathsf{ext}} + \mathsf{h}(\boldsymbol{M})$.
for all $i \in [k]$ **do**
    $\boldsymbol{\alpha}_i \leftarrow \boldsymbol{g}_i^\top \boldsymbol{M}$.
    for all $u \in [k]$ **do**
        $(y^{\mathsf{D}}_{i,u}, \pi^{\mathsf{D}}_{i,u}) \leftarrow \mathsf{Basis2DLCCom.Eval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, (i, u))$.
    end for
end for
Reliably broadcast $\langle$"commits", $(\mathsf{com}, \boldsymbol{z})\rangle$.
for all $i \in [k]$ **do**
    Send privately to P_i $\langle$"data", $\boldsymbol{\alpha}_i, \{(y^{\mathsf{D}}_{i,u}, \pi^{\mathsf{D}}_{i,u})\}_{u\in[k]}\rangle$.
end for

Algorithm 2 LatBingoShare$_i()$, part 1 of 2

$\mathsf{com}_i, \boldsymbol{z}_i, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i \leftarrow \perp$; $\mathcal{P}_i^X, \mathcal{P}_i^Y, \mathcal{D}_i \leftarrow \emptyset$.
$\mathsf{sentRow}_i, \mathsf{sentColumn}_i, \mathsf{sentDone}_i \leftarrow \mathsf{false}$.
if P_i is the dealer with input $s_0, \ldots, s_\nu$ **then**
    Run LatBingoDeal$(s_0, \ldots, s_\nu)$.
end if
upon delivery of the dealer's first reliable broadcast $\langle$"commits", $(\mathsf{com}', \boldsymbol{z}')\rangle$, **do**
    if $\boldsymbol{z}' \in \mathbb{F}_q^{\ell}$ and com' has the prescribed syntax **then**
        $(\mathsf{com}_i, \boldsymbol{z}_i) \leftarrow (\mathsf{com}', \boldsymbol{z}')$.
    end if
upon receiving the first $\langle$"data", $\boldsymbol{\alpha}'_i, \{(y^{\mathsf{D}}_{i,u}, \pi^{\mathsf{D}}_{i,u})\}_{u\in[k]}\rangle$ from the dealer, **do**
    upon $\mathsf{com}_i \neq \perp$, **do**
        if $\boldsymbol{\alpha}_i = \perp$, $\boldsymbol{\alpha}'_i \in \mathbb{F}_q^c$, and $\forall\ u \in [k]$, $\boldsymbol{\alpha}'_i \boldsymbol{h}_u = y^{\mathsf{D}}_{i,u} \wedge \mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}_i, (i, u), y^{\mathsf{D}}_{i,u}, \pi^{\mathsf{D}}_{i,u}) = 1$ **then**
            $\boldsymbol{\alpha}_i \leftarrow \boldsymbol{\alpha}'_i$.
            $\mathcal{P}_i^X \leftarrow \{(u, y^{\mathsf{D}}_{i,u}, \pi^{\mathsf{D}}_{i,u})\}_{u\in[k]}$.
        end if
upon $\boldsymbol{\alpha}_i \neq \perp$ and $\mathsf{sentRow}_i = \mathsf{false}$, **do**
    Choose $\mathcal{Q}_i^X \subseteq \mathcal{P}_i^X$ with $|\mathcal{Q}_i^X| = t_c^H$.
    $\{(u, z^{\alpha}_{i,u}, \pi^{\alpha}_{i,u})\}_{u\in[k]} \leftarrow \mathsf{Basis2DLCCom.GetProofs}(\mathsf{crs}, \mathsf{com}_i, \mathsf{row}, i, \mathcal{Q}_i^X, [k])$.
    for all $u \in [k]$ **do**
        Send $\langle$"row", $z^{\alpha}_{i,u}, \pi^{\alpha}_{i,u}\rangle$ to P_u. ▷ $z^{\alpha}_{i,u} = \boldsymbol{\alpha}_i \boldsymbol{h}_u$.
    end for
    $\mathsf{sentRow}_i \leftarrow \mathsf{true}$.

Algorithm 3 $\mathsf{LatBingoShare}_i()$, continued from Algorithm 2

upon receiving the first $\langle$"row", $z^{\alpha}_{u,i}, \pi^{\alpha}_{u,i}\rangle$ from P_u, **do**
  upon $\mathsf{com}_i \neq \perp$, **do**
    if $|\mathcal{P}_i^Y| < t_c^G$ and $\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}_i, (u,i), z^{\alpha}_{u,i}, \pi^{\alpha}_{u,i}) = 1$ **then**
      $\mathcal{P}_i^Y \leftarrow \mathcal{P}_i^Y \cup \{(u, z^{\alpha}_{u,i}, \pi^{\alpha}_{u,i})\}$.
    end if
    if $|\mathcal{P}_i^Y| = t_c^G$ and $\mathsf{sentColumn}_i = \mathsf{false}$ **then**
      $\boldsymbol{\beta}_i \leftarrow \mathsf{Decode}^G(\{(u,z) : (u,z,\pi) \in \mathcal{P}_i^Y\})$.
      $\{(u, z^{\beta}_{i,u}, \pi^{\beta}_{i,u})\}_{u\in[k]} \leftarrow \mathsf{Basis2DLCCom.GetProofs}(\mathsf{crs}, \mathsf{com}_i, \mathsf{col}, i, \mathcal{P}_i^Y, [k])$.
      for all $u \in [k]$ **do**
        Send $\langle$"column", $z^{\beta}_{i,u}, \pi^{\beta}_{i,u}\rangle$ to P_u. ▷ $z^{\beta}_{i,u} = \boldsymbol{g}_u^\top \boldsymbol{\beta}_i$.
      end for
      $\mathsf{sentColumn}_i \leftarrow \mathsf{true}$.
    end if
upon receiving the first $\langle$"column", $z^{\beta}_{u,i}, \pi^{\beta}_{u,i}\rangle$ from P_u, **do**
  upon $\mathsf{com}_i \neq \perp$, **do**
    if $\boldsymbol{\alpha}_i = \perp \vee |\mathcal{P}_i^X| < t_c^H$, and $\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}_i, (i,u), z^{\beta}_{u,i}, \pi^{\beta}_{u,i}) = 1$ **then**
      $\mathcal{P}_i^X \leftarrow \mathcal{P}_i^X \cup \{(u, z^{\beta}_{u,i}, \pi^{\beta}_{u,i})\}$.
    end if
    if $\boldsymbol{\alpha}_i = \perp$ and $|\mathcal{P}_i^X| = t_c^H$ **then**
      $\boldsymbol{\alpha}_i \leftarrow \left(\mathsf{Decode}^H(\{(u,z) : (u,z,\pi) \in \mathcal{P}_i^X\})\right)^\top$.
    end if
upon $\boldsymbol{\alpha}_i \neq \perp$, $|\mathcal{P}_i^Y| = t_c^G$, $\mathsf{sentRow}_i = \mathsf{sentColumn}_i = \mathsf{true}$, and $\mathsf{sentDone}_i = \mathsf{false}$, **do**
  Send $\langle done \rangle$ to all parties.
  $\mathsf{sentDone}_i \leftarrow \mathsf{true}$.
upon receiving the first $\langle done \rangle$ from P_u, **do**
  $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{u\}$.
upon $|\mathcal{D}_i| \geq k - f$, $\boldsymbol{\alpha}_i \neq \perp$, $|\mathcal{P}_i^Y| = t_c^G$, and $\mathsf{sentDone}_i = \mathsf{true}$, **do**
  Terminate.

Algorithm 4 $\mathsf{LatBingoReconst}_i(j)$ for $j \in \{0, \ldots, \nu\}$

$\mathcal{S}_{i,j} \leftarrow \emptyset$.
Choose $\mathcal{Q}_i^X \subseteq \mathcal{P}_i^X$ with $|\mathcal{Q}_i^X| = t_c^H$.
$\{(-j, z_{i,-j}, \pi_{i,-j})\} \leftarrow \mathsf{Basis2DLCCom.GetProofs}(\mathsf{crs}, \mathsf{com}_i, \mathsf{row}, i, \mathcal{Q}_i^X, \{-j\})$.
Send $\langle$"rec", $j, z_{i,-j}, \pi_{i,-j}\rangle$ to all parties. ▷ $z_{i,-j} = \boldsymbol{\alpha}_i \boldsymbol{a}_{j+1}$.
upon receiving the first $\langle$"rec", $j, z_{u,-j}, \pi_{u,-j}\rangle$ from P_u, **do**
  if $\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}_i, (u,-j), z_{u,-j}, \pi_{u,-j}) = 1$ **then**
    $\mathcal{S}_{i,j} \leftarrow \mathcal{S}_{i,j} \cup \{(u, z_{u,-j})\}$.
  end if
  if $|\mathcal{S}_{i,j}| = t_c^G$ **then**
    $\boldsymbol{\beta}_{-j} \leftarrow \mathsf{Decode}^G(\mathcal{S}_{i,j})$. ▷ $\boldsymbol{\beta}_{-j} = \boldsymbol{M}\boldsymbol{a}_{j+1}$.
    $\widehat{s}_j \leftarrow (\boldsymbol{z}_i)_{j+1} - \boldsymbol{b}_0^\top \boldsymbol{\beta}_{-j} \in \mathbb{F}_q$.
    Output $\widehat{s}_j$ and **terminate**.
  end if

6.1 Security

Theorem 6 (Security of LatBingo). *Consider reliable broadcast and authenticated private channels. Suppose* Basis2DLCCom *satisfies correctness, evaluation binding, proof interpolation in both*

directions, computational proof canonicality and one-shot adaptive simulation-hiding (as defined in Definition 12). Moreover, suppose the parameters satisfy the following conditions.

1. *q is an odd prime, k, r, c, ℓ are positive integers, ν, f are nonnegative integers, and*
$$k > 3f, \quad \nu + 1 \le \ell, \quad f + 1 \le r \le t_c^G \le k - 2f, \quad f + \ell \le c \le t_c^H \le k - f. \tag{63}$$
2. *The two codes have constant rate, i.e. $r = R_G k + O(1)$, $c = R_H k + O(1)$, $t_c^G = \lceil \tau_c^G k \rceil$ and $t_c^H = \lceil \tau_c^H k \rceil$, with $0 < R_G \le \tau_c^G < 1$ and $0 < R_H \le \tau_c^H < 1$.*
3. *The codes and packing vectors satisfy the good-setup conditions of Lemma 1, except with probability $\varepsilon_{\mathsf{code}} + \varepsilon_{\mathsf{mask}} = \mathsf{negl}(\lambda)$.*

Then (LatBingoShare, LatBingoReconst) *is an f-resilient packed AVSS for $\nu+1$ field-element secrets. It satisfies correctness and termination, and one-shot adaptive secrecy.*

Proof. We separately prove correctness, termination and one-shot adaptive secrecy in Theorems 7, 8 and 9 respectively. □

Theorem 7 (Correctness of LatBingo). *Suppose that the dimension and decoding constraints in Theorem 6 hold and that* Basis2DLCCom *satisfies correctness, evaluation binding, and proof interpolation in both directions. Then* LatBingo *satisfies the correctness property of PAVSS.*

Proof. Assume that some nonfaulty party completes LatBingoShare. We construct the extractor required by PAVSS correctness directly from the states of the nonfaulty parties; in particular, we do not invoke the extractor of Basis2DLCCom.

The case of a malformed commitment. Let the reliably broadcast pair be $(\mathsf{com}, \boldsymbol{z})$, with $\mathsf{com} = ((\mathsf{com}_b)_{b=0}^{c-1}, \pi_{\mathsf{wf}})$. If π_{wf} does not verify, the well-formedness check in Basis2DLCCom.Verify rejects every opening under com. Consequently, no nonfaulty party accepts the dealer's row message or any column message, and hence no nonfaulty party sets $\boldsymbol{\alpha}_i \neq \bot$. Since a nonfaulty party completes LatBingoShare only after setting $\boldsymbol{\alpha}_i \neq \bot$, either malformed case contradicts the premise. We may therefore assume that π_{wf} verifies fpr the rest of the analysis.

Construction of the AVSS extractor. Let t be the first time at which a nonfaulty party completes LatBingoShare, and let P_i be such a party. Before completing, P_i received *done* messages from at least $k - f$ distinct parties. Let D_i be any set of exactly $k - f$ such senders. Since at most f parties are faulty,
$$|D_i \cap \mathcal{NF}| \ge k - 2f \ge t_c^G.$$
Using a fixed ordering of the parties, let $\mathcal{H} \subseteq D_i \cap \mathcal{NF}$ be the first t_c^G such parties.

Before sending *done*, every $P_u \in \mathcal{H}$ had set $\boldsymbol{\alpha}_u \neq \bot$. Moreover, it held a set $\mathcal{P}_u^X$ containing at least t_c^H valid openings at distinct second coordinates and fixed first coordinate u. The vector $\boldsymbol{\alpha}_u$ is the decoded row associated with these openings. There are two ways in which P_u may have set it:

1. If P_u accepted $\boldsymbol{\alpha}_u$ directly from the dealer, it checked that $\boldsymbol{\alpha}_u \in \mathbb{F}_q^c$, that every supplied value equals $\boldsymbol{\alpha}_u \boldsymbol{h}_v$, and that every corresponding opening at (u, v) verifies.
2. Otherwise, P_u set $\boldsymbol{\alpha}_u$ by applying Decode^H to exactly t_c^H accepted column messages.

Consequently, for any subset $\mathcal{Q}_u^X \subseteq \mathcal{P}_u^X$ of size t_c^H that the protocol uses, we have
$$\boldsymbol{\alpha}_u^\top = \boldsymbol{D}_H(U_u)(y_{u,v})_{v \in U_u}, \qquad \mathcal{Q}_u^X = \{(v, y_{u,v}, \pi_{u,v}) : v \in U_u\}. \tag{64}$$

We now stack these extracted row vectors $\boldsymbol{\alpha}_u$ in the order of parties in $\mathcal{H}$:

$$\boldsymbol{R}_{\mathcal{H}} = (\boldsymbol{\alpha}_u)_{u\in\mathcal{H}} \in \mathbb{F}_q^{t_c^G \times c}.$$

Define the extracted matrix by

$$\boldsymbol{M}^\star := \boldsymbol{D}_G(\mathcal{H})\boldsymbol{R}_{\mathcal{H}} \in \mathbb{F}_q^{r\times c}. \tag{65}$$

Equivalently, if $d_{a,u}$ is entry (a,u) of $\boldsymbol{D}_G(\mathcal{H})$, then

$$M^\star_{a,b} = \sum_{u\in\mathcal{H}} d_{a,u}(\boldsymbol{\alpha}_u)_b.$$

Essentially, we have run linear decoding of the $\boldsymbol{G}_{\mathsf{code}}$ code on $\boldsymbol{R}_{\mathcal{H}}$ over $\mathbb{F}_q$. We will later show that $\boldsymbol{\alpha}_u$ is indeed the decoded uth row of $\boldsymbol{G}_{\mathsf{code}}\boldsymbol{M}^\star\boldsymbol{H}_{\mathsf{code}}^\top$.

For each $j \in \{0,\dots,\nu\}$, define

$$r_j := z_{j+1} - \boldsymbol{b}_0^\top \boldsymbol{M}^\star \boldsymbol{a}_{j+1}. \tag{66}$$

The algorithm $\mathsf{Ext}_{\mathsf{AVSS}}(\mathsf{view}_t)$ outputs $(r_0,\dots,r_\nu)$.

Existence of valid openings to the extracted matrix. We first derive accepting openings to $\boldsymbol{g}_a^\top\boldsymbol{M}^\star\boldsymbol{h}_b$ at every $(a,b)\in[k]^2$. For every $u\in\mathcal{H}$, use its source set $\mathcal{Q}_u^X$ of size t_c^H to compute

$$\{(b, v_{u,b}, \rho_{u,b})\}_{b\in[k]} \leftarrow \mathsf{Basis2DLCCom.GetProofs}(\mathsf{crs},\mathsf{com},\mathsf{row},u,\mathcal{Q}_u^X,[k]).$$

Every source opening is valid and every source point belongs to the ordinary grid $[k]\times[k]$. Therefore row-direction proof interpolation implies

$$\mathsf{Basis2DLCCom.Verify}(\mathsf{crs},\mathsf{com},(u,b),v_{u,b},\rho_{u,b}) = 1 \qquad (b\in[k]). \tag{67}$$

By the decoded-value clause of proof interpolation (Definition 11) and (64),

$$v_{u,b} = \boldsymbol{\alpha}_u\boldsymbol{h}_b. \tag{68}$$

Thus every column b has the source set

$$\mathcal{Q}_b^Y = \{(u, \boldsymbol{\alpha}_u\boldsymbol{h}_b, \rho_{u,b}) : u\in\mathcal{H}\}.$$

It contains t_c^G accepting openings at the points (u,b). We may therefore invoke column-direction proof interpolation:

$$\{(a, v^\star_{a,b}, \rho^\star_{a,b})\}_{a\in\{0\}\cup[k]} \leftarrow \mathsf{Basis2DLCCom.GetProofs}(\mathsf{crs},\mathsf{com},\mathsf{col},b,\mathcal{Q}_b^Y,\{0\}\cup[k]).$$

The returned openings satisfy

$$\mathsf{Basis2DLCCom.Verify}(\mathsf{crs},\mathsf{com},(a,b),v^\star_{a,b},\rho^\star_{a,b}) = 1. \tag{69}$$

Moreover, the decoded-value clause gives

$$\begin{aligned} v^\star_{a,b} &= \boldsymbol{g}_a^\top \boldsymbol{D}_G(\mathcal{H})(\boldsymbol{\alpha}_u\boldsymbol{h}_b)_{u\in\mathcal{H}} \\ &= \boldsymbol{g}_a^\top \boldsymbol{D}_G(\mathcal{H})\boldsymbol{R}_{\mathcal{H}}\boldsymbol{h}_b = \boldsymbol{g}_a^\top\boldsymbol{M}^\star\boldsymbol{h}_b. \end{aligned} \tag{70}$$

This derives a valid opening at every point in the grid $[k]^2$ as well as every point $(0, b)$ with $b \in [k]$.
Extension to the reconstruction columns. For each row $a \in [k]$, take any t_c^H of the derived ordinary openings (a, b) and invoke $\mathsf{Basis2DLCCom.GetProofs}$ in the row direction with targets $1-\ell, \ldots, 0$. The H decoder recovers $(\boldsymbol{g}_a^\top \boldsymbol{M}^\star)^\top$, so the opening returned at target $1-t$ has value

$$\boldsymbol{g}_a^\top \boldsymbol{M}^\star \boldsymbol{a}_t = \mathsf{val}_{\boldsymbol{M}^\star}(a, 1-t).$$

It verifies by row-direction proof interpolation. Hence accepting openings can be derived at every point in

$$\mathcal{D}_{\mathsf{AVSS}} = \big([k] \times (\{1-\ell, \ldots, 0\} \cup [k])\big) \cup \big(\{0\} \times [k]\big).$$

Every accepted value is consistent with $\boldsymbol{M}^\star$. We now prove that, except with negligible probability, every opening accepted by a nonfaulty party under com has the value prescribed by $\boldsymbol{M}^\star$. Suppose a nonfaulty party accepts an opening (z, π) at $(a, b) \in \mathcal{D}_{\mathsf{AVSS}}$, so

$$\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}, (a, b), z, \pi) = 1.$$

The preceding argument supplies another accepting opening $(\mathsf{val}_{\boldsymbol{M}^\star}(a, b), \rho^\star_{a,b})$ at the same commitment and point. If these values differ, the tuple

$$\big(\mathsf{com}, (a, b), (z, \pi), (\mathsf{val}_{\boldsymbol{M}^\star}(a, b), \rho^\star_{a,b})\big)$$

violates evaluation binding. Consequently, except with negligible probability,

$$\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}, (a, b), z, \pi) = 1 \quad \Longrightarrow \quad z = \boldsymbol{g}_a^\top \boldsymbol{M}^\star \boldsymbol{h}_b \tag{71}$$

for every opening occurring in the execution. A union bound over the polynomially many executed and derived openings preserves negligibility.
Consistency of the rows held by nonfaulty parties. Let P_v be any nonfaulty party that sets $\boldsymbol{\alpha}_v \neq \perp$. If it accepts the row directly from the dealer, its dimension check gives $\boldsymbol{\alpha}_v \in \mathbb{F}_q^c$ and its verification checks give at least t_c^H accepted values $\boldsymbol{\alpha}_v \boldsymbol{h}_u$ at distinct points (v, u). Equation (71) implies

$$\boldsymbol{\alpha}_v \boldsymbol{h}_u = \boldsymbol{g}_v^\top \boldsymbol{M}^\star \boldsymbol{h}_u \qquad (u \in U,\ |U| \geq t_c^H).$$

Applying $\boldsymbol{D}_H(U)$ to these equal encoded values yields

$$\boldsymbol{\alpha}_v = \boldsymbol{g}_v^\top \boldsymbol{M}^\star. \tag{72}$$

If P_v instead constructs its row from t_c^H column messages, every source value equals $\boldsymbol{g}_v^\top \boldsymbol{M}^\star \boldsymbol{h}_u$ by the same binding argument. The identity $\boldsymbol{D}_H(U)(\boldsymbol{H}_{\mathsf{code}})_U = \boldsymbol{I}_c$ again gives (72). In particular, applying this conclusion to $v \in \mathcal{H}$ proves:

$$\boldsymbol{g}_u^\top \boldsymbol{M}^\star = \boldsymbol{\alpha}_u \qquad (u \in \mathcal{H}). \tag{73}$$

The analogous argument applies to a nonfaulty party's column: It obtains t_c^G valid row messages at (u, v) (which must be consistent with $\boldsymbol{M}^\star$ as proven above) and decodes

$$\boldsymbol{\beta}_v = \boldsymbol{D}_G(U)(\boldsymbol{g}_u^\top \boldsymbol{M}^\star \boldsymbol{h}_v)_{u \in U} = \boldsymbol{M}^\star \boldsymbol{h}_v.$$

Hence, both the row and column states of all nonfaulty parties are consistent with $\boldsymbol{M}^\star$.

Validity for a nonfaulty dealer. Suppose the dealer is nonfaulty and write $\boldsymbol{M}^{\mathsf{D}}$ for its honestly sampled matrix. Commitment correctness implies that every opening produced by the dealer's Eval calls verifies and has value $\boldsymbol{g}_a^\top \boldsymbol{M}^{\mathsf{D}} \boldsymbol{h}_b$. Fix $u \in \mathcal{H}$ and a tuple $(b, v, \pi) \in \mathcal{Q}_u^X$. The honest dealer can also compute an accepting proof $\pi_{u,b}^{\mathsf{D}}$ such that

$$\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}, (u, b), \boldsymbol{g}_u^\top \boldsymbol{M}^{\mathsf{D}} \boldsymbol{h}_b, \pi_{u,b}^{\mathsf{D}}) = 1.$$

Evaluation binding implies $v = \boldsymbol{g}_u^\top \boldsymbol{M}^{\mathsf{D}} \boldsymbol{h}_b$. This holds at the t_c^H distinct source columns in $\mathcal{Q}_u^X$, so H decoding gives $\boldsymbol{\alpha}_u = \boldsymbol{g}_u^\top \boldsymbol{M}^{\mathsf{D}}$. Stacking these identities for all $u \in \mathcal{H}$ and applying the G decoder gives

$$\boldsymbol{M}^\star = \boldsymbol{D}_G(\mathcal{H})(\boldsymbol{G}_{\mathsf{code}})_{\mathcal{H}} \boldsymbol{M}^{\mathsf{D}} = \boldsymbol{M}^{\mathsf{D}}. \tag{74}$$

The dealer broadcasts $z_{j+1} = s_j + \boldsymbol{b}_0^\top \boldsymbol{M}^{\mathsf{D}} \boldsymbol{a}_{j+1}$. Combining this identity with (66) and (74) gives $r_j = s_j$ for every $j \in \{0, \ldots, \nu\}$. This proves the nonfaulty-dealer part of PAVSS correctness.

Correctness of reconstruction. Fix $j \in \{0, \ldots, \nu\}$, and suppose that a nonfaulty party P_i completes $\mathsf{LatBingoReconst}(j)$. It collected

$$\mathcal{S}_{i,j} = \{(u, z_{u,-j}) : u \in U\} \qquad \text{with } |U| = t_c^G.$$

Every value in this set was accompanied by a proof satisfying

$$\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}, (u, -j), z_{u,-j}, \pi_{u,-j}) = 1.$$

Since $\boldsymbol{h}_{-j} = \boldsymbol{a}_{j+1}$, consistency of accepted values (71) gives $z_{u,-j} = \boldsymbol{g}_u^\top \boldsymbol{M}^\star \boldsymbol{a}_{j+1}$ for every $u \in U$. The reconstructed vector therefore satisfies

$$\boldsymbol{\beta}_{-j} = \mathsf{Decode}^G(\mathcal{S}_{i,j}) = \boldsymbol{D}_G(U)(\boldsymbol{G}_{\mathsf{code}})_U \boldsymbol{M}^\star \boldsymbol{a}_{j+1} = \boldsymbol{M}^\star \boldsymbol{a}_{j+1}.$$

All nonfaulty parties use the same reliable-broadcast pad vector, so the output of P_i is

$$\widehat{s}_j = (\boldsymbol{z}_i)_{j+1} - \boldsymbol{b}_0^\top \boldsymbol{\beta}_{-j} = z_{j+1} - \boldsymbol{b}_0^\top \boldsymbol{M}^\star \boldsymbol{a}_{j+1} = r_j.$$

Thus every nonfaulty party that completes reconstruction of index j outputs the same extracted field element. Together with the nonfaulty-dealer argument, this proves PAVSS correctness. □

Theorem 8 (Termination of LatBingo). *Under reliable delivery and the parameter constraints $k - 2f \geq t_c^G$ and $k - f \geq t_c^H$, if $\mathsf{Basis2DLCCom}$ satisfies correctness and proof interpolation in both directions, then $\mathsf{LatBingo}$ satisfies termination, except with negligible probability.*

Proof. We first observe that if π_{wf} does not verify, the well-formedness check in Verify rejects every dealer, row, column, and reconstruction opening. Hence no nonfaulty party sets $\boldsymbol{\alpha}_i \neq \perp$ or sends *done*. This case cannot occur for a nonfaulty dealer except with the commitment's completeness error, and when the dealer is faulty, no non-faulty party will complete $\mathsf{LatBingoShare}$ because of the observation above. Hence for the remaining analysis, we assume that the reliably broadcast commitment has a verifying well-formedness proof.

Part (1): nonfaulty dealer. Every nonfaulty party eventually receives the same reliable-broadcast pair and its private decoded-row message, by reliable broadcast and valid scheduling. By commitment correctness, every supplied proof will verify, with value $\boldsymbol{\alpha}_i \boldsymbol{h}_u$. The vector has dimension c,

so every nonfaulty party accepts its row, fills $\mathcal{P}_i^X$, and sets $\boldsymbol{\alpha}_i \neq \perp$. Its row-direction invocation of GetProofs has at least t_c^H valid sources and therefore produces valid row messages for all parties.

There are at least $k-f$ nonfaulty parties, and $k-f \geq k-2f \geq t_c^G$. Consequently every nonfaulty P_i eventually receives t_c^G valid row messages at (u,i), fills $\mathcal{P}_i^Y$, and decodes $\boldsymbol{\beta}_i$ with the G decoder. Column-direction proof interpolation then makes every column message that it sends accepting. Both sending flags are now true, so every nonfaulty party sends *done*. By valid scheduling, every nonfaulty party eventually receives *done* from the $k-f$ nonfaulty parties, and terminates LatBingoShare.

Part (2): one nonfaulty party completes sharing. Let P_h be a nonfaulty party that completes sharing, and let D_h be a set of $k-f$ parties from which it accepted *done*. At most f members are faulty, so $\mathcal{H} = D_h \cap \mathcal{NF}$ has size $|\mathcal{H}| \geq k-2f \geq t_c^G$. Before sending *done*, each party in $\mathcal{H}$ held at least t_c^H valid row-direction sources in $\mathcal{P}_u^X$ and t_c^G valid column-direction sources in $\mathcal{P}_u^Y$. The sending flags ensure that it had already sent both row and column messages. Proof interpolation makes all those openings accepting.

Reliable broadcast eventually gives every participating nonfaulty party the same pair $(\mathsf{com}, \boldsymbol{z})$. The row messages sent by $\mathcal{H}$ then provide every such party with at least t_c^G valid sources at (u,i). It can decode $\boldsymbol{\beta}_i$ and send its own column messages, even if its $\boldsymbol{\alpha}_i$ is still undefined. This is the first repair step, and it is precisely where $k-2f \geq t_c^G$ is used.

All $k-f$ nonfaulty parties consequently send valid column messages. Their delivery gives any nonfaulty P_i lacking a dealer row at least t_c^H valid values at (i,u), because $k-f \geq t_c^H$. It applies the H decoder, sets $\boldsymbol{\alpha}_i$, and sends its row messages using the retained repair proofs. A party that already accepted its row keeps that row and its source set; it does not need to replace them with the arriving repair messages. This is the second repair step, and it uses $k-f \geq t_c^H$. Every nonfaulty party now has its row, a full column source set, and both sending flags. It sends *done* if it has not already done so. Delivery of the $k-f$ nonfaulty *done* messages makes all nonfaulty parties complete sharing.

Part (3): reconstruction after sharing. After completing sharing, every nonfaulty P_i has at least t_c^H valid openings in $\mathcal{P}_i^X$ at distinct columns of row i. Hence its row-direction invocation of GetProofs with target $-j$ will return an accepting opening at $(i,-j)$. If every nonfaulty party invokes LatBingoReconst(j), valid scheduling delivers at least $k-f \geq t_c^G$ such openings to every nonfaulty party. Each recipient applies the G decoder to its first t_c^G accepted values, evaluates the decoded vector against $\boldsymbol{b}_0^\top$, subtracts the resulting mask from the common pad, and terminates. The same argument applies separately to every requested secret index. Note no reconstruction step requires t_c^H senders: that threshold is used locally to generate a row proof, while t_c^G is used to recover the reconstruction column. □

Theorem 9 (One-shot adaptive secrecy of LatBingo). *Under the assumptions and parameter constraints of Theorem 6 and assuming* Basis2DLCCom *satisfies one-shot adaptive simulation-hiding and computational proof canonicality,* LatBingo *satisfies one-shot adaptive secrecy.*

Proof. Fix a PPT adversary $\mathcal{A}$. We construct a simulator Sim for the one-shot adaptive secrecy game. The simulator uses SimSetup and SimCommit, and subsequently uses exactly one of SimBatch and SimFull based on the adversary's query. Recall that the adversary may choose the corruption and reconstruction query adaptively, based on the public transcript so far – but it only allowed a single batch query, although it can still control delivery scheduling and its declared corrupted parties.

We use $J \subseteq [k]$ for the complete corruption set and $\mathcal{T} \subseteq \{0, \ldots, \nu\}$ for the complete set of reconstruction indices. The associated index set for the commitment is

$$T = \{j + 1 : j \in \mathcal{T}\} \subseteq [\ell].$$

Simulator state. The simulator maintains the public message buffer, each party's control state, and an ordered event log $\mathcal{L}$. Before the batch declaration, private messages whose contents depend on the sharing matrix are represented symbolically. A symbolic message records its sender, recipient, type, and delivery position, but leaves its decoded row, field values, and opening vectors unspecified.

As in the execution model, private messages are transmitted over forward-secure channels. Hence, before the complete corruption set is declared, the simulator may place opaque ciphertexts in the buffer and later determine the corresponding plaintexts when constructing a corrupted recipient's view. Public messages and messages supplied by $\mathcal{A}$ are never treated symbolically.

Setup. The simulator computes

$$(\mathsf{crs}, \tau_{\mathsf{sim}}) \leftarrow\!\$\ \mathsf{Basis2DLCCom.SimSetup}(1^\lambda, k, r, c, \ell, f)$$

and gives crs to $\mathcal{A}$. It initializes every party's variables exactly as in LatBingoShare. The code parameters, masks, index domains, and decoder ordering are those contained in this same CRS.

Starting the dealer. When $\mathcal{A}$ delivers a start message to a nonfaulty dealer, the simulator computes

$$(\mathsf{com}, \omega) \leftarrow\!\$\ \mathsf{Basis2DLCCom.SimCommit}(\mathsf{crs}, \tau_{\mathsf{sim}}) \qquad \text{and} \qquad \boldsymbol{z} \leftarrow\!\$\ \mathbb{F}_q^\ell.$$

It adds the reliable-broadcast message $\langle \text{“commits”}, (\mathsf{com}, \boldsymbol{z}) \rangle$ to the buffer. For every $i \in [k]$, it also adds the private symbolic message $\langle \text{“data”}, \square_i \rangle$ addressed to P_i. Here $\square_i$ represents a row $\boldsymbol{\alpha}_i \in \mathbb{F}_q^{1\times c}$ and the k opening pairs at (i, u) for $u \in [k]$.

Informally, the pads can be chosen before learning the dealer's inputs because for a uniform matrix $\boldsymbol{M}$, the vector $\mathsf{h}(\boldsymbol{M})$ is uniform in $\mathbb{F}_q^\ell$. We will formally argue this below.

Simulation before the batch query. Before receiving the batch query, Sim handles messages originating from $\mathcal{A}$ exactly as a real nonfaulty recipient would. In particular, it invokes Basis2DLCCom.Verify and updates the recipient's state only when verification succeeds.

A symbolic message sent between two simulated nonfaulty parties is assumed to be accepting. The simulator records this in $\mathcal{L}$ and performs the same actions as the real party:

- After accepting a decoded row and its row-opening source set, a party buffers its row messages, using GetProofs.
- After collecting t_c^G valid row messages at (u, i), a party decodes $\boldsymbol{\beta}_i$, applies the column-direction GetProofs procedure, and buffers its column messages.
- If its row is undefined and it collects t_c^H valid column messages at (i, u), it decodes $\boldsymbol{\alpha}_i$ with the H decoder and buffers its row messages from that repair set.
- Once the source-set and sending-flag conditions hold, it buffers its *done* messages.
- upon an invocation of $\mathsf{LatBingoReconst}(j)$, it buffers the prescribed reconstruct message.

The contents of messages sent by a simulated nonfaulty party remain symbolic until the batch query fixes the polynomial. Messages delivered to nonfaulty parties can therefore be simulated at the level of protocol control flow. Messages sent to a party that is declared corrupted in the batch query are materialized as described next.

The batch query. Suppose $\mathcal{A}$ makes its query $(J, \mathcal{T})$. The simulator receives s_j for every $j \in \mathcal{T}$. If the dealer belongs to J, it additionally receives all dealer inputs. Recall the scalar leakage map from the commitment definition:

$$E = \mathcal{L}_{J,T}(\boldsymbol{M}) = \big((\boldsymbol{G}_{\mathsf{code}})_J \boldsymbol{M},\ \boldsymbol{M}(\boldsymbol{H}_{\mathsf{code}})_J^\top,\ (\boldsymbol{G}_{\mathsf{code}})_{\bar{J}} \boldsymbol{M} A_T\big), \qquad A_T = [\boldsymbol{a}_t]_{t\in T}. \tag{75}$$

The first block is the collection of corrupted decoded rows, the second block is the collection of corrupted decoded columns, and the third block contains honest-party reconstruction values for the declared secrets. The reconstruction values of parties in J can be computed from the first block. Thus, for $t \in T$, the leakage determines the full codeword $\boldsymbol{G}_{\mathsf{code}}\boldsymbol{M}\boldsymbol{a}_t$, its decoded vector $\boldsymbol{M}\boldsymbol{a}_t$, and the corresponding mask $\boldsymbol{b}_0^\top \boldsymbol{M}\boldsymbol{a}_t$.

The associated joint proof list contains the row strips (i, u) with $i \in J$, all column strips (u, i) with $i \in J$, and reconstruction positions $(u, 1-t)$ with $u \in \bar{J}$ and $t \in T$. We will show how the simulator computes these proofs below.

Sampling the scalar leakage when inputs remain hidden. First suppose the dealer is not in J and at least one of its $\nu + 1$ inputs is not in $\mathcal{T}$. Then $|T| \le \nu < \ell$, so $\mathsf{Batch}(J, T)$ is admissible. Define a completed input vector $\widetilde{\boldsymbol{s}} \in \mathbb{F}_q^\ell$ by

$$\widetilde{s}_t = \begin{cases} s_{t-1}, & t \in T, \\ 0, & t \notin T. \end{cases}$$

This also assigns zero to every unused dummy-input position (i.e. positions from $\nu + 1$ to $\ell - 1$). Sample

$$\widetilde{\boldsymbol{M}} \leftarrow\!\$ \left\{\boldsymbol{X} \in \mathbb{F}_q^{r\times c} : \mathsf{h}(\boldsymbol{X}) = \boldsymbol{z} - \widetilde{\boldsymbol{s}}\right\} \quad \text{uniformly}, \tag{76}$$

and define $\widetilde{E} = \mathcal{L}_{J,T}(\widetilde{\boldsymbol{M}})$. Now make the single simulation call

$$\mathit{\Pi} \leftarrow\!\$ \mathsf{Basis2DLCCom.SimBatch}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{z}, J, T, \widetilde{E}). \tag{77}$$

Only $\widetilde{E}$ and the returned joint proof list are used to materialize the exposed views. The matrix $\widetilde{\boldsymbol{M}}$ in (76) is needed just for sampling the leakage; it is not actually used as the real matrix. In particular, the simulator neither obtains nor reveals a full opening state in this branch. Moreover, the internal matrix used by $\mathsf{SimBatch}$ need not be $\widetilde{\boldsymbol{M}}$.

The full branch when all dealer inputs are available. If the dealer is in J, or if $\mathcal{T} = \{0, \ldots, \nu\}$, the simulator knows the full vector $\boldsymbol{s}^{\mathsf{ext}}$. It samples

$$\widetilde{\boldsymbol{M}} \leftarrow\!\$ \{\boldsymbol{X} : \mathsf{h}(\boldsymbol{X}) = \boldsymbol{z} - \boldsymbol{s}^{\mathsf{ext}}\} \quad \text{uniformly}$$

and makes the single call

$$\mathsf{st}^\star \leftarrow\!\$ \mathsf{Basis2DLCCom.SimFull}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \widetilde{\boldsymbol{M}}). \tag{78}$$

This is an alternative to (77) (since the dealer is either in J or not in J, the simulator will only execute one of these branches). The simulator can now run all dealer Eval calls and all party $\mathsf{GetProofs}$ calls using $\mathsf{st}^\star$. If the dealer is corrupted, its persistent state contains $\widetilde{\boldsymbol{M}}, \mathsf{st}^\star, \boldsymbol{s}^{\mathsf{ext}}, \boldsymbol{z}$ and its deterministic protocol data, together with its ordinary local state. Note we treat the commitment-sampling and argument-prover randomness as ephemeral, and hence are not part of dealer state.

Materializing LatBingoDeal. In the full branch, the simulator has $\widetilde{\boldsymbol{M}}$ and $\mathsf{st}^\star$ from its single SimFull call. For every $i, u \in [k]$, it computes

$$\widetilde{\boldsymbol{\alpha}}_i = \boldsymbol{g}_i^\top \widetilde{\boldsymbol{M}}, \qquad (y^{\mathsf{D}}_{i,u}, \boldsymbol{\pi}^{\mathsf{D}}_{i,u}) \leftarrow \mathsf{Basis2DLCCom.Eval}(\mathsf{crs}, \widetilde{\boldsymbol{M}}, \mathsf{st}^\star, (i,u)).$$

It replaces the placeholder $\square_i$ with

$$\left(\widetilde{\boldsymbol{\alpha}}_i,\ \{(y^{\mathsf{D}}_{i,u}, \boldsymbol{\pi}^{\mathsf{D}}_{i,u})\}_{u\in[k]}\right).$$

It then runs the actual party algorithms on these messages and on the adversary's subsequent messages. If the dealer is corrupted, its persistent state is supplied as described above. No batch query is made in this branch.

In the batch branch, let $\mathit{\Pi}$ be the joint proof response from (77). For a point in that response, write $\widehat{\boldsymbol{\pi}}_{a,b}$ for its proof vector, and $\widehat{y}_{a,b}$ for the scalar value determined by the leakage $\widetilde{E}$. The response covers

$$\mathcal{Q}_{J,T} = (J \times [k]) \cup ([k] \times J) \cup \{(u, 1-t) : u \in \bar{J},\ t \in T\}.$$

For every corrupted party P_i, the first block of $\widetilde{E}$ supplies $\boldsymbol{\alpha}_i$. Its dealer-message placeholder is replaced with

$$\left(\boldsymbol{\alpha}_i,\ \{(\boldsymbol{\alpha}_i \boldsymbol{h}_u, \widehat{\boldsymbol{\pi}}_{i,u})\}_{u\in[k]}\right).$$

Dealer messages to parties outside J remain symbolic.

Why the joint batch proofs suffice. If the dealer is corrupt, observe that the simulator essentially learns the full state of the dealer, and hence can simulate all of the sharing and reconstruction messages using the full state. On the other hand, if the dealer is honest, then, for an honestly generated commitment state $\mathsf{st} = (\boldsymbol{s}_{a,b})_{a<r,b<c}$, define the reference opening vectors

$$\boldsymbol{\pi}^\circ_{i,u} = \sum_{a<r,b<c} g_{i,a} h_{u,b} \boldsymbol{s}_{a,b}.$$

By correctness of Basis2DLCCom, GetProofs applied to reference source proofs returns accepting proofs at its targets, and by computational proof canonicality these coincide with the reference proofs except with negligible probability, independently of which admissible source subset is used. The proof equalities hold over $\mathbb{Z}$, and the corresponding scalar equalities hold over $\mathbb{F}_q$. They also hold for the special targets in $\mathcal{D}$ allowed by the commitment interface.

In the real execution, correctness, evaluation binding, and computational proof canonicality imply that every accepted source value-proof pair equals its reference pair, except with negligible probability. Thus the real proof sent by an honest party at an exposed point is exactly the reference proof included in the joint batch response. This covers proofs derived from accepted adversarial messages as well as proofs derived entirely from honest messages. The simulator can therefore use the cached batch proof at each exposed target when simulating the row/column messages in LatBingoShare.

Note that we do not assume canonicality under the simulated CRS. In the hybrids below, we will first apply canonicality to the real execution before invoking one-shot adaptive simulation hiding. In the batch simulation it uses the following concrete abort rule: if an adversarial opening accepted by the public verifier differs in either value or proof vector from the cached reference pair at that exposed point, it aborts. Every adversarial row or column message has a sender in J, so the corresponding reference pair is available. Reference pairs for reconstruction messages from J are derived from the available row strips as described below. Hence, the simulator can implement all these checks without a full opening state.

Replaying LatBingoShare. The simulator replays the event log $\mathcal{L}$ in the original delivery order, retaining all first-message choices, source sets, and sending flags. For a corrupted party's dealer message, it performs the actual checks

$$\boldsymbol{\alpha}_i \in \mathbb{F}_q^c, \qquad \boldsymbol{\alpha}_i \boldsymbol{h}_u = y_{i,u}^{\mathsf{D}} \quad (u \in [k]),$$
$$\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}, (i,u), y_{i,u}^{\mathsf{D}}, \boldsymbol{\pi}_{i,u}^{\mathsf{D}}) = 1 \quad (u \in [k]).$$

Its materialized local state contains the accepted row and the actual indexed proof set $\mathcal{P}_i^X$. The corresponding state of a party outside J can remain symbolic in the batch branch. The full branch represents all party states concretely.

A row message sent by an honest P_u to a corrupted P_i has point (u,i) and value

$$\widehat{y}_{u,i} = \boldsymbol{g}_u^\top \boldsymbol{\beta}_i,$$

where $\boldsymbol{\beta}_i$ is supplied by the second block of $\widetilde{E}$. Its proof is $\widehat{\boldsymbol{\pi}}_{u,i}$, also included in the SimBatch response. The simulator inserts $\langle$"row", $\widehat{y}_{u,i}, \widehat{\boldsymbol{\pi}}_{u,i}\rangle$ at the time prescribed by the recorded protocol execution.

A column message sent by an honest P_u to a corrupted P_i has point (i,u) and value

$$\widehat{y}_{i,u} = \boldsymbol{\alpha}_i \boldsymbol{h}_u,$$

where $\boldsymbol{\alpha}_i$ is supplied by the first block of $\widetilde{E}$. Its proof is $\widehat{\boldsymbol{\pi}}_{i,u}$ from the row strip indexed by $i \in J$ in the SimBatch response. The simulator inserts $\langle$"column", $\widehat{y}_{i,u}, \widehat{\boldsymbol{\pi}}_{i,u}\rangle$ at the corresponding scheduled time. Thus both kinds of messages to corrupted recipients use points already included in the SimBatch response.

For every party, receiving t_c^G valid row messages at (u,i) fills $\mathcal{P}_i^Y$, triggers G decoding of $\boldsymbol{\beta}_i$, and triggers its column messages. Receiving t_c^H valid column messages at (i,u) repairs an undefined $\boldsymbol{\alpha}_i$ by H decoding and triggers its row messages. The source indices and stored proof strings are the ones selected by the actual delivery order. For a party whose pre-corruption state is exposed, the simulator replays its local GetProofs calls on those concrete sources and checks their agreement with the cached exposed target proofs. Such checks are public computations on the batch response. For parties outside J, it only needs the source indices, message-acceptance flags, and the resulting control flow (since their real messages still remain symbolic).

An adversarial message is first checked with the real Basis2DLCCom.Verify algorithm. Invalid messages are treated exactly as in the protocol. An accepting message is subjected to the cached-pair check above (i.e. it must match the value and opening proof returned by SimBatch); if that check passes, the simulator makes the actual state update. An honest-to-honest message remains symbolic and is provisionally treated as accepting. All *done* messages, sets $\mathcal{D}_i$, sending flags, and termination conditions are processed exactly as in LatBingoShare.

Replaying LatBingoReconst. For any index $j \in \mathcal{T}$, $j+1 \in T$ and $\boldsymbol{h}_{-j} = \boldsymbol{a}_{j+1}$. For each honest sender $P_u \notin J$, the batch response contains the pair

$$(\widehat{y}_{u,-j}, \widehat{\boldsymbol{\pi}}_{u,-j}) \quad \text{at point } (u,-j).$$

Its scalar value is the corresponding entry of the reconstruction block of $\widetilde{E}$. The simulator uses this pair in the sender's reconstruction message at the scheduled time. This includes reconstruction messages sent to corrupted recipients.

For a sender $P_u \in J$, the simulator already has the entire row strip at (u, v), $v \in [k]$. It chooses any fixed t_c^H indices from that strip and computes

$$\{(-j, \widehat{y}_{u,-j}, \widehat{\boldsymbol{\pi}}_{u,-j})\} \leftarrow \mathsf{Basis2DLCCom.GetProofs}\big(\mathsf{crs}, \mathsf{com}, \mathsf{row}, u, \{(v, \widehat{y}_{u,v}, \widehat{\boldsymbol{\pi}}_{u,v}) : v \in V\}, \{-j\}\big).$$

Observe that this is identical to what P_u would have done in the real protocol execution for reconstruction.

Each recipient verifies a received reconstruction message at $(u, -j)$. After accepting t_c^G distinct senders it computes

$$\boldsymbol{\beta}_{-j} = \mathsf{Decode}^G(\mathcal{S}_{i,j}), \qquad \widehat{s}_j = z_{j+1} - \boldsymbol{b}_0^\top \boldsymbol{\beta}_{-j}.$$

Observe that the leakage $\widetilde{E}$ determines all these accepted reference values. Moreover, by the constraints used to sample $\widetilde{E}$, the reconstructed output equals the disclosed secret s_j. The full branch executes the same operations directly using $\widetilde{\boldsymbol{M}}, \mathsf{st}^\star$.

The simulator returns the jointly materialized corrupted-party views and continues under the adversary's schedule. Honest-to-honest plaintexts can remain symbolic in the batch branch, since the complete corruption set has already been fixed. All subsequent reconstruction invocations use indices already declared in $\mathcal{T}$.

We now prove indistinguishability of the simulation with a real execution via a sequence of hybrids:

Game G_0*: the real execution.* Game G_0 is the real execution with the actual dealer inputs, a uniform matrix $\boldsymbol{M}$, an honestly generated commitment and state, and pads $\boldsymbol{z} = \boldsymbol{s}^{\mathsf{ext}} + \mathsf{h}(\boldsymbol{M})$. The adversary chooses its single query after the public transcript it is allowed to observe. For analysis, we look at the honest dealer's $\boldsymbol{M}, \mathsf{st}$ and define the reference pair at every point by

$$(y^\circ_{a,b}, \boldsymbol{\pi}^\circ_{a,b}) \leftarrow \mathsf{Basis2DLCCom.Eval}(\mathsf{crs}, \boldsymbol{M}, \mathsf{st}, (a, b)).$$

This state is available to the reductions, which generate the real commitment themselves. But it is never revealed to the adversary.

Game G_1*: canonical exposed proofs.* Let $\mathsf{Bad}_{\mathsf{val}}$ be the event that a nonfaulty party accepts an opening with $y \neq y^\circ_{a,b}$. Combining that opening with the reference opening (defined above) gives an evaluation-binding violation. Let $\mathsf{Bad}_{\mathsf{can}}$ be the event that a nonfaulty party accepts an opening with $y = y^\circ_{a,b}$ but $\boldsymbol{\pi} \neq \boldsymbol{\pi}^\circ_{a,b}$. Combining these two accepting proofs for the same commitment, point, and value gives a violation of computational proof canonicality (Definition 9).

In more detail, both reductions run the real setup experiment. They can inspect the polynomial execution log, generate the reference openings, and output the first violation of either evaluation binding or canonicality. Together with correctness, these properties imply that all accepted source pairs equal the reference pairs except with negligible probability.

The above also applies to the decoded rows and columns. If a party accepts t_c^H correct symbols at (i, u), the identity $\boldsymbol{D}_H(U)(\boldsymbol{H}_{\mathsf{code}})_U = \boldsymbol{I}_c$ recovers $\boldsymbol{\alpha}_i = \boldsymbol{g}_i^\top \boldsymbol{M}$. Similarly, t_c^G correct symbols at (u, i) recover $\boldsymbol{\beta}_i = \boldsymbol{M}\boldsymbol{h}_i$. By correctness and computational proof canonicality of $\mathsf{Basis2DLCCom}$, applying $\mathsf{GetProofs}$ to the corresponding reference proof vectors returns exactly the reference target proofs. In particular, even if the valid source proofs are chosen adversarially, the proofs output by interpolation must still be the same as the reference pair, except for the event $\mathsf{Bad}_{\mathsf{val}} \vee \mathsf{Bad}_{\mathsf{can}}$.

Also let $\mathsf{Bad}_{\mathsf{int}}$ include any admissible honest proof-interpolation call on accepting sources that returns $\perp$, omits a requested target, or returns a nonverifying opening. The commitment's proof-interpolation guarantee bounds the probability this event by negligible.

In G_1, we implement the batch branch using the single real joint batch response and the public post-processing described above, including the cached-pair abort rule. We keep the full branch as direct execution from the actual matrix and opening state. Outside the bad events, all cached-pair checks pass and every replaced exposed proof equals the exact proof string generated by the real protocol. The corrupted states and the public transcript are therefore identical.

Hence we get that, there exist PPT adversaries $\mathcal{B}_{\mathsf{eb}}, \mathcal{B}_{\mathsf{can}}$ and $\mathcal{B}_{\mathsf{pi}}$ such that:

$$\begin{aligned}|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq\ & \mathsf{Adv}^{\mathsf{eval\text{-}binding}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{eb}}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{can}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{can}}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{proof\text{-}int}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{pi}}}(\lambda).\end{aligned}$$

Game G_2: replacing the commitment transcript. Game G_2 proceeds as G_1, except that it uses the simulated branch of the commitment's one-shot adaptive simulation-hiding experiment.

Concretely, the game samples

$$\begin{aligned}(\mathsf{crs}, \tau_{\mathsf{sim}}) &\leftarrow\$ \mathsf{Basis2DLCCom.SimSetup}(1^\lambda, k, r, c, \ell, f), \\ (\mathsf{com}, \omega) &\leftarrow\$ \mathsf{Basis2DLCCom.SimCommit}(\mathsf{crs}, \tau_{\mathsf{sim}}).\end{aligned}$$

It still uses the actual dealer input vector $\boldsymbol{s}^{\mathsf{ext}}$, samples $\boldsymbol{M} \leftarrow\$ \mathbb{F}_q^{r\times c}$ independently of commitment generation, and sets

$$\boldsymbol{z} = \boldsymbol{s}^{\mathsf{ext}} + \mathsf{h}(\boldsymbol{M}).$$

It supplies the resulting public information to $\mathcal{A}$ and processes the execution before the adversary's batch query as in G_1.

Upon receiving $\mathcal{A}$'s batch query $(J, \mathcal{T})$, the game sets $T = \{j+1 : j \in \mathcal{T}\}$ and forms the single commitment-game query

$$Q = \begin{cases} \mathsf{Batch}(J, T), & \text{if the dealer is not in } J \text{ and some dealer input remains unreconstructed,} \\ \mathsf{Full}, & \text{otherwise.} \end{cases}$$

The batch query is admissible because $|J| \leq f$ and $|T| < \ell$, as established above. The game answers this query using the simulated commitment component as follows.

- If $Q = \mathsf{Batch}(J, T)$, the game computes

  $$E = \mathcal{L}_{J,T}(\boldsymbol{M}), \qquad \mathit{\Pi} \leftarrow\$ \mathsf{Basis2DLCCom.SimBatch}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{z}, J, T, E).$$

  It then applies exactly the batch post-processing used in G_1. In particular, it materializes the exposed scalar values and cached proofs from $(E, \mathit{\Pi})$, derives corrupted-sender reconstruction proofs from their row strips, replays corrupted local states in the recorded delivery order, and performs the public verification and cache-consistency checks on adversarial messages. Unexposed honest-to-honest plaintexts remain symbolic.

  This post-processing uses only $(\mathsf{crs}, \mathsf{com}, \boldsymbol{z}, J, T, E, \mathit{\Pi})$ and the maintained execution log, together with the adversary's subsequent messages and schedule. Crucially, it does not use $\boldsymbol{M}$ or τ_{sim}.

– If $Q = \mathsf{Full}$, the game computes

$$\mathsf{st}^\star \leftarrow\!\$\ \mathsf{Basis2DLCCom.SimFull}(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{M}).$$

It applies the full-branch post-processing of G_1 using $(\boldsymbol{M}, \mathsf{st}^\star)$: it materializes the execution by running the actual protocol algorithms, and exposes the dealer's state only if the dealer is corrupted.

The game uses the same checks and abort conventions as G_1, and otherwise continues under $\mathcal{A}$'s schedule and outputs its final bit.

Suppose now that $\mathcal{A}$ distinguishes G_1 from G_2. We construct a PPT adversary $\mathcal{B}_{\mathsf{sh}}$ that uses $\mathcal{A}$ to distinguish the real and simulated branches of the commitment's one-shot adaptive simulation-hiding experiment.

The reduction receives crs from its commitment challenger, runs the AVSS adversary and input-selection procedure as in G_1, and submits the actual padded dealer input vector $\boldsymbol{s}^{\mathsf{ext}}$ to the challenger. It uses the challenger's commitment and pads $(\mathsf{com}, \boldsymbol{z})$ to generate the AVSS public transcript. Upon receiving $(J, \mathcal{T})$ from $\mathcal{A}$, it submits exactly the query Q defined above to the commitment challenger.

For a batch query, the challenger returns $E = \mathcal{L}_{J,T}(\boldsymbol{M})$ and a real or simulated joint proof list $\mathit{\Pi}$. For a full query, it returns the challenge matrix $\boldsymbol{M}$ and a real or simulated opening state. In either case, $\mathcal{B}_{\mathsf{sh}}$ applies the corresponding post-processing specified above and outputs $\mathcal{A}$'s final bit.

When the commitment challenger uses its real branch, this execution is distributed exactly as G_1; when it uses its simulated branch, it is distributed exactly as G_2. Therefore,

$$|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| \le \mathsf{Adv}^{\mathsf{one\text{-}shot\text{-}sh}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{sh}}}(\lambda).$$

The same efficiently computable cache-consistency checks and abort rules are applied in both branches. Note that canonicality was used only to justify the transition from G_0 to G_1; this transition makes no canonicality assumption under the simulated CRS.

Game G_3: completing the unrevealed inputs. Under simulated commitment generation, the commitment and its simulation state are independent of the sharing matrix. Moreover, if the code and masking parameters are good (which occurs with overwhelming probability for our parameters), the map h is surjective. Thus we may sample $\boldsymbol{z}$ uniformly first; conditioned on $\boldsymbol{z}$ and the actual secret vector $\boldsymbol{s}^{\mathsf{ext}}$, the matrix is uniform in

$$\{\boldsymbol{X} \in \mathbb{F}_q^{r\times c} : \mathsf{h}(\boldsymbol{X}) = \boldsymbol{z} - \boldsymbol{s}^{\mathsf{ext}}\}.$$

In the full branch all inputs are known, so this is exactly the conditional matrix sampler used by the simulator.

For the batch branch, the actual input vector and $\widetilde{\boldsymbol{s}}$ agree on T. Lemma 2 gives

$$\big(\boldsymbol{s}^{\mathsf{ext}} + \mathsf{h}(\boldsymbol{M}), \mathcal{L}_{J,T}(\boldsymbol{M})\big) \overset{d}{=} \big(\widetilde{\boldsymbol{s}} + \mathsf{h}(\boldsymbol{M}), \mathcal{L}_{J,T}(\boldsymbol{M})\big), \qquad \boldsymbol{M} \leftarrow\!\$\ \mathbb{F}_q^{r\times c}.$$

Since the pad marginal distribution is uniform on both sides, the conditional distribution of the leakage is also equal at each fixed pad vector. This holds for all admissible J, T, hence allowing the adversary to adaptively pick the batch corruption query.

Consequently, we can replace the actual leakage E by $\widetilde{E}$ sampled in (76). At this point the proof generator is already $\mathsf{SimBatch}$, whose inputs are only

$$(\mathsf{crs}, \tau_{\mathsf{sim}}, \mathsf{com}, \omega, \boldsymbol{z}, J, T, E).$$

Replacing the leakage by an identically distributed sample preserves the joint proof response, the cache checks, and the entire subsequent post-processing. Hence, conditioned on good code and mask parameters, which occur with overwhelming probability, $\Pr[\mathsf{G}_2 = 1] = \Pr[\mathsf{G}_3 = 1]$.

Game G_4: the ideal execution. Game G_3 now uses only the information supplied to the AVSS simulator: the public transcript, the complete declaration of the adversary's query, the reconstructed inputs, and the full dealer input vector when the full branch is selected. Its pad sampling, conditional leakage or matrix sampling, and single commitment-simulation call are precisely those specified before the materialization paragraphs. The remaining computation is the simulator described above.

In the batch branch, it provides the jointly sampled exposed proofs and the decoded scalar states from the leakage, while unexposed honest messages remain symbolic. In the full branch, it materializes the execution from the returned state and the sampled matrix. Replacing the event-log presentation by this online implementation changes neither the adversary's view nor the source indices and control decisions it observes. Hence the final game is the ideal execution with Sim.

Combining the games, for suitable PPT reductions,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{one\text{-}shot\text{-}secrecy}}_{\mathsf{LatBingo},\mathcal{A}}(\lambda) \le\ & \varepsilon_{\mathsf{code}} + \varepsilon_{\mathsf{mask}} \\ & + \mathsf{Adv}^{\mathsf{eval\text{-}binding}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{eb}}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{can}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{can}}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{proof\text{-}int}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{pi}}}(\lambda) \\ & + \mathsf{Adv}^{\mathsf{one\text{-}shot\text{-}sh}}_{\mathsf{Basis2DLCCom},\mathcal{B}_{\mathsf{sh}}}(\lambda).\end{aligned}$$

All error terms are negligible under the stated parameters. This proves the theorem. □

6.2 Instantiation and Efficiency

We now give a concrete instantiation of $\mathsf{LatBingo}$, in which the two bounded-encoder linear codes of Section 5.2.1 are instantiated with random integer encoder matrices, and discuss how to set the parameters of our commitment scheme and of the AVSS protocol. We stress that the results of Section 6.1 hold for all bounded-encoder linear codes satisfying Definition 7.

6.2.1 Instantiation with Random Integer Encoder Matrices Component codes. Recall that q is an odd prime, $\boldsymbol{M} \in \mathbb{F}_q^{r\times c}$, and that the coefficient openings belong to $\mathbb{Z}^m$. Fix an integer $2 \le D_{\mathsf{code}} < q$. Sample every entry of $\boldsymbol{G}_{\mathsf{code}} \in \mathbb{Z}^{k\times r}$ and $\boldsymbol{H}_{\mathsf{code}} \in \mathbb{Z}^{k\times c}$ independently and uniformly from $\{0, \dots, D_{\mathsf{code}} - 1\}$. Then, the encoder bounds satisfy

$$L_G \le r(D_{\mathsf{code}} - 1), \qquad L_H \le c(D_{\mathsf{code}} - 1). \tag{79}$$

Lemma 4 (Reconstruction thresholds). *Let $1 \le d \le t \le k$, and sample every entry of $\boldsymbol{X} \in \mathbb{Z}^{k\times d}$ independently and uniformly from $\{0,\dots,D_{\mathsf{code}}-1\}$. Then*

$$\begin{aligned}\Pr\big[\exists I \subseteq [k],\ |I| = t : \operatorname{rank}_{\mathbb{F}_q}(\boldsymbol{X}_I) < d\big] &\le \binom{k}{t}\binom{t}{d-1} D_{\mathsf{code}}^{-(t-d+1)} \\ &\le 3^k D_{\mathsf{code}}^{-(t-d+1)}.\end{aligned} \tag{80}$$

Consequently, if $t-d \ge \delta k$ for a fixed constant $\delta > 0$, a sufficiently large constant D_{code} makes this probability at most $\exp(-2k)$.

Proof. Fix a set I of size t. If $\operatorname{rank}(\boldsymbol{X}_I) < d$, there is a subset $U \subseteq I$ of size $d-1$ whose rows span all rows of $\boldsymbol{X}_I$. Fix U and condition on its rows. Their span has dimension at most $d-1$, so there is a nonzero vector $\boldsymbol{z} \in \mathbb{F}_q^d$ such that $\boldsymbol{X}_U\boldsymbol{z} = \boldsymbol{0}$. For an independent remaining row $\boldsymbol{x}^\top$, fix all coordinates except one for which the corresponding coordinate of $\boldsymbol{z}$ is nonzero. The equation $\boldsymbol{x}^\top\boldsymbol{z} = 0$ uniquely determines that coordinate over $\mathbb{F}_q$. Each required value has probability at most $1/D_{\mathsf{code}}$. Hence the probability that all $t-d+1$ remaining rows belong to the span is at most $D_{\mathsf{code}}^{-(t-d+1)}$.

Lastly, we union bound over the $\binom{t}{d-1}$ possible sets U and the $\binom{k}{t}$ possible sets I. The inequality $\binom{k}{t}\binom{t}{d-1} \le 3^k$ gives (80). The choice $D_{\mathsf{code}} \ge \exp((\log 3 + 2)/\delta)$ proves the last statement. □

Fix reconstruction thresholds t_c^G, t_c^H. A good code setup means that every set of at least t_c^G rows of $\boldsymbol{G}_{\mathsf{code}}$ has rank r over $\mathbb{F}_q$, and every set of at least t_c^H rows of $\boldsymbol{H}_{\mathsf{code}}$ has rank c over $\mathbb{F}_q$. Conditioned on this event, there are decoder matrices satisfying

$$\boldsymbol{D}_G(I)(\boldsymbol{G}_{\mathsf{code}})_I = \boldsymbol{I}_r, \qquad \boldsymbol{D}_H(J)(\boldsymbol{H}_{\mathsf{code}})_J = \boldsymbol{I}_c \quad \text{over } \mathbb{F}_q. \tag{81}$$

Here, $|I| \ge t_c^G$ and $|J| \ge t_c^H$. To compute these matrices, choose r or c independent rows and compute the corresponding inverse over $\mathbb{F}_q$. We use a fixed public ordering of indices to choose these rows.

Encoding and decoding cost. Recall that m denotes the number of coordinates of a coefficient opening and of a point proof. The dealer obtains all k^2 grid proofs by applying the two encoder matrices to each of the m coordinates of the coefficient-opening state, which costs $O(k^3 m)$ field operations in total. Decoding from a source set requires computing the decoder matrix $\boldsymbol{D}_G(I)$ or $\boldsymbol{D}_H(J)$ by Gaussian elimination over $\mathbb{F}_q$, which costs $O(k^3)$ field operations, and applying it to the source values and to the m coordinates of the source proofs, which costs $O(k^2 m)$ field operations.

6.2.2 Setting the Parameters

We first collect the requirements on the parameters and explain their roles.

Required constraints. We use the encoder matrices and reconstruction thresholds of (15) and (17), an odd prime modulus q, a corruption bound $0 \le f \le k$, and a packing parameter $\ell \ge 1$.

1. *Short directions and packing masks.* For integers $B_G, B_H \ge 1$, we require $q > 2\max\{B_G, B_H\}$ and $2 \le D \le q$. Recall that the mask vector elements are sampled independently and uniformly

from $\{0, \ldots, D-1\}$. The counts from (39) give:

$$N_G = \frac{(B_G+1)^r}{(2B_G L_G+1)^f},$$
$$N_H = \frac{(B_H+1)^c}{(2B_H L_H+1)^f \left(2B_H c(D-1)+1\right)^{\ell-1}},$$
$$N_f = \sum_{j=0}^{f} \binom{k}{j} \le 2^k.$$

We require $N_G, N_H > 1$ and

$$\varepsilon_{\mathsf{mask}} = \min\left\{1,\ N_f\left(D^{-\log_q N_G} + \ell D^{-\log_q N_H}\right)\right\} = \mathrm{negl}(\lambda). \tag{82}$$

By Lemma 1, this guarantees short directions simultaneously for every corruption set of size at most f and every packing position. This is crucial for the one-shot adaptive simulation hiding property of Basis2DLCCom.

2. *Statistical error and Gaussian tails.* We require $0 < \varepsilon \le 1/4$ with $\varepsilon = \mathsf{negl}(\lambda)$, and

$$\tau = rc\,\frac{1+\varepsilon}{1-\varepsilon}\,2^{-m} = \mathsf{negl}(\lambda). \tag{83}$$

The smoothing errors in the one-shot adaptive hiding proof contribute $2(c+\ell)\varepsilon/(1-\varepsilon)$; setting ε to be negligible allows this error to be negligible. The term τ, defined in (52), bounds the probability that any of the rc coefficient openings during Gaussian sampling exceeds its prescribed norm bound. Then, a sufficient dimension condition to make τ negligible is

$$m \ge \lambda + \lceil \log_2(rc) \rceil. \tag{84}$$

3. *Sampling and smoothing widths.* We use the preimage-sampling guarantee from the preliminaries, following the gadget-trapdoor formulation of [MP12]. For the integer-lattice construction, $s_1(\boldsymbol{T})$ denotes the spectral norm of the integer trapdoor matrix $\boldsymbol{T}$. Let

$$d_B = rm + d_g, \qquad \Lambda = \Lambda_q^{\perp}(\boldsymbol{A}), \qquad \Lambda_{\boldsymbol{G}} = \Lambda_q^{\perp}(\boldsymbol{G}),$$

where d_B is the number of columns of $\boldsymbol{B}_r$ and $\boldsymbol{G}$ is the binary gadget matrix. We require

$$\begin{aligned} \sigma_0 &\ge 2s_1(\widetilde{\boldsymbol{R}}) \cdot \omega\left(\sqrt{\log d_B}\right), \\ \sigma &\ge \max\left\{ \begin{array}{cc} 2s_1(\boldsymbol{T}_r) \cdot \omega\left(\sqrt{\log d_B}\right), & 2s_1(\boldsymbol{R}) \cdot \omega\left(\sqrt{\log m}\right), \\ \eta_{\varepsilon/(2r)}(\Lambda), & \eta_{\varepsilon}(\Lambda_{\boldsymbol{G}}), \quad B_G B_H \sqrt{rc}\,\eta_{\varepsilon}(\Lambda) \end{array} \right\}. \end{aligned} \tag{85}$$

The bound on σ_0 permits sampling the columns of the public trapdoor $\boldsymbol{T}_r$ using $\widetilde{\boldsymbol{R}}$. The first two bounds on σ permit the real commitment sampler to use $\boldsymbol{T}_r$ and the simulator's state sampler to use $\boldsymbol{R}$, respectively. All of these bounds are standard in lattice-based cryptography to ensure that SamplePre algorithm outputs are close to a discrete gaussian distribution. Let ϵ_0 bound the statistical error per sampled column in the generation of $\boldsymbol{T}_r$, and let $\varepsilon_{\mathsf{samp}}$ bound

the statistical error per invocation of either state sampler. The sampling guarantees make both errors negligible.

The next two bounds on σ justify Gaussian splitting and replacing the public commitment vectors by uniform vectors. Finally, $\sigma \geq B_G B_H \sqrt{rc}\, \eta_\varepsilon(\Lambda)$ is needed for simulation hiding: every direction supplied by Lemma 1 has Euclidean norm at most $K_{\mathsf{hid}} = B_G B_H \sqrt{rc}$, so we need the above condition to be able to apply Lemma 3.

4. *Opening norms and the binding reduction.* We set the coefficient-opening bound and the largest point-proof bound to

$$\begin{gathered} \beta_s = 2\sigma\sqrt{m + \lambda + \ln(2rc)}, \qquad B_{\max} = L_{\mathcal{I}} L_{\mathcal{J}} \beta_s, \\ \beta_{\mathsf{PB}} \geq 2B_{\max}, \qquad q > 2B_{\max}. \end{gathered} \tag{86}$$

For every canonical point proof, the coefficient bound gives the advertised verification bound $B(i,j) = \|\boldsymbol{g}_i\|_1 \|\boldsymbol{h}_j\|_1 \beta_s$. The last two inequalities allow the difference of two accepting proofs to be used as a short, nonzero PowerBASIS solution in the binding and proof-canonicality reductions. We also assume that PowerBASIS is hard at the resulting parameters.

5. *AVSS repair, reconstruction, and quorums.* Recall that a row needs t_c^H valid entries at distinct ordinary columns, and a column needs t_c^G valid entries at distinct ordinary rows. The correctness and termination arguments require

$$t_c^G \leq k - 2f, \qquad t_c^H \leq k - f, \qquad k > 3f. \tag{87}$$

Setting the parameters. We now give concrete parameter choices satisfying the preceding constraints for the bounded-encoder linear codes with random integer encoder matrices of Section 6.2.1. Fix a constant $0 < \epsilon \leq 1$, and take $\lambda \leq k \leq \mathrm{poly}(\lambda)$. Here ϵ is a constant; ε denotes the statistical error parameter in Theorem 5. Set

$$\begin{gathered} \eta = \epsilon/8, \qquad k = \lceil (3+\epsilon) f \rceil, \\ r = \lceil (1+\eta) f \rceil, \qquad c = \lfloor (2-\eta) f \rfloor, \qquad \ell = \lfloor (1-2\eta) f \rfloor, \\ t_c^G = k - 2f, \qquad t_c^H = k - f. \end{gathered} \tag{88}$$

For sufficiently large f, these choices give

$$\begin{aligned} r - f &\geq \eta f, & c - f - \ell + 1 &\geq \eta f, \\ t_c^G - r &\geq (\epsilon - \eta) f - 1, & t_c^H - c &\geq (\epsilon + \eta) f. \end{aligned} \tag{89}$$

In particular, $r, c, \ell = \Theta(k)$ and $f + 1 \leq r \leq t_c^G$, $f + \ell \leq c \leq t_c^H$. Both $t_c^G - r$ and $t_c^H - c$ are at least δk for $\delta = \epsilon/16$, for sufficiently large k. Choose a constant D_{code} as in Lemma 4. The total code failure probability is at most $2\exp(-2k)$, which is negligible in λ. Note that the setup algorithm simply samples the entries; it does not need to check that the properties stated in Lemma 4 actually hold.

Short directions and packing masks. Set $a = 4/\eta$ and $B_G = B_H = B = \lceil k^a \rceil$. Recall the counts from (39):

$$N_G = \frac{(B+1)^r}{(2BL_G + 1)^f}, \qquad N_H = \frac{(B+1)^c}{(2BL_H + 1)^f (2Bc(D-1) + 1)^{\ell - 1}}.$$

For any fixed packing alphabet size D, the row bounds in (79) give

$$\log N_G \geq \big(a(r-f)-f\big)\log k - O(k) \geq 3f\log k - O(k),$$
$$\log N_H \geq \big(a(c-f-\ell+1)-(f+\ell-1)\big)\log k - O(k) \geq 2f\log k - O(k).$$

The constants in the $O(k)$ terms may depend on $\epsilon, D_{\mathsf{code}}, D$. For sufficiently large k, it follows that $\log N_G \geq 2f\log k$ and $\log N_H \geq f\log k$.

Gaussian parameters and sampling width. Use the preliminary lattice dimensions $n = \Theta(\lambda)$ and $m = \Theta(n\log q)$, with $m \geq \lambda + \lceil \log_2(rc)\rceil$. Take the statistical and sampling errors to be negligible in λ, with $\varepsilon \leq 2^{-\lambda}k^{-3}$. Choose a polynomially bounded σ_0 satisfying the first condition in (85). For a fixed exponent C_0, let k^{C_0} bound the preliminary sampling widths, the required smoothing parameters, and $2\sqrt{m+\lambda+\ln(2rc)}$ on the good setup event. These bounds depend polynomially on the dimensions and $\log q$. Thus C_0 can be chosen independently of the fixed exponent Q in a modulus $q \in [k^Q, 2k^Q]$. Choose

$$Q > 2a+5+2C_0, \qquad \mu = \frac{1}{6(Q+1)}, \qquad D = 2^{\lceil 4/\mu\rceil}, \qquad q \in [k^Q, 2k^Q] \text{ prime.} \tag{90}$$

For sufficiently large k, $q > \max\{D, D_{\mathsf{code}}, 2B\}$ and $f \geq k/5$. The preceding counts therefore give $\log_q N_G, \log_q N_H \geq \mu k$. Consequently,

$$\varepsilon_{\mathsf{mask}} \leq 2^k(1+\ell)D^{-\mu k} \leq (1+\ell)2^{-3k}. \tag{91}$$

The choice of D after Q changes the constants in the $O(k)$ terms above, but does not change these asymptotics.

The short directions have $K_{\mathsf{hid}} = B^2\sqrt{rc} = O(k^{2a+1})$. For sufficiently large constants C_1, C_2, choose

$$\sigma = C_1 k^{2a+1+C_0}, \qquad \beta_s = 2\sigma\sqrt{m+\lambda+\ln(2rc)} \leq C_2 k^{2a+1+2C_0}. \tag{92}$$

Together with σ_0, this satisfies the sampling and smoothing conditions in (85), including $\sigma \geq K_{\mathsf{hid}}\eta_\varepsilon(\Lambda)$. All ordinary and packing query vectors have row norm $O(k)$. Hence

$$B_{\max} = L_{\mathcal{I}}L_{\mathcal{J}}\beta_s = O(k^{2a+3+2C_0}), \qquad 2B_{\max} < q$$

for sufficiently large k. The constants may depend on ϵ. The PowerBASIS assumption is used at the resulting parameters, with power length r and solution bound at least $2B_{\max}$.

Corollary 1 (Security of LatBingo with bounded-encoder linear codes with random integer encoder matrices). *Assume reliable broadcast and authenticated private channels. Assume the PowerBASIS assumption of Section 3.5 at the parameters above, and that Π_{wf} is complete, knowledge sound, and zero-knowledge as required by the commitment security theorems. For every fixed $0 < \epsilon \leq 1$ and sufficiently large parameters satisfying (88)–(92), LatBingo, using Basis2DLCCom with the bounded-encoder linear codes with random integer encoder matrices of Section 6.2.1, is an f-resilient packed AVSS for $\ell = \Theta(k)$ field-element secrets with $k = (3+\epsilon)f + O(1)$ parties. It satisfies correctness and termination, and one-shot adaptive secrecy.*

Proof. The code and mask failure probabilities are negligible, and the parameters satisfy the width and norm requirements above. Theorems 1, 2, 3, 4 and 5 give correctness, evaluation binding,

computational proof canonicality, proof interpolation in both directions, and one-shot adaptive simulation-hiding of Basis2DLCCom with these codes. The dimensions satisfy

$$k > 3f, \qquad f+1 \le r \le t_c^G \le k-2f, \qquad f+\ell \le c \le t_c^H \le k-f.$$

These are the parameter conditions of Theorem 6, which gives the claim. □

6.2.3 Efficiency of LatBingo We now prove the efficiency of our packed AVSS protocol LatBingo.

Throughout this section, communication bounds are stated as functions of k, suppressing polynomial factors in the security parameter λ. Thus the bounds $O(k)$ and $O(k^2)$ below correspond to $O(\mathsf{poly}(\lambda)\cdot k)$ and $O(\mathsf{poly}(\lambda)\cdot k^2)$ words, respectively.

Theorem 10 (Efficiency of LatBingo). *Under the parameter and security assumptions of Theorem 6, suppose that $r, c, \ell = O(k)$, the lattice dimensions are polynomial in λ, and one field element, one commitment column, and one point-opening proof each have length $O(\mathsf{poly}(\lambda))$. Let L_{wf} denote the maximum permitted length of the well-formedness proof under the counting convention above. Then, the commitment has size $O(k)+L_{\mathsf{wf}}$, and LatBingoShare shares ℓ field elements using $O(k^2+kL_{\mathsf{wf}})$ words and $O(k^2)$ messages from nonfaulty parties. In particular, assuming $L_{\mathsf{wf}} = O(k)$, the commitment has size $O(k)$, its reliable broadcast together with the pads costs $O(k^2)$ words, and the total sharing communication is $O(k^2)$ words. When $\ell = \Theta(k)$, this shares $\Theta(k)$ secrets with total communication $O(k^2)$.*

Every participating nonfaulty party completes sharing in $O(1)$ asynchronous rounds after the first nonfaulty party does so; if the dealer is nonfaulty, all participating nonfaulty parties complete sharing in $O(1)$ rounds. For every j, LatBingoReconst(j) requires $O(k^2)$ words and $O(k^2)$ messages, and takes $O(1)$ asynchronous rounds after the participating nonfaulty parties complete sharing and invoke reconstruction. The reusable CRS has size $O(k^2+L_{\mathsf{crs,wf}})$. Its size is $O(k^2)$ if additionally $L_{\mathsf{crs,wf}} = O(k^2)$.

Proof. We count communication within a protocol instance, with crs already available to all parties. Note that the same public parameters can be used for multiple instances under the corresponding multi-instance security guarantee.

Recall that the secrets are elements of $\mathbb{F}_q$. Let $L_{\mathbb{F}}$ be the number of words used for one such element, L_{col} the size of one commitment column, and L_{pt} the size of one point-opening proof. All coefficient bit lengths are polynomially bounded in the security parameter. In the integer-lattice instantiation, one commitment column is a vector in $\mathbb{F}_q^n$, and one point-opening proof is an integer vector of dimension m with the prescribed norm bound. Thus, under the theorem's dimension and length assumptions, each of these is $O(\mathsf{poly}(\lambda))$ words. With security-parameter factors suppressed, we write $L_{\mathbb{F}}, L_{\mathsf{col}}, L_{\mathsf{pt}} = O(1)$.

The commitment consists of c commitment columns and one well-formedness proof. Its length is therefore

$$|\mathsf{com}| = cL_{\mathsf{col}} + L_{\mathsf{wf}} = O(k) + L_{\mathsf{wf}}.$$

We establish the sharing bound $O(k^2+kL_{\mathsf{wf}})$ before using $L_{\mathsf{wf}} = O(k)$. This keeps the contribution of the argument system explicit and makes the final quadratic bound conditional on its proof size.

Communication of sharing. The dealer reliably broadcasts

$$\big(\mathsf{com}, \boldsymbol{z}\big) = \big(((\mathsf{com}_b)_{b=0}^{c-1}, \pi_{\mathsf{wf}}), \boldsymbol{z}\big).$$

There are c commitment columns and ℓ pads, so its payload has size

$$L_{\mathsf{RBC}} = cL_{\mathsf{col}} + \ell L_{\mathbb{F}} + L_{\mathsf{wf}} = O(k) + L_{\mathsf{wf}}.$$

Using the reliable broadcast protocol of [DXR21], a payload of length L_{RBC} contributes $O(kL_{\mathsf{RBC}} + k^2)$ words and $O(k^2)$ messages from nonfaulty parties. Consequently, this broadcast costs $O(k^2 + kL_{\mathsf{wf}})$ words. Under $L_{\mathsf{wf}} = O(k)$, the payload is $O(k)$ and the reliable broadcast costs $O(k^2)$ words.

A nonfaulty dealer also sends one "data" message to each party. Each message contains a row vector of c field elements and k evaluation-value/proof pairs. Its size is

$$cL_{\mathbb{F}} + k(L_{\mathbb{F}} + L_{\mathsf{pt}}) = O(k)$$

words. The k private dealer messages therefore require $O(k^2)$ words and $O(k)$ messages.

The flags $\mathsf{sentRow}_i$ and $\mathsf{sentColumn}_i$ ensure that each nonfaulty party sends at most one "row" message and one "column" message to each recipient. Every such message contains a field element and a point-opening proof, and hence occupies $O(L_{\mathbb{F}} + L_{\mathsf{pt}}) = O(1)$ words. Together these transmissions cost $O(k^2)$ words and $O(k^2)$ messages. Recipients verify the point proofs against the commitment already obtained by reliable broadcast, so the well-formedness proof is accounted for in that broadcast. The $\mathsf{sentDone}_i$ flag bounds the total number of *done* messages by k^2, contributing $O(k^2)$ words and messages. Local decoding and proof interpolation do not add communication.

Thus the total sharing communication is

$$O(k^2 + kL_{\mathsf{wf}}) + O(k^2) + O(k^2) = O(k^2 + kL_{\mathsf{wf}})$$

words and $O(k^2)$ messages. Assuming $L_{\mathsf{wf}} = O(k)$, the total is $O(k^2)$ words. For $\ell = \Theta(k)$, a single instance therefore shares $\Theta(k)$ field elements at amortized communication $O(k)$ words per secret.

Round complexity with a nonfaulty dealer. All round bounds refer to participating nonfaulty parties and hold except with negligible probability. Messages received before their prerequisites hold are buffered.

If the dealer is nonfaulty, every nonfaulty party receives the reliably broadcast pair and its private row message within a constant number of rounds. Correctness of $\mathsf{Basis2DLCCom}$ makes all supplied proofs accepting. Thus every nonfaulty party obtains $\boldsymbol{\alpha}_i \neq \perp$ and a valid row-opening set $\mathcal{P}_i^X$. Each then sends its row messages after a row-direction $\mathsf{GetProofs}$ call using t_c^H sources.

Within one further round, every nonfaulty party receives row messages from at least $k - f \geq t_c^G$ nonfaulty senders, so $|\mathcal{P}_i^Y| = t_c^G$. It decodes $\boldsymbol{\beta}_i$, sends column messages, and meets the local conditions for sending *done*. Another round delivers *done* from the $k - f$ nonfaulty parties. All such parties therefore terminate sharing after $O(1)$ rounds.

Round complexity after the first nonfaulty completion. Suppose the first nonfaulty party completes sharing at time t. It has received *done* from $k - f$ distinct parties, of which at least $k - 2f \geq t_c^G$ are nonfaulty. Let $\mathcal{H}$ be this set of nonfaulty senders. Before sending *done*, every party in $\mathcal{H}$ obtained a valid row-opening source set of size at least t_c^H, and a valid column-opening source set of size t_c^G. It had sent both its row and its column messages. Proof interpolation makes all those transmitted proofs accepting.

Reliable broadcast gives every participating nonfaulty party the same commitment and pads within $O(1)$ rounds after time t. The row messages from $\mathcal{H}$ supply at least t_c^G valid column-direction sources for each such party. It can therefore decode its $\boldsymbol{\beta}_i$ and send valid column messages, if it has not already done so. Within one more delivery round, every nonfaulty party receives column

messages from at least $k - f \geq t_c^H$ nonfaulty senders. Any party whose $\boldsymbol{\alpha}_i$ is still undefined repairs it with the H decoder and sends its row messages.

Every nonfaulty party now satisfies the local *done* conditions. One additional round delivers the $k - f$ nonfaulty *done* messages to all parties, which complete sharing. Hence the number of rounds after the first nonfaulty completion is $O(1)$. This argument crucially relies on $k - 2f \geq t_c^G$ for the first propagation step and $k - f \geq t_c^H$ for the second.

If the reliably broadcast commitment contains a well-formedness proof that does not verify, Basis2DLCCom.Verify rejects every opening under that commitment. No nonfaulty party can then meet the conditions for sending *done* or completing sharing. Thus we do not need to worry about this case in the preceding argument. In the case when the dealer is nonfaulty, this occurs with only negligible probability by completeness of the proof system. The same reasoning applies to a malformed pad vector.

Communication and rounds of reconstruction. In LatBingoReconst(j), each nonfaulty party sends one message to every recipient. It contains an index, a field value, and one point-opening proof, so it occupies $O(1)$ words. The total is $O(k^2)$ words and $O(k^2)$ messages. Generating the special-column proof from t_c^H retained sources requires only local computation.

After sharing, every nonfaulty party has these t_c^H valid row-direction sources. Proof interpolation returns an accepting opening at $(i, -j)$. Once all participating nonfaulty parties invoke reconstruction, one delivery round supplies at least $k - f \geq t_c^G$ valid reconstruction messages to each nonfaulty recipient. Each applies the G decoder, evaluates against $\boldsymbol{b}_0^\top$, subtracts the resulting mask from its stored pad, and outputs s_j. Hence reconstruction takes $O(1)$ asynchronous rounds.

Size of the common reference string. We count the explicitly published code and lattice parameters, suppressing polynomial factors in the security parameter λ. The two encoder matrices

$$\boldsymbol{G}_{\mathsf{code}} \in \mathbb{Z}^{k \times r}, \qquad \boldsymbol{H}_{\mathsf{code}} \in \mathbb{Z}^{k \times c}$$

contain kr and kc coefficients, respectively. Since $r, c = O(k)$, each matrix contributes $O(k^2)$ words. The CRS also contains the masking vectors

$$\boldsymbol{b}_0 \in \mathbb{F}_q^r, \qquad \boldsymbol{a}_1, \ldots, \boldsymbol{a}_\ell \in \mathbb{F}_q^c.$$

There are $\ell + 1 = O(k)$ such vectors, each of length $O(k)$, so their total contribution is $r + \ell c = O(k^2)$ field elements.

For the lattice parameters, put $d_g = n\lceil \log_2 q \rceil$. The published matrices are

$$\boldsymbol{A} \in \mathbb{F}_q^{n \times m}, \qquad \boldsymbol{W} \in \mathbb{F}_q^{n \times n}, \qquad \boldsymbol{T}_r \in \mathbb{Z}^{(rm + d_g) \times r d_g}.$$

The first two matrices contain $nm + n^2$ field elements. The public trapdoor $\boldsymbol{T}_r$ contains

$$(rm + d_g) r d_g = r^2 m d_g + r d_g^2$$

integer entries, and hence contributes $O(k^2)$ words under our counting convention.

More explicitly, let L_{entry} bound the word length of a published coefficient, and let $L_{\mathsf{crs,wf}}$ denote the size of the well-formedness proof system's public parameters. The total CRS size is

$$O\big([\, kr + kc + r + \ell c + nm + n^2 + (rm + d_g) r d_g \,] L_{\mathsf{entry}} + L_{\mathsf{crs,wf}}\big).$$

Since the lattice dimensions and coefficient lengths are polynomial in λ, this is $O(k^2 + L_{\mathsf{crs,wf}})$ words. In particular, the total size is $O(k^2)$ whenever $L_{\mathsf{crs,wf}} = O(k^2)$. □

Corollary 2. *Suppose the component-code and packing parameters satisfy Theorem 6 with $\ell = \Theta(k)$, the communication-length assumptions of the efficiency theorem, and $L_{\mathsf{wf}} = O(k)$. For any $\mu = \Omega(k)$, there exists a semi-packed AVSS protocol sharing μ secrets requiring*

$$O\big((k^2 + kL_{\mathsf{wf}})\lceil \mu/\ell \rceil\big) = O\big(k^2 \lceil \mu/\ell \rceil\big) = O(k\mu)$$

words (with factors polynomial in λ suppressed). Reconstruction of one field element costs $O(k^2)$ words and $O(k^2)$ messages, and takes $O(1)$ rounds.

Proof. Let us set $\nu+1 = \ell$ for LatBingo. We partition the μ field-element inputs into $\lceil \mu/\ell \rceil = \Theta(\mu/k)$ blocks, each containing at most ℓ inputs, and pad the last block with zeros if needed. The dealer can run LatBingoShare independently on these blocks. Adding the per-instance communication gives the stated sharing cost. Calling LatBingoReconst in the relevant instance of AVSS reconstructs one input at the per-instance reconstruction cost.

□

6.2.4 Batch Reconstruction for LatBingo We now show how to extend LatBingo to support (i) reconstructing all the packed secrets and, (ii) reconstructing the sum of shared secrets, both with the same cost as reconstructing a single secret.

On Full Batch Reconstruction. Similar to [AJM+23], we support reconstructing all $\nu + 1$ secrets of one instance at once, while sending only $O(\mathsf{poly}(\lambda) \cdot k^2)$ words. Every secret is $s_j = z_{j+1} - \boldsymbol{b}_0^\top \boldsymbol{M} \boldsymbol{a}_{j+1}$. Instead of recovering each $\boldsymbol{\beta}_{-j} = \boldsymbol{M}\boldsymbol{a}_{j+1}$ separately, parties can recover the single row $\boldsymbol{\alpha}_0 = \boldsymbol{b}_0^\top \boldsymbol{M}$ and evaluate it on all the packing vectors $\boldsymbol{a}_{j+1}$.

Each party already has t_c^G column-direction sources in $\mathcal{P}_i^Y$. They allow it to produce an opening at $(0, i)$, whose value is $\boldsymbol{b}_0^\top \boldsymbol{\beta}_i = \boldsymbol{\alpha}_0 \boldsymbol{h}_i$. These values are H-code symbols of $\boldsymbol{\alpha}_0^\top$, so t_c^H senders suffice to recover that row. Algorithm 5 specifies the protocol.

Algorithm 5 $\mathsf{LatBingoReconstructBatch}_i()$

$\mathcal{S}_i^{\mathsf{batch}} \leftarrow \emptyset$.
$\{(0, v_i, \pi_i)\} \leftarrow \mathsf{Basis2DLCCom.GetProofs}(\mathsf{crs}, \mathsf{com}_i, \mathsf{col}, i, \mathcal{P}_i^Y, \{0\})$. ▷ $v_i = \boldsymbol{b}_0^\top \boldsymbol{\beta}_i = \boldsymbol{\alpha}_0 \boldsymbol{h}_i$.
Send $\langle$"rec", $\mathsf{batch}, v_i, \pi_i\rangle$ to all parties.
upon receiving the first $\langle$"rec", $\mathsf{batch}, v_u, \pi_u\rangle$ from P_u, **do**
  if $\mathsf{Basis2DLCCom.Verify}(\mathsf{crs}, \mathsf{com}_i, (0, u), v_u, \pi_u) = 1$ **then**
    $\mathcal{S}_i^{\mathsf{batch}} \leftarrow \mathcal{S}_i^{\mathsf{batch}} \cup \{(u, v_u)\}$.
    if $|\mathcal{S}_i^{\mathsf{batch}}| = t_c^H$ **then**
      $\boldsymbol{\alpha}_0 \leftarrow \big(\mathsf{Decode}^H(\mathcal{S}_i^{\mathsf{batch}})\big)^\top$. ▷ $\boldsymbol{\alpha}_0 = \boldsymbol{b}_0^\top \boldsymbol{M}$.
      for $j = 0, \ldots, \nu$ **do**
        $\widehat{s}_j \leftarrow (\boldsymbol{z}_i)_{j+1} - \boldsymbol{\alpha}_0 \boldsymbol{a}_{j+1}$.
      end for
      Output $(\widehat{s}_0, \ldots, \widehat{s}_\nu)$ and **terminate**.
    end if
  end if

On Reconstructing Sum of Secrets. Additionally, the linear verification equations support reconstructing the sum of secrets shared by different dealers. If com_1 and com_2 have matrix witnesses $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, adding their commitment-column vectors and their opening vectors gives the verification equation for $\boldsymbol{M}_1 + \boldsymbol{M}_2$. The opening norm bound grows by a factor of two. For t dealers, it grows by a factor of t. Hence the commitments satisfy a (weak) notion of linear homomorphism.

Let dealers be a common nonempty set of dealers whose sharing instances each participant has completed, and fix $j \in \{0, \ldots, \nu\}$. If we define $r_{u,j} \in \mathbb{F}_q$ to be the j-the secret in the LatBingoShare invocation with u as dealer, then parties can reconstruct $\sum_{u \in \mathsf{dealers}} r_{u,j}$. Essentially, all the parties sum the reconstruction mask values and their proofs, decode the summed mask column, and subtract its value at $\boldsymbol{b}_0$ from the sum of the corresponding public pads. Algorithm 6 gives the formal procedure.

Algorithm 6 $\mathsf{LatBingoReconstructSum}_i(\mathsf{dealers}, j)$ for $j \in \{0, \ldots, \nu\}$

Require: A common nonempty set dealers; P_i has completed LatBingoShare for every $u \in \mathsf{dealers}$ under the same CRS.

$t \leftarrow |\mathsf{dealers}|$; $\mathcal{S}^{\Sigma}_{i,j} \leftarrow \emptyset$.
for all $u \in \mathsf{dealers}$ **do**
  Parse $\mathsf{com}_{i,u} = ((\mathsf{com}_{i,u,b})_{b=0}^{c-1}, \pi_{\mathsf{wf},i,u})$ and retain its pad vector $\boldsymbol{z}_{i,u}$.
  Choose $\mathcal{Q}^X_{i,u} \subseteq \mathcal{P}^X_{i,u}$ with $|\mathcal{Q}^X_{i,u}| = t^H_c$.
  $\{(-j, z^{(u)}_{i,-j}, \boldsymbol{\pi}^{(u)}_{i,-j})\} \leftarrow \mathsf{Basis2DLCCom.GetProofs}(\mathsf{crs}, \mathsf{com}_{i,u}, \mathsf{row}, i, \mathcal{Q}^X_{i,u}, \{-j\})$.
end for
for $b = 0, \ldots, c-1$ **do**
  $\mathsf{com}^{\Sigma}_{i,b} \leftarrow \sum_{u \in \mathsf{dealers}} \mathsf{com}_{i,u,b}$.
end for
$z^{\Sigma}_{i,-j} \leftarrow \sum_{u \in \mathsf{dealers}} z^{(u)}_{i,-j}$.
$\boldsymbol{\pi}^{\Sigma}_{i,-j} \leftarrow \sum_{u \in \mathsf{dealers}} \boldsymbol{\pi}^{(u)}_{i,-j}$.
$z^{\mathsf{pad},\Sigma}_{i,j+1} \leftarrow \sum_{u \in \mathsf{dealers}} (\boldsymbol{z}_{i,u})_{j+1}$.
Send $\langle$"rec", sum, $j, z^{\Sigma}_{i,-j}, \boldsymbol{\pi}^{\Sigma}_{i,-j}\rangle$ to all parties.
upon receiving the first $\langle$"rec", sum, $j, z^{\Sigma}_{v,-j}, \boldsymbol{\pi}^{\Sigma}_{v,-j}\rangle$ from P_v, **do**
  $\widehat{\mathsf{com}}^{\Sigma}_{i,j} \leftarrow \sum_{b=0}^{c-1} (\boldsymbol{a}_{j+1})_b \mathsf{com}^{\Sigma}_{i,b}$.
  $\boldsymbol{t}_{i,v,j} \leftarrow \boldsymbol{K}(\boldsymbol{g}_v)\widehat{\mathsf{com}}^{\Sigma}_{i,j}$.
  if $\boldsymbol{\pi}^{\Sigma}_{v,-j} \in \mathbb{Z}^m$, $z^{\Sigma}_{v,-j} \in \mathbb{F}_q$, $\|\boldsymbol{\pi}^{\Sigma}_{v,-j}\|_2 \le tB(v,-j)$, and $\boldsymbol{A}\boldsymbol{\pi}^{\Sigma}_{v,-j} + z^{\Sigma}_{v,-j}\boldsymbol{e}_1 = \boldsymbol{t}_{i,v,j} \pmod q$ **then**
    $\mathcal{S}^{\Sigma}_{i,j} \leftarrow \mathcal{S}^{\Sigma}_{i,j} \cup \{(v, z^{\Sigma}_{v,-j})\}$.
    if $|\mathcal{S}^{\Sigma}_{i,j}| = t^G_c$ **then**
      $\boldsymbol{\beta}^{\Sigma}_{-j} \leftarrow \mathsf{Decode}^G(\mathcal{S}^{\Sigma}_{i,j})$.
      $\widehat{s}^{\Sigma}_j \leftarrow z^{\mathsf{pad},\Sigma}_{i,j+1} - \boldsymbol{b}_0^{\top}\boldsymbol{\beta}^{\Sigma}_{-j}$.
      Output $\widehat{s}^{\Sigma}_j$ and **terminate**.
    end if
  end if

Verification of summed openings. We assume that all AVSS instances being summed use the same CRS, matrix dimensions, component codes, packing vectors, and index conventions. The set dealers is a common, nonempty input to all participants. As in the ordinary verifier,

$$\boldsymbol{K}(\boldsymbol{g}_v) = \sum_{a=0}^{r-1} g_{v,a}\boldsymbol{W}^{-a}, \qquad B(v,-j) = \|\boldsymbol{g}_v\|_1 \|\boldsymbol{a}_{j+1}\|_1 \beta_s.$$

Let $\boldsymbol{M}^{\Sigma} = \sum_{u \in \mathsf{dealers}} \boldsymbol{M}^{(u)}$ denote the sum of the matrices of all the dealers. The honest summed value is $z^{\Sigma}_{v,-j} = \boldsymbol{g}_v^{\top}\boldsymbol{M}^{\Sigma}\boldsymbol{a}_{j+1}$. The sum of the constituent verification equations is

$$\boldsymbol{A}\boldsymbol{\pi}^{\Sigma}_{v,-j} + z^{\Sigma}_{v,-j}\boldsymbol{e}_1 = \boldsymbol{K}(\boldsymbol{g}_v)\sum_{b=0}^{c-1}(\boldsymbol{a}_{j+1})_b \mathsf{com}^{\Sigma}_{i,b} \pmod q.$$

The summed-up proof is hence verified with the same linear equation, but the norm is at most $tB(v, -j)$ by the triangle inequality. Correspondingly, soundness would now require the Power-BASIS assumption for $\boldsymbol{A}_\star$ with a larger solution bound: For a declared upper bound $t_{\max}$ on the number of summed dealers, require

$$2t_{\max}B_{\max} < q, \qquad \beta_{\mathsf{PB}} \geq 2t_{\max}B_{\max}, \qquad |\mathsf{dealers}| \leq t_{\max}.$$

Note that the well-formedness proof included in each commitment is verified and retained; these arguments are neither added nor retransmitted.

The message sent by each party in both of the algorithms above contains one field value and one opening vector, along with the secret index, so its asymptotic size matches that in Algorithm 4. We do not give formal proofs for these protocols since they essentially follow the proofs for LatBingo.

AI acknowledgement We used GPT-5.6 Sol and GPT-6 Astra to improve the editorial content and to push through the analysis of our constructions. We have verified all AI-generated results and take responsibility for their correctness.

References

ABCP23. Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. VSS from distributed ZK proofs and applications. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 405–440, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

ABN+25. Shahla Atapoor, Karim Baghery, Georgio Nicolas, Robi Pedersen, and Jannik Spiessens. Batched and packed (publicly) verifiable secret sharing: A unified framework and applications. Cryptology ePrint Archive, Report 2025/2018, 2025.

ACF+26. Behzad Abdolmaleki, John Clark, Mohammad Foroutani, Shahram Khazaei, and Sajjad Nasirzadeh. A practical framework for lattice-based non-interactive publicly verifiable secret sharing. In *Post-Quantum Cryptography: 17th International Workshop, PQCrypto 2026, Saint-Malo, France, April 14–16, 2026, Proceedings, Part II*, page 107–142, Berlin, Heidelberg, 2026. Springer-Verlag.

ACK21. Thomas Attema, Ronald Cramer, and Lisa Kohl. A compressed Σ-protocol theory for lattices. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 549–579, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

ACL+22. Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 102–132, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

AFLN24. Martin R. Albrecht, Giacomo Fenzi, Oleksandra Lapiha, and Ngoc Khanh Nguyen. SLAP: Succinct lattice-based polynomial commitments from standard assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 90–119, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

AJM+23. Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 39–70, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

ANP23. Benny Applebaum, Oded Nir, and Benny Pinkas. How to recover a secret with $o(n)$ additions. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 236–262, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

Bag25. Karim Baghery. Π: A unified framework for computational verifiable secret sharing. In Tibor Jager and Jiaxin Pan, editors, *PKC 2025, Part IV*, volume 15677 of *LNCS*, pages 110–142, Røros, Norway, May 12–15, 2025. Springer, Cham, Switzerland.

BBB+24. Akhil Bandarupalli, Adithya Bhat, Saurabh Bagchi, Aniket Kate, and Michael K. Reiter. Random beacons in Monte Carlo: Efficient asynchronous random beacon *without* threshold cryptography. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 2621–2635, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

BCG93. Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *25th ACM STOC*, pages 52–61, San Diego, CA, USA, May 16–18, 1993. ACM Press.

BCL26. Renas Bacho, Yanbo Chen, and Julian Loss. Adaptively secure (aggregatable) PVSS from standard assumptions. In *Advances in Cryptology – CRYPTO 2026*, LNCS. Springer, 2026. Full version: Cryptology ePrint Archive, Paper 2026/1100.

BCS21. Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 742–773, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

BCS23. Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Lattice-based succinct arguments for NP with polylogarithmic-time verification. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 227–251, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

BDK12. Michael Backes, Amit Datta, and Aniket Kate. Asynchronous computational VSS with reduced communication complexity. Cryptology ePrint Archive, Report 2012/619, 2012.

BJK+25a. Akhil Bandarupalli, Xiaoyu Ji, Aniket Kate, Chen-Da Liu-Zhang, Daniel Pöllmann, and Yifan Song. Velox: Scalable fair asynchronous MPC from lightweight cryptography. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *ACM CCS 2025*, pages 1799–1813, Taipei, Taiwan, October 13–17, 2025. ACM Press.

BJK+25b. Akhil Bandarupalli, Xiaoyu Ji, Aniket Kate, Chen-Da Liu-Zhang, and Yifan Song. Computationally efficient asynchronous MPC with linear communication and low additive overhead. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part IV*, volume 16003 of *LNCS*, pages 261–294, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

BL23. Renas Bacho and Julian Loss. Adaptively secure (aggregatable) PVSS and application to distributed randomness beacons. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1791–1804, Copenhagen, Denmark, November 26–30, 2023. ACM Press.

BLNS20. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 441–469, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

BNNP25. Dan Boneh, Joachim Neu, Valeria Nikolaenko, and Aditi Partap. Data availability sampling with repair. Cryptology ePrint Archive, Report 2025/1414, 2025.

BS23. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

CCG24. Ignacio Cascudo, Daniele Cozzo, and Emanuele Giunta. Verifiable secret sharing from symmetric key cryptography with improved optimistic complexity. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VII*, volume 15490 of *LNCS*, pages 100–128, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

CD24. Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 216–248, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

CGMA85. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th FOCS*, pages 383–395, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press.

CKLS02. Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 88–97, Washington, DC, USA, November 18–22, 2002. ACM Press.

CKS00. Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. Cryptology ePrint Archive, Report 2000/034, 2000.

CMNW24. Valerio Cini, Giulio Malavolta, Ngoc Khanh Nguyen, and Hoeteck Wee. Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 207–242, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

CR93. Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *25th ACM STOC*, pages 42–51, San Diego, CA, USA, May 16–18, 1993. ACM Press.

DR82. Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. In Robert L. Probert, Michael J. Fischer, and Nicola Santoro, editors, *1st ACM PODC*, pages 132–140, Ottawa, Canada, August 18–20, 1982. ACM.

DXR21. Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2705–2721, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

DXT+25. Sourav Das, Zhuolun Xiang, Alin Tomescu, Alexander Spiegelman, Benny Pinkas, and Ling Ren. Verifiable secret sharing simplified. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 633–651, San Francisco, CA, USA, May 12–15, 2025. IEEE Computer Society Press.

ENP24. Thomas Espitau, Guilhem Niot, and Thomas Prest. Flood and submerse: Distributed key generation and robust threshold signature from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 425–458, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

FMN24. Giacomo Fenzi, Hossein Moghaddas, and Ngoc Khanh Nguyen. Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. *Journal of Cryptology*, 37(3):31, July 2024.

FY92. Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *24th ACM STOC*, pages 699–710, Victoria, BC, Canada, May 4–6, 1992. ACM Press.

GHL22. Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 458–487, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

GJM+21. Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 147–176, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

HASW24. Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. Foundations of data availability sampling. *CiC*, 1(4):34, 2024.

HASW25. Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. Silent threshold encryption with one-shot adaptive security. Cryptology ePrint Archive, Report 2025/1384, 2025.

HSS24. Intak Hwang, Jinyeong Seo, and Yongsoo Song. Concretely efficient lattice-based polynomial commitment from standard assumptions. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 414–448, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

JLS24. Xiaoyu Ji, Junru Li, and Yifan Song. Linear-communication asynchronous complete secret sharing with optimal resilience. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VIII*, volume 14927 of *LNCS*, pages 418–453, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

JS26. Xiaoyu Ji and Yifan Song. Statistically secure asynchronous MPC with linear communication and $\mathcal{O}(n^5)$ additive overhead. Cryptology ePrint Archive, Paper 2026/320, 2026.

KMM+24. Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive VSS using class groups and application to DKG. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 4286–4300, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

KMS+26. Aniket Kate, Pratyay Mukherjee, Hamza Saleem, Pratik Sarkar, and Rohit Sinha. Practical post-quantum secure publicly verifiable secret sharing and applications. Cryptology ePrint Archive, Paper 2026/813, 2026.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors,

CRYPTO 2022, Part II, volume 13508 of *LNCS*, pages 71–101, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

MR04. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

NNT21. Kamilla Nazirkhanova, Joachim Neu, and David Tse. Information dispersal with provable retrievability for rollups. Cryptology ePrint Archive, Report 2021/1544, 2021.

NS22. Ngoc Khanh Nguyen and Gregor Seiler. Practical sublinear proofs for R1CS from lattices. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 133–162, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

NS24. Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 243–275, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

PCR15. Arpita Patra, Ashish Choudhury, and C. Pandu Rangan. Efficient asynchronous verifiable secret sharing and multiparty computation. *Journal of Cryptology*, 28(1):49–109, January 2015.

PNSNA26. Nhat-Minh Pham, Khoa Nguyen, Willy Susilo, and Khuong Nguyen-An. Publicly verifiable secret sharing: Generic constructions and lattice-based instantiations in the standard model. *Computer Standards and Interfaces*, 96:104090, March 2026.

SJK+17. Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris-Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *2017 IEEE Symposium on Security and Privacy*, pages 444–460, San Jose, CA, USA, May 22–26, 2017. IEEE Computer Society Press.

SS24. Victor Shoup and Nigel P. Smart. Lightweight asynchronous verifiable secret sharing with optimal resilience. *Journal of Cryptology*, 37(3):27, July 2024.

Wee25. Hoeteck Wee. Functional commitments and SNARGs for P/poly from SIS. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 617–640, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

WW23. Hoeteck Wee and David J. Wu. Succinct vector, polynomial, and functional commitments from lattices. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 385–416, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

WYZJ24. Hongxiao Wang, Siu-Ming Yiu, Yanmin Zhao, and Zoe Lin Jiang. Updatable, aggregatable, succinct mercurial vector commitment from lattice. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14602 of *LNCS*, pages 3–35, Sydney, NSW, Australia, April 15–17, 2024. Springer, Cham, Switzerland.

XLL+26. Xinxin Xing, Yizhong Liu, Boyang Liao, Jianwei Liu, Bin Hu, Xun Lin, Yuan Lu, and Tianwei Zhang. Gossamer: Lightweight and linear-communication asynchronous (dynamic proactive) secret sharing and the applications. pages 2482–2501, 05 2026.

ZLG+24. Zongyang Zhang, Weihan Li, Yanpei Guo, Kexin Shi, Sherman S. M. Chow, Ximeng Liu, and Jin Dong. Fast RS-IOP multivariate polynomial commitments and verifiable secret sharing. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association.