Laminate: Succinct SIMD-Friendly Verifiable FHE

Kabir Peshawaria, Boston University
Zeyu Liu, Yale University
Ben Fisch, Yale University
Eran Tromer, Boston University

October 21, 2026

Abstract

In outsourcing computation to untrusted servers, one can cryptographically ensure privacy using Fully Homomorphic Encryption (FHE) or integrity using Verifiable Computation (VC) such as SNARK proofs. While each is practical for some applications in isolation, efficiently composing FHE and VC into *Verifiable Computation over Encrypted Data (VCoED)* with verification faster than local computation remains a challenge.

We introduce Laminate, a family of VCoED protocols for adding integrity to computations delegated using BGV/BFV-style FHE. Our approach combines the blind interactive proof framework with the GKR proof system, which avoids computing commitments to intermediate computation states under FHE. Our client-efficient variant employs novel transcript packing and folding techniques that reduce proof size to $O(\log^2 n + N)$ words (suppressing security parameters) for bounded-depth size-n circuits, where N is the BGV/BFV ring dimension. Unlike prior FHE-over-SNARK schemes, Laminate preserves the SIMD parallelism of FHE for both payload circuit evaluation and proof generation; eliminates the need for witness reductions; is field-agnostic; and adds only constant multiplicative depth on top of payload evaluation while performing $\widetilde{O}(n_{\mathsf{FHE}})$ FHE operations, where n_{FHE} is the number of payload FHE operations.

We give an end-to-end concrete integrity analysis, implement the complete protocol, and provide a reproducible methodology for measuring server overhead and per-request client computation. We benchmark Laminate on payload circuits with $n = 2^{34}$ scalar gates. One scheme variant has an overhead of 21–33× compared to FHE evaluation without integrity, with proofs of 52–54 MB. Another variant reduces overhead to 10–14×, with larger proofs. For both, per-request client computation is much faster than local evaluation, demonstrating concrete *outsourced computation* with privacy and integrity, a key distinction from prior VCoED work.

Revision Note (May 2026). Earlier versions of this preprint contained a soundness bug in the transcript packing compiler, which caused us to report smaller proof sizes than what is actually needed. We describe the bug and a concrete attack exploiting it in Section 6.1. This revision corrects the issue by introducing a randomized well-formedness test, at the cost of one ciphertext per test repetition (see Appendix F.1.1). End-to-end multiplicative depth remains unchanged.

Revision Note (October 2026). This revision adds a C++ implementation of Laminate and replaces microbenchmark-based performance estimates with measurements of server time, proof size, and per-request client time (Section 7). We also give an end-to-end concrete integrity analysis covering GKR, polynomial commitments, transcript packing, and Fiat–Shamir (Appendices G and H.2).

Contents

1 Introduction **4**
- 1.1 Challenges in server-efficient outsourceable VCoED . . . 4
- 1.2 Our Contribution . . . 6
- 1.3 Summary of Evaluation . . . 7
- 1.4 Related Work . . . 8

2 Overview **9**
- 2.1 VCoED . . . 9
- 2.2 Construction Overview . . . 9

3 Preliminaries **12**
- 3.1 (Leveled) Homomorphic Encryption . . . 12
- 3.2 Public-Coin Blind Proofs . . . 12
- 3.3 Multilinear Extensions and Sumcheck . . . 13

4 Requisite hIP and HE Schemes **14**
- 4.1 GKR Protocol . . . 14
- 4.2 BGV/BFV Leveled HE Protocol . . . 15
  - 4.2.1 Algebraic Foundations . . . 16
  - 4.2.2 Encryption, Decryption, and Homomorphic Operations . . . 16
  - 4.2.3 Noise Budget and Efficiency . . . 18

5 BhIPs for Wide Payload Circuits **18**
- 5.1 Overview and Prerequisites . . . 18
  - 5.1.1 Class of Supported Payload Circuits . . . 19
  - 5.1.2 Data Layout and Notation . . . 19
  - 5.1.3 The Folding Operator . . . 20
- 5.2 Base and Folded BhIPs . . . 20
  - 5.2.1 Prover Complexity . . . 21

6 Achieving Efficient VCoED **22**
- 6.1 Transcript Packing Compiler . . . 22
- 6.2 The Laminate VCoED Protocols . . . 24

7 Evaluation **27**
- 7.1 Methodology . . . 28
- 7.2 Client Time and Proof Size . . . 29
- 7.3 Server Overhead over FHE Evaluation . . . 30
- 7.4 Comparison with Prior Work . . . 30

8 Open Questions **32**

A Other Approaches to VCoED **40**
- A.1 SNARK-over-FHE . . . 40
- A.2 VOLE-over-FHE . . . 40
- A.3 Composing Garbled Circuits and FHE . . . 41
- A.4 Hidden Challenges . . . 41

B Additional Background on GKR **41**
- B.1 Leveraging Circuit Structure . . . 41

C Affine Coefficients and Openings **42**
D Layer Reductions for Wide Payload Circuits **44**
D.1 Ciphertext and Plaintext Notation . . . 44
D.2 Affine Layer Reduction . . . 45
D.3 Multiplication Layer Reduction . . . 45
D.4 Prover Complexity Analysis . . . 47
E Specialized BhIP for Laned Payload Circuits **50**
E.1 Overview and Prerequisites . . . 50
E.1.1 Payload Representation . . . 50
E.1.2 Dense Alternating Layered Arithmetization . . . 50
E.2 Witness Packing . . . 50
E.2.1 Packing Procedure . . . 51
E.2.2 Data Layout After Packing . . . 51
E.2.3 Witness Packing Cost . . . 51
E.3 Layer Reductions . . . 51
E.3.1 Affine Layer Reduction . . . 52
E.3.2 Multiplication Layer Reduction . . . 54
E.4 Prover Complexity . . . 54
F Transcript Packing **54**
F.1 Protocol and Packing Construction . . . 55
F.1.1 Compiler Description . . . 55
F.1.2 The Packing Argument . . . 56
F.1.3 Packing Layout and Cost . . . 57
F.2 Well-Formedness and Digit Packing . . . 57
F.3 Well-Formedness Tests . . . 59
F.4 Proofs of the Well-Formedness Guarantees . . . 61
F.4.1 Coefficient proof of the packing bound . . . 61
F.4.2 The centered-ternary sampling bound . . . 61
F.4.3 Proof of the BGV guarantee . . . 63
F.4.4 BFV instantiation . . . 64
G Integrity Analysis **65**
G.1 Security Model and PCS Assumptions . . . 65
G.2 Soundness in the Oracle Model . . . 67
G.3 From Oracles to Hash Commitments . . . 71
G.4 Extracting the Packing Witness . . . 72
G.5 Soundness of the Compiled Protocol . . . 74
H Evaluation Methodology Details **75**
H.1 Builds and Timing Boundaries . . . 75
H.2 Integrity Parameters . . . 75
H.3 Asymptotic Comparison with Prior Work . . . 78
H.4 Concrete Estimates for Prior Work . . . 79
H.5 Round Counts and Communication for Laminate . . . 80

1 Introduction

Delegation of expensive computation to cloud services is common nowadays, and raises privacy issues when clients' data is private or sensitive. Fully homomorphic encryption (FHE) [45] is a powerful cryptographic tool for *private* delegation of computation: it enables a server to perform arbitrary computations directly on encrypted data, without learning the underlying plaintext. Over the past decade, a series of breakthroughs [18, 36, 29, 25, 17, 16, 37] improved its efficiency, making FHE increasingly practical for some classes of privacy-sensitive delegation applications, including machine learning [57, 65, 58, 53, 34], secure data processing [44], private information retrieval [3, 67, 39, 82, 51], private set intersection [24, 23, 31], oblivious message retrieval [62, 63, 64, 59, 61], and single secret leader election [15, 77].

FHE alone does not provide *integrity guarantees* for the requested computation. A malicious server can evaluate a circuit of its choice instead of the intended one, e.g. setting the output to a maliciously-chosen value or to a function of secret inputs that leaks information through observable client behavior. To address this, Gennaro et al. [44] introduced the concept of *Verifiable Computation over Encrypted Data (VCoED)*. VCoED enables computation delegation to an untrusted server while (1) preserving data confidentiality and (2) allowing the client to verify the correctness of the computation. In the designated-verifier model studied here, the client keeps its decrypted output and verification decision private.

Beyond privacy and integrity guarantees, a useful VCoED scheme should satisfy *outsourceability*. That is, a VCoED client's total computation after preprocessing takes less time than locally performing the computation itself. Combining this property with efficient server computation has been a central challenge in VCoED. This work presents Laminate, a compiler for adding integrity to native SIMD evaluation using the BGV homomorphic encryption scheme [17]. For the $n = 2^{34}$-gate circuits in our evaluation, the client's computation after preprocessing (input encryption, output decryption, and proof verification) takes less time than evaluating the same circuit locally on one CPU core (Sections 1.3 and 7.1). We also prove asymptotic outsourceability under the conditions of Theorem 6.2.

1.1 Challenges in server-efficient outsourceable VCoED

One approach to realizing outsourceable VCoED is using succinct cryptographic proofs of FHE evaluation. Specifically, SNARKs (Succinct Non-interactive Arguments of Knowledge) allow the server to produce a succinct proof that its output is indeed the result of applying all the designated ciphertext evaluations correctly.[1] For computations using BGV or the related BFV scheme [16, 37], proving *non-algebraic* ciphertext operations such as relinearization incurs high concrete costs.

A recent line of work [43, 5, 40, 80] explores an alternative approach to VCoED: *FHE-over-SNARK*, which flips the order of composition by using FHE to homomorphically compute the SNARK proof in encrypted form. The server homomorphically evaluates the payload circuit (storing ciphertexts for intermediate wires), then homomorphically evaluates a SNARK prover circuit on the resulting wire ciphertexts. The payload trace and homomorphically computed proof messages remain encrypted. A malicious server can still manipulate the payload evaluation, but this is detected when the client decrypts and verifies the proof.

Reducing server overhead and client bandwidth remains a central challenge for these FHE-over-SNARK realizations [5, 40, 80], and faces the following issues.

Witness reduction overheads. SNARK-based verifiable computation often incurs high *witness reduction* costs to transform payload values into a format the SNARK prover can process.

[1] SNARG (Succinct Non-Interactive Argument) suffices, as knowledge-soundness is not needed here.

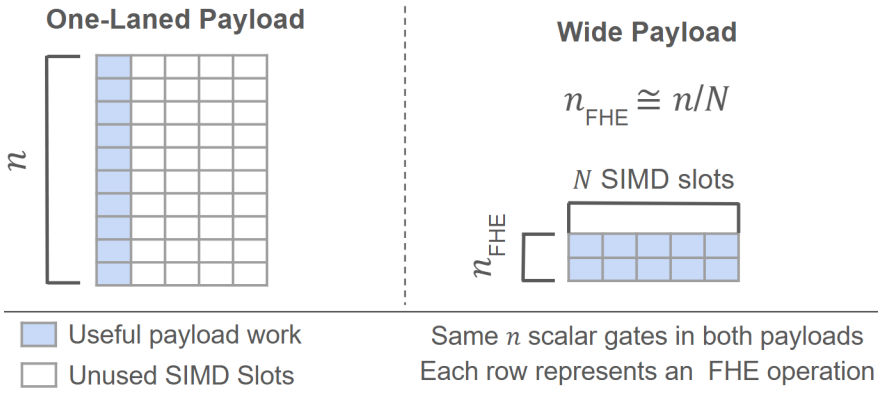


Figure 1: Payload layouts used in our comparison with prior FHE-over-SNARK evaluations [40, 80]. Each row represents one payload FHE operation with N available slots, and blue cells denote useful scalar computation. The single-slot layout requires n FHE operations before proof generation. With full SIMD utilization, the payload instead requires $n_{\mathsf{FHE}} \approx n/N$ FHE operations. For bounded-depth payload circuits, Laminate performs $\widetilde{O}_{\lambda_{\mathsf{int}}}(n_{\mathsf{FHE}})$ FHE operations including proof generation. Asymptotic and concrete comparisons appear in Table 2 and Section 7.4.

For instance, a payload computation consisting of a single $\mathbb{F}_q$ addition would intuitively require a proof for a single addition. However, if the SNARK prover operates over a different field $\mathbb{F}_p$, $x_1 + x_2 \equiv x_3 \pmod{p}$ does *not* imply that $x_1 + x_2 \equiv x_3 \pmod{q}$. Representing the intended arithmetic may require additional witness values and constraints for modular reduction and range checks. Generating this witness under FHE can add work and multiplicative depth, depending on the arithmetization. For example, Phalanx's evaluated instantiation [80] uses a 50-bit base field, whereas BGV/BFV applications often use smaller fields to reduce noise growth [62, 64, 39].

Inefficient proof generation. Server efficiency is further bottlenecked by reliance on polynomial commitments (PCs) for the extended witness, whose size is proportional to the number of gates in the payload circuit. Blind Fractal and Phalanx [40, 80] use NTT-based PCs. The standard butterfly NTT over a length-w vector uses $O(w \log w)$ work and $O(\log w)$ sequential scalar-multiplication layers. Reducing its homomorphic depth by grouping layers increases work. The cited instantiations use this tradeoff to keep the proving depth bounded, performing $w^{1+\Omega(1)}$ work for the polynomial commitment. The extra noise budget for proof generation must also be reserved in advance during payload evaluation, increasing its cost.

Preserving FHE parallelism. Blind Fractal and Phalanx [40, 80] use BGV and BFV, which natively support SIMD parallelism across N slots (typically $N \approx 2^{14}$).[2] Suppose the delegated payload circuit has n gates. A BGV/BFV-based delegation scheme can often evaluate the payload circuit using $n_{\mathsf{FHE}} \ll n$ FHE operations, where each FHE operation is a SIMD instruction that operates on N slots. Crucially, an FHE operation has essentially the same cost whether one slot or all N slots contain useful values. When full SIMD utilization is possible, $n_{\mathsf{FHE}} = n/N$. Indeed, practical applications of BGV/BFV-based delegation schemes (e.g., [62, 39, 24]) rely on this full SIMD utilization to achieve high throughput.

The prior evaluations do not account for constructing their required encrypted witness from a payload trace in an arbitrary native SIMD layout. Their reported prover overheads therefore do not directly measure the cost of adding integrity to a payload that already uses SIMD throughout evaluation. Estimates of prior work assume each scheme already has its required witness (Section 7.4). Figure 1 illustrates the difference between single-slot and fully utilized payload layouts.

Useful delegation requires large payloads. Both works [40, 80] estimate proving costs for

[2]SIMD stands for Single Instruction Multiple Data. These are vectorized operations applied element-wise.

$n = 2^{20}$-size instances from HE microbenchmarks. At this size, the prior-work proof-decryption estimates in Table 9 already exceed scalar evaluation time for our one-lane workload. Their proofs contain $\Omega(\log n)$ ciphertexts, each consisting of $\Omega(N)$ field elements, so proof decryption contributes a substantial client cost. Native SIMD evaluation lets a single payload trace represent many more scalar gates, providing a larger computation over which to amortize verification.

1.2 Our Contribution

We introduce Laminate, a family of VCoED protocols that preserve native SIMD evaluation while adding integrity through the FHE-over-SNARK paradigm. The server homomorphically evaluates the payload circuit, and then uses its intermediate ciphertexts to evaluate a tailored variant of the Goldwasser-Kalai-Rothblum (GKR) proof system [46].[3] We instantiate Laminate over BGV and give the BFV analogue of the packing well-formedness test in Appendix F.4.4. We obtain designated-verifier integrity with negligible soundness error and client privacy in the random oracle model under the assumptions of Theorem 6.2, with no verification oracle and with the client's output and verification decision kept private.

Native SIMD support for payload circuits. Our prover uses the intermediate ciphertexts in the payload's native SIMD layout. Our main protocols, $\mathsf{Laminate}_{\mathsf{base}}$ (optimized for server) and $\mathsf{Laminate}_{\mathsf{fold}}$ (optimized for client) take as input the intermediate values that arise from computing the payload under FHE and assume that all N slots of each intermediate ciphertext have semantic meaning for the n-gate delegated computation. Let n_{FHE} denote the number of FHE operations used to evaluate the payload. Our general-purpose schemes assume that these operations evaluate a plaintext circuit with $n \approx N \cdot n_{\mathsf{FHE}}$ scalar gates (Fig. 1). For example, 2^{20} SIMD-packed arithmetic operations with $N = 2^{14}$ slots represent approximately 2^{34} scalar gates.

Of course, not all payload circuits can fully utilize all N slots. Depending on structure known *a priori*, one can construct VCoED variants that are more server-efficient than general purpose protocols. To compare with prior work, we consider payload circuits with *laned structure*. We say an FHE payload is B-*laned* if B logical lanes independently execute the same scalar arithmetic circuit with no cross-slot interactions (Definition E.1). We give a specialized variant $\mathsf{Laminate}_{\mathsf{laned}}$ based on the laned BhIP construction in Appendix E. We call FHE payloads without such restrictions *wide*, and in particular assume they operate on all N available SIMD slots and may include cross-slot operations such as rotations (Definition 5.1).

Quasi-linear proving without growing multiplicative depth. Our homomorphic prover performs $\widetilde{O}(n_{\mathsf{FHE}})$ FHE operations and adds only a constant number of multiplicative levels beyond the payload evaluation (Theorems 5.4 and 5.5). These properties hold simultaneously, avoiding the tradeoff between work and depth in prior NTT-based instantiations. Thus, increasing the payload size does not require reserving progressively more multiplicative levels for proof generation, which would also increase payload evaluation cost. Our construction combines low-depth sumcheck evaluation with separate ciphertext-index and slot-index reductions, yielding $\mathsf{Laminate}_{\mathsf{base}}$ and $\mathsf{Laminate}_{\mathsf{fold}}$ with different balances of server work and client cost. The prover works over compatible BGV plaintext fields, using extension-field challenges, and *avoids committing to the extended payload witness under FHE.*

Succinct proofs and efficient clients. In our benchmarks, per-request client computation for $\mathsf{Laminate}_{\mathsf{base}}$ and $\mathsf{Laminate}_{\mathsf{fold}}$ takes approximately 1.2–9.1% of local scalar evaluation time (Section 7.1). Further, both schemes admit asymptotically sublinear client verification under the con-

[3] The name Laminate alludes to proof generation forming a thin layer of extra computation, with very low multiplicative depth, on top of the payload computation.

Scheme	Additional Depth	Server FHE Ops	Communication
$\mathsf{Laminate}_{\mathsf{base}}$	1 Level	$O(n_{\mathsf{FHE}} \log n_{\mathsf{FHE}})$	$O(dN \log n_{\mathsf{FHE}} + d \log^2 n)$
$\mathsf{Laminate}_{\mathsf{fold}}$	4.5 Levels	$O(n_{\mathsf{FHE}} \log n + d \log n \log N)$	$O(d \log^2 n + N)$
$\mathsf{Laminate}_{\mathsf{laned}}$	5.5 Levels	$O\big(n_{\mathsf{FHE}}(1 + \frac{B}{N} \log n_{\mathsf{FHE}}) + d \log n \log N\big)$	$O(d \log^2 n + N)$
Phalanx [80]	$\max\{4, 1 + t/2\}$ Levels	$\Omega\big(n + \frac{t}{N} n^{1+1/(2t)}\big)$	$\Omega\big(\sqrt{n} + N \log(n/N)\big)$
Blind Fractal [40]	$\geq 4t + 4$ Levels	$\Omega\big(n/N + t(n/N)^{1+1/t}\big)$	$\Omega\big(N \log(n/N)\big)$

Table 2: Comparison with FHE-over-SNARK schemes for an n-gate, B-laned payload (Definition E.1) with ciphertext-ciphertext depth $d_\times$, $d = 2d_\times + 1$, and $n_{\mathsf{FHE}} = n/B$ payload FHE operations. We assume $n \geq N$. Depth follows the convention of Section 4.2 for all schemes. Communication counts field elements, ciphertext coefficients, and hash words, omitting their bit lengths and fixed factors R, D. The Laminate bounds include all proof components, assuming polynomial openings with $O(\log^2 m)$ communication for m coefficients. Prior-work bounds use the ciphertext layouts from the respective papers, with t denoting the number of sequential multiplication stages per NTT (Appendix H.3).

ditions of Theorem 6.2. $\mathsf{Laminate}_{\mathsf{fold}}$ compresses the committed transcript while checking that malicious ciphertext noise cannot alter the reconstructed plaintext messages. A detailed comparison of runtime and communication asymptotics is available in Table 2.

End-to-end concrete integrity analysis. We analyze the complete noninteractive VCoED protocol in the random oracle model, accounting jointly for GKR, polynomial commitments, ciphertext well-formedness, transcript packing, and Fiat–Shamir (Appendix G). For the parameters used in our benchmarks, Corollary H.1 bounds the probability that a classical malicious server makes the client accept an incorrect result below 2^{-50}. The bound uses a budget of $q_{\mathsf{mv}} = 2^{60}$ fresh protocol challenges across all attempted transcripts and final verification, together with the hash assumptions, separate computational budgets, and packed-variant parameter conditions in Appendix H.2.

Complete implementation and reproducible evaluation. We implement setup and the full client and server computations for all three Laminate variants. Our evaluation accounts for the noise budget needed for proving, server storage and serialization, and client encryption, decryption, and verification (Section 7.1). The public artifact's benchmarking guide provides parameter-selection procedures, baselines, and cost-accounting checks that support comparable evaluations of other VCoED constructions.

1.3 Summary of Evaluation

We benchmark Laminate on two synthetic payload circuits (balancedLaned and mulHeavyPyramid) with $n = 2^{34}$ scalar gates, using $n_{\mathsf{FHE}} = 2^{20}$ BGV operations over $N = 2^{14}$ SIMD slots. The circuits differ in both the fraction of multiplication gates and their placement. Here, client time means per-request computation after preprocessing: input encryption, output decryption, and proof verification. A single-core local evaluator takes 61.94 s and 132.00 s, while $\mathsf{Laminate}_{\mathsf{base}}$ client time is 5.660 s and 5.612 s. $\mathsf{Laminate}_{\mathsf{fold}}$ further reduces client time to 1.663 s and 1.597 s, respectively. These client times are approximately 1.2–9.1% of the times it would take to locally evaluate the payload.

For $\mathsf{Laminate}_{\mathsf{base}}$, complete server time is 13.6× the FHE evaluation time without integrity on balancedLaned and 10.0× on mulHeavyPyramid. For $\mathsf{Laminate}_{\mathsf{fold}}$, the overhead is 32.9× on balancedLaned and 20.7× on mulHeavyPyramid. Across both payloads, $\mathsf{Laminate}_{\mathsf{fold}}$ spends 2.1–2.4 times as much server time as $\mathsf{Laminate}_{\mathsf{base}}$ to obtain 21.5–21.6 times smaller proofs and 3.4–3.5 times lower client time. Its proofs are 54.1 MB and 52.1 MB, respectively. The Laminate timings are single-threaded wall-clock measurements, and server totals include payload evaluation, proof

generation, and the associated storage and serialization. Section 7 gives the methodology and cost breakdown.

We compare with estimates for selected prover stages in Phalanx [80] and Blind Fractal [40], assuming their required encrypted witnesses are already available (Section 1.1). These estimates for prior work omit payload evaluation, witness reduction, any required parameter enlargement, and all storage costs. At $n = 2^{34}$, $\mathsf{Laminate}_{\mathsf{laned}}$'s complete server time is less than 1/86 of Phalanx's matrix-product estimate and 1/4800 of the counted Blind Fractal stages. Section 7.4 continues the comparison in more detail.

1.4 Related Work

Our focus is outsourcing computation from a weak client to a powerful server, with both sublinear client work and a concretely efficient server. Several approaches to VCoED provide one of these benefits at the expense of the other.

FHE-over-SNARK. The FHE-over-SNARK paradigm was introduced by Garg et al. [43], with subsequent constructions based on FRI, Fractal, and Spartan [5, 40, 80]. These are the closest prior constructions, with their evaluated witness layouts and server and client costs discussed in Section 1.1. Blind Fractal [40] also studies private delegation of zero-knowledge proof generation, where the client's savings are relative to generating a proof locally. That task is substantially more expensive than evaluating the underlying plaintext computation, so savings in proof delegation do not establish outsourceability for the computation itself. Phalanx [80] also considers an extension to maliciously secure two-party computation, where the server may have private inputs that prevent the client from evaluating the computation locally. We treat this as out of scope. Our comparison concerns the use of these constructions to outsource the client's computation.

Follow-up work to Laminate includes Lasagne [81], which uses GKR to avoid homomorphic witness commitments and packs each round's messages into a single ciphertext, as in the first stage of our transcript packing construction (Appendix F.1.3). Like our Base BhIP prover, it stops sumchecks early and lets the verifier complete the remaining computation locally. Their GKR treatment uses higher-degree polynomial encodings and supports SIMD-aware proof generation. They measure an end-to-end encrypted face-matching application and project costs for larger circuits from microbenchmarks. Our transcript packing additionally combines committed rounds and checks that ciphertext arithmetic preserves their plaintext messages. In another follow-up, Huang et al. [52] propose a Spartan-based VCoED scheme using a linear-work, constant-depth PCS based on RAA codes, giving a constant-depth quasilinear blind SNARK. Their evaluation starts from an already packed encrypted R1CS witness, leaving the cost of obtaining this witness from the client's encrypted inputs outside the measured pipeline (Section 1.1).

Recent work by Kang, Pohle, and Spiessens [54] addresses one-bit leakage under additional assumptions and for sufficiently large plaintext fields by running the blind-proof protocol on $D+1$ independently encrypted shares, where D is the payload polynomial's total degree. This increases server and verification work by a factor of $D+1$, which is attractive for low-degree payloads but can grow exponentially with multiplicative depth.

Other approaches. SNARK-over-FHE [38, 14, 42, 6] provides succinct verification, but proving non-algebraic ciphertext operations such as relinearization is expensive for the server [56]. VOLE-over-FHE, as in ZHE [83], improves server efficiency but requires linear client work. Composing garbled circuits and FHE [44] requires the server to homomorphically evaluate symmetric decryption for every Boolean payload gate. Constructions based on hidden challenges or MACs [30, 4] incur substantial server overhead from homomorphic evaluation of cryptographic computation. We discuss these approaches and their limitations for our outsourcing setting in Appendix A.

2 Overview

2.1 VCoED

Verifiable Computation over Encrypted Data (VCoED) is a cryptographic primitive that allows a client to outsource computation to a server that is untrusted in terms of both privacy and integrity.

Threat model. A server (assumed to be computationally bounded) should learn nothing about the client's input to the computation (*client privacy*). Moreover, the server should be unable to convince the client of an incorrect output (*soundness*). We consider designated verification without a verification oracle: the client does not reveal its decrypted output and acceptance decision (see Remark 3.1 about possible leakage from revealing verification decisions). This is captured by the following definition, adapted from [44, 80].

Definition 2.1 (Verifiable Computation over Encrypted Data [44, 80])**.** A *Verifiable Computation over Encrypted Data (VCoED)* scheme consists of four polynomial time algorithms:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^{\lambda_{\mathsf{int}}}, 1^{\lambda_{\mathsf{priv}}}, \mathsf{Ckt})$: given circuit Ckt and the integrity and privacy parameters, the setup algorithm outputs a preprocessed key pair.
- $\mathsf{Enc}_{\mathsf{in}} \leftarrow \mathsf{ProbGen}(\mathsf{pk}, \mathsf{in})$: an encryption algorithm takes input data in and public key pk and outputs an encryption $\mathsf{Enc}_{\mathsf{in}}$.
- $(\mathsf{Enc}_{\mathsf{out}}, \pi) \leftarrow \mathsf{Compute}(\mathsf{pk}, \mathsf{Enc}_{\mathsf{in}}, \mathsf{Ckt})$: given public key pk, encrypted input $\mathsf{Enc}_{\mathsf{in}}$, and circuit Ckt, the algorithm outputs an encrypted result $\mathsf{Enc}_{\mathsf{out}}$ and a proof π.
- $\mathsf{out}/\perp \leftarrow \mathsf{Verify}(\mathsf{sk}, \mathsf{Enc}_{\mathsf{out}}, \pi)$: given secret key sk, encrypted result $\mathsf{Enc}_{\mathsf{out}}$, and proof π, the verification algorithm should return out if $\mathsf{out} = \mathsf{Ckt}(\mathsf{in})$ and $\perp$ otherwise.

We suppress the client's per-request state in the notation for Verify. This includes the input and its encryption, retained by $\mathsf{ProbGen}$ for verification.

A VCoED scheme must satisfy the following four properties.

- **Completeness**: An honest server will pass the verification of an honest client.
- **Outsourceability**: The proof size and client time are asymptotically sublinear in the time required to evaluate the circuit.
- **Client Privacy**: A malicious computationally-bounded server should learn no information about the client's chosen input.
- **Soundness**: A malicious computationally-bounded server cannot produce a proof that makes an honest client output an incorrect $\mathsf{out}' \neq \mathsf{Ckt}(\mathsf{in})$ with non-negligible probability.

2.2 Construction Overview

Blind interactive proofs. We start with the *Blind (holographic) Interactive Proof* (BhIP) framework from [40]. In a public-coin interactive proof, the prover sends messages to a verifier who replies with random challenges. To obtain a blind proof, the server homomorphically computes the prover's messages, and the client decrypts them to perform the verifier's checks. A key observation in [43, 5, 40] is that interaction can be removed without homomorphically evaluating the entire non-interactive prover. Fiat-Shamir challenges are computed by hashing the ciphertext transcript

outside FHE. Each ciphertext fixes its decrypted message before the next public challenge. This preserves the order needed by the interactive soundness proof.

The verifier's challenges can therefore be treated as unencrypted *scalar inputs*, free of noise. This allows some of the prover's circuit to be evaluated via cheap scalar-ciphertext operations, reducing both runtime and noise consumption. Even though the IP prover executes multiple rounds, each logically dependent on the previous ones, their depths need not add up: our prover recomputes each message from the original wire ciphertexts or a fixed checkpoint and the public challenges, without chaining the noise growth of earlier encrypted messages. Thus, our task reduces to designing a public-coin interactive prover whose messages can be computed efficiently under FHE.

Core insight: GKR is the right proof system. The GKR public-coin IP is well-suited for homomorphic evaluation. GKR avoids the homomorphic witness commitments responsible for the work-depth tradeoff discussed in Section 1.1.[4] GKR [46] proves correctness of a layered arithmetic circuit by successively reducing a claim about its output layer to one about its input layer that is checked directly by the verifier. Each layer reduction uses sumcheck to relate the claimed layer values to those in the preceding layer. The prover evaluates multilinear polynomials by taking linear combinations of encrypted wire values with public scalar coefficients. Its remaining task is to compute sumcheck round polynomials efficiently under FHE.

Quasi-linear proving with constant multiplicative depth. The classic, linear-time sumcheck algorithm [33, 72] reuses work from previous rounds. This dependency gives it an $\mathcal{O}(\mu)$ multiplicative depth (where μ is the number of rounds), which is a super-constant depth we wish to avoid under FHE. Instead, we use the quasi-linear time sumcheck prover of [32], which computes each round polynomial from scratch, admitting a *constant multiplicative depth circuit*. In our setting, each ciphertext-index message is computed directly from the relevant layer's encrypted wire values and the public challenges sampled so far. Figure 3 illustrates why these sequential rounds do not form a chain of increasing multiplicative depth. For the Folded prover's slot-index rounds, we first fix the ciphertext-index challenges in these wire values and use the resulting ciphertexts as a fixed checkpoint.

The use of low-depth sumcheck evaluation for Spartan was already observed in Phalanx [80]. The work and depth bounds for the two variants are given in Theorems 5.4 and 5.5.

Tailoring GKR to wide FHE payloads. To apply GKR to a wide payload, we index each wire by the ciphertext containing it and its slot within that ciphertext. For example, a layer represented by S ciphertexts contains SN scalar wire values. The sumcheck variables therefore divide into $\log S$ ciphertext-index variables and $\log N$ slot-index variables. During the ciphertext-index rounds, the prover performs the same arithmetic across all N slots in parallel. The intermediate ciphertexts from payload evaluation supply these values in the required layout.

Wide payloads may also contain operations that mix slots, such as *slot automorphisms* (sometimes referred to as slot permutations or rotations). We represent the payload using alternating layers of SIMD multiplication gates and affine gates, where the affine gates describe additions, multiplication by public plaintexts, and slot automorphisms. The layer reductions use this public structure to relate output slots to their source slots. In particular, SIMD evaluation of the prover does not require the payload slots to execute independent computations. A sparse description of the circuit's affine operations is committed during preprocessing and checked through polynomial openings outside FHE. These openings let the client check the circuit-dependent quantities in each

[4] One could consider using a non-NTT based PCS, such as Ligero [2] or Brakedown [47] with a linear code that admits fast, low-multiplicative depth encoding (e.g. RAA codes). However, this retains square-root communication in the witness length.

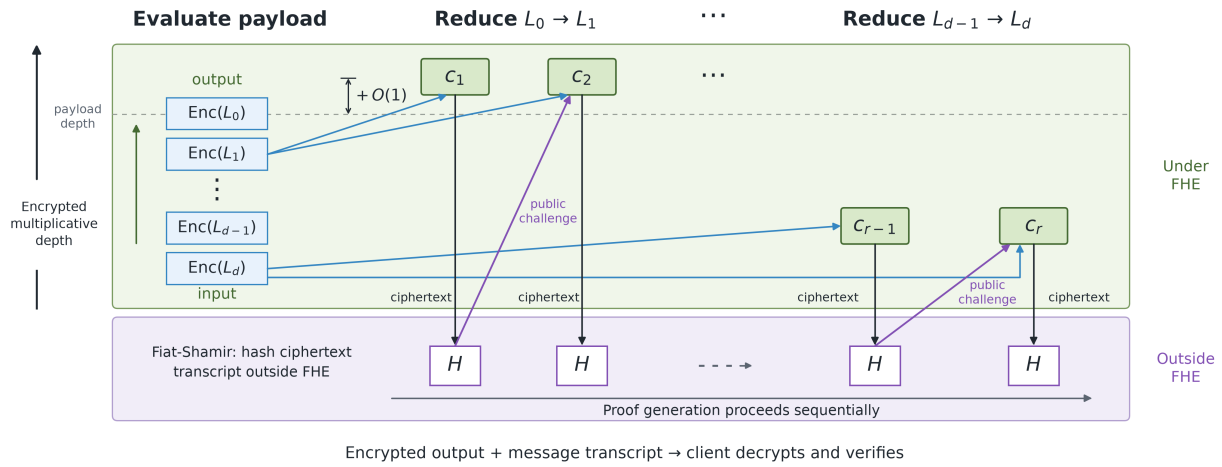


Figure 3: Blind GKR with Fiat–Shamir for the Base variant. Packed variants apply the transcript packing technique in Section 6.1 before Fiat–Shamir. Payload evaluation proceeds from input layer L_d to output layer L_0, while GKR reduces claims in the opposite direction. Each encrypted message c_j is computed from saved wire ciphertexts using public challenges obtained by hashing the ciphertext transcript outside FHE. Successive rounds therefore do not accumulate multiplicative depth.

layer reduction without scanning the public circuit. We describe the arithmetization and the corresponding layer reductions in Section 5.

Base and folded proofs. The distinction between ciphertext indices and slot indices gives a choice about where to complete each layer reduction. Our *base* prover stops after the ciphertext-index sumcheck rounds and sends ciphertexts containing the separate slot contributions. The client decrypts these ciphertexts and performs the remaining aggregation locally. This reduces the server's homomorphic work and noise consumption, at the cost of communicating all N slot contributions and processing them at the client. Applying Fiat-Shamir directly to this BhIP gives $\mathsf{Laminate}_{\mathsf{base}}$.

Our *folded* prover performs the slot-index rounds as well, and homomorphically aggregates the slot contributions. Each resulting message ciphertext contains one field element replicated across its N slots. Compiling the folded BhIP with transcript packing and Fiat-Shamir gives $\mathsf{Laminate}_{\mathsf{fold}}$.

Client proof sizes. For fixed FHE parameters and field-extension degree, the $\mathsf{Laminate}_{\mathsf{base}}$ encrypted transcript contains $O(d \log n_{\mathsf{FHE}})$ ciphertexts, where $d = 2d_{\times} + 1$. The total proof size, including public coefficient openings, is given in Table 2. For a wide payload with $n \approx N n_{\mathsf{FHE}}$ scalar gates and bounded multiplicative depth, this is already asymptotically sublinear in n. Under the conditions of Theorem 6.2, $\mathsf{Laminate}_{\mathsf{base}}$ therefore admits outsourceability. However, we would like to further reduce the client's communication and decryption costs.

Transcript packing. Each folded message ciphertext repeats the same value across all N slots. A natural approach is to pack these values densely into shared ciphertexts and send them at the end of the protocol. The prover selects each message's destination slot using a plaintext mask and sums the masked ciphertexts. However, sending only this final packed transcript breaks the temporal ordering required for soundness. In every round, the prover must fix its message before receiving the next challenge. If the verifier sends its challenges without first receiving anything that binds the prover to these messages, the prover can choose earlier messages as a function of later challenges.

Proof of packing. Our solution is to have the prover commit to its messages during the interaction and give a *proof of packing* at the end. In each round, the prover aligns its messages into assigned slots of a single ciphertext and sends a hash commitment to that ciphertext before receiving the

next challenge. After GKR and its public gate openings, interactive Spartan proves that the packed ciphertexts are the prescribed masked sums of ciphertexts opening these commitments. We apply Fiat–Shamir once to the whole interaction, including Spartan's polynomial openings. This proof is computed outside FHE, since the ciphertext coefficients are known to the server.

Well-formedness of the committed ciphertexts. The proof of packing constrains ciphertext arithmetic, but does not ensure that the committed ciphertexts have sufficiently small noise. Excessive noise can cause the plaintext masks to change the intended messages upon decryption, even when the packing argument accepts. We therefore test random linear combinations of the committed ciphertexts, sampled after all round commitments are fixed. The packing argument ties these test ciphertexts to the committed list, and the client checks their noise using its secret key. Low-noise ciphertexts pass with high probability. Under the protocol's parameter bounds, a fixed committed list whose prescribed packing changes a plaintext message passes all T tests with probability at most 2^{-T}.

Cost of transcript packing. Each test adds one ciphertext, so choosing $T = \Theta(\lambda_{\mathsf{int}})$ contributes $O(\lambda_{\mathsf{int}} N)$ ciphertext coefficients. With succinct polynomial openings, the total proof size of $\mathsf{Laminate}_{\mathsf{fold}}$ is $O_{\lambda_{\mathsf{int}}}(d \log^2 n + N)$, including the public coefficient openings and packing argument (Table 2). The packing argument is an instance of the SNARK-over-FHE approach discussed in Section 1.1, but the FHE computation being proved is *linear* and does not require non-algebraic ciphertext operations that make this approach expensive for the prover. The compiler's guarantees are given in Theorem 6.1, the construction in Appendix F.1.1, and soundness analysis in Appendix G.

3 Preliminaries

3.1 (Leveled) Homomorphic Encryption

A (leveled) homomorphic encryption (HE) scheme provides four algorithms: $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}$. The Eval algorithm allows for public computation on encrypted data. Given a function f and ciphertexts $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, m_i)$, it computes $\mathsf{ct}' \leftarrow \mathsf{Eval}(\mathsf{evk}, f, \{\mathsf{ct}_i\})$ such that $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}') = f(m_1, \ldots, m_w)$. We use leveled HE, which supports circuits of fixed multiplicative depth without bootstrapping.

Field notation. Let $\mathbb{F}_t$ be the plaintext field and let $\mathbb{K} = \mathbb{F}_{t^R}$ be its degree-R extension. After fixing an $\mathbb{F}_t$-basis, an element of $\mathbb{K}$ is represented by R elements of $\mathbb{F}_t$, and an extended ciphertext consists of R ciphertexts encrypting these coordinates. Additions and multiplications involving extended plaintexts or ciphertexts are interpreted over $\mathbb{K}$. For GKR, we take $q = t$, so $\mathbb{F}_q = \mathbb{F}_t$ and $\mathbb{K} = \mathbb{F}_{q^R}$.

3.2 Public-Coin Blind Proofs

Indexed languages and public-coin proofs. We consider indexed languages $\mathcal{L} = \{\mathcal{L}_i\}$, where an index i, such as a circuit, defines a language $\mathcal{L}_i$. In a preprocessing proof system, the index is processed offline so that the verifier's online work can be sublinear in $|i|$. Our work builds on public-coin holographic interactive proofs (pc-hIPs) for indexed languages [7, 27]. In each round, the prover sends a message and the verifier responds with a uniformly random challenge. The verifier decides whether to accept after inspecting the transcript.

State-restoration soundness. Round-by-round (RBR) soundness bounds the probability that a false claim becomes true at a fresh verifier challenge by ϵ_{rbr} [28]. A Fiat–Shamir prover can try several messages to obtain a favorable challenge. State-restoration soundness measures its success when it can explore several continuations of earlier transcripts, with consistent answers to repeated

queries [10]. Write $\varepsilon^{\mathrm{sr}}(q_{\mathsf{mv}}, n)$ for its maximum probability of producing an accepting transcript for a false statement of size n, with at most q_{mv} fresh challenges, including final verification. RBR soundness gives $\varepsilon^{\mathrm{sr}}(q_{\mathsf{mv}}, n) \leq q_{\mathsf{mv}}\epsilon_{\mathsf{rbr}}$ by a union bound. Appendix G.1 gives the formal definitions and query conventions.

Blind holographic interactive proofs. Given a pc-hIP Π for an indexed language $\mathcal{L}$ and an HE scheme $\mathcal{E}$, a *public-coin blind holographic interactive proof (BhIP)* $\mathcal{E}[\Pi]$ is an interactive proof system for $\mathcal{L}$ in which the encrypted components of each prover message are computed by homomorphically evaluating the plaintext prover's next-message function. The prover may also send public auxiliary messages. The designated verifier decrypts the encrypted message components and then runs the plaintext verifier's decision logic on the resulting transcript. This adapts the definition from [40] to the public-coin setting. An accepting transcript for the blind protocol decrypts to an accepting transcript for the plaintext protocol with the same public challenges. For a fixed secret key, each ciphertext fixes its plaintext before the next challenge, so the plaintext RBR analysis applies.

Fiat–Shamir compilation. Applying Fiat–Shamir to a pc-BhIP gives a blind non-interactive argument in the random oracle model. For classical adversaries, its soundness error is at most $\varepsilon^{\mathrm{sr}}(q_{\mathsf{mv}}, n) + O(q_{\mathsf{FS}}^2/2^{\ell_{\mathsf{FS}}})$, where q_{FS} counts queries to the ℓ_{FS}-bit transcript hash, including verification [10, 28].

Remark 3.1 (One-bit leakage)**.** As noted in prior work [5, 40, 80], a malicious prover that learns whether its blind proof passed verification can learn one bit of information about the encrypted input. SNARK-over-FHE avoids it by proving statements about ciphertexts directly. Recent work [54] also addresses this leakage within the blind-proof approach under additional assumptions, with overhead depending on the payload's total degree (see Section 1.4).

3.3 Multilinear Extensions and Sumcheck

Multilinear extensions. A function $h : \{0,1\}^\mu \to \mathbb{F}$ has a unique multilinear extension (MLE) $\widetilde{h} : \mathbb{F}^\mu \to \mathbb{F}$, which is linear in each variable and agrees with h on the Boolean hypercube. The values $(h(\vec{v}))_{\vec{v}\in\{0,1\}^\mu}$ are the Lagrange coefficients of $\widetilde{h}$. When $\mathbb{K}$ is an extension of $\mathbb{F}$, we also evaluate $\widetilde{h}$ at points in $\mathbb{K}^\mu$.

The multilinear extension of the Boolean equality function is $\widetilde{\mathsf{eq}}_\mu(\vec{X}, \vec{Y}) = \prod_{i=1}^{\mu}\big(X_iY_i + (1 - X_i)(1 - Y_i)\big)$. On Boolean inputs, it equals 1 when $\vec{X} = \vec{Y}$ and 0 otherwise. At any point $\vec{X}$, the value $\widetilde{h}(\vec{X})$ is the inner product of the Lagrange coefficients with $(\widetilde{\mathsf{eq}}_\mu(\vec{X}, \vec{v}))_{\vec{v}\in\{0,1\}^\mu}$.

Multivariate sumcheck. We recall the multivariate sumcheck protocol [66]. Let $\mathbb{K}$ be a finite field whose size is chosen so that the sumcheck error meets the integrity target λ_{int}. The prover holds a μ-variate polynomial $f \in \mathbb{K}[X_0, \ldots, X_{\mu-1}]$ of individual degree at most d and wants to prove $\sum_{\vec{x}\in\{0,1\}^\mu} f(\vec{x}) = s_0$ for a claimed value $s_0 \in \mathbb{K}$. For each $i \in \{0, \ldots, \mu-1\}$, after receiving challenges $\gamma_0, \ldots, \gamma_{i-1} \in \mathbb{K}$, the prover sends a polynomial $R_i(X) \in \mathbb{K}[X]$ of degree at most d claiming that $R_i(X) = \sum_{\vec{x}\in\{0,1\}^{\mu-i-1}} f(\gamma_0, \ldots, \gamma_{i-1}, X, \vec{x})$, where the challenge prefix is empty when $i = 0$. The verifier checks $R_i(0) + R_i(1) = s_i$, samples $\gamma_i \xleftarrow{\$} \mathbb{K}$, and sets $s_{i+1} := R_i(\gamma_i)$. After μ rounds, the verifier checks $s_\mu = f(\gamma_0, \ldots, \gamma_{\mu-1})$ directly. The protocol is perfectly complete. If the initial claim is false, a false claim can become true in a round only when two distinct degree-d univariate polynomials agree at the verifier's random challenge. By the Schwartz–Zippel lemma and a union bound, the verifier accepts a false claim with probability at most $d\mu/|\mathbb{K}|$. This also proves that sumcheck has RBR error at most $d/|\mathbb{K}|$.

Remark 3.2 (Omitting one round polynomial evaluation)**.** A degree-d round polynomial is specified by $d + 1$ distinct evaluations, including $R(0)$ and $R(1)$. Since the verifier checks $R(0) + R(1) = s$,

the prover can omit $R(1)$, which the verifier recovers as $s - R(0)$ before interpolation. Thus d evaluations suffice.

4 Requisite hIP and HE Schemes

4.1 GKR Protocol

We recall the GKR protocol [46, 32, 73]. GKR proves that a layered arithmetic circuit maps a stated input to a stated output. Its main step reduces one claim about a layer to one claim about the layer below it. Circuit-dependent information can be prepared in an offline indexing phase.

Let Ckt be a depth-d layered fan-in-two arithmetic circuit over $\mathbb{F}_q$. We write L_0 for the output layer, L_d for the input layer, and L_i for the wires in layer i. Every wire in L_i, for $i \in \{0, \ldots, d-1\}$, is the output of an ADD or MUL gate whose inputs are wires in L_{i+1}. We pad each layer to a power of two and let $S_i := |L_i|$ denote its padded size and $s_i := \log_2 S_i$. Padded wires have value zero, and both gate indicators defined below are zero on padded gate indices. We index the wires in L_i by $\{0,1\}^{s_i}$. The wire values define a table $L_i : \{0,1\}^{s_i} \to \mathbb{F}_q$ whose multilinear extension is $\widetilde{L}_i : \mathbb{K}^{s_i} \to \mathbb{K}$.

Gate indicators. For every $i \in \{0, \ldots, d-1\}$, define

$$\mathsf{add}_i, \mathsf{mul}_i : \{0,1\}^{s_i} \times \{0,1\}^{s_{i+1}} \times \{0,1\}^{s_{i+1}} \to \{0,1\}.$$

For $\vec{g} \in \{0,1\}^{s_i}$ and $\vec{x}, \vec{y} \in \{0,1\}^{s_{i+1}}$, $\mathsf{add}_i(\vec{g}, \vec{x}, \vec{y}) = 1$ exactly when wire $\vec{g}$ is the output of an ADD gate with inputs $\vec{x}$ and $\vec{y}$. The function mul_i is defined analogously for MUL gates. We write $\tilde{\mathsf{add}}_i$ and $\tilde{\mathsf{mul}}_i$ for their multilinear extensions over $\mathbb{K}$.

For $\vec{r} \in \mathbb{K}^{s_i}$ and $\vec{x}, \vec{y} \in \mathbb{K}^{s_{i+1}}$, define

$$\begin{aligned} G_i(\vec{r}, \vec{x}, \vec{y}) := {} & \tilde{\mathsf{add}}_i(\vec{r}, \vec{x}, \vec{y})\big(\widetilde{L}_{i+1}(\vec{x}) + \widetilde{L}_{i+1}(\vec{y})\big) \\ & + \tilde{\mathsf{mul}}_i(\vec{r}, \vec{x}, \vec{y})\widetilde{L}_{i+1}(\vec{x})\widetilde{L}_{i+1}(\vec{y}). \end{aligned}$$

The gate indicators give the standard GKR layer identity as follows:

$$\widetilde{L}_i(\vec{r}) = \sum_{\vec{x} \in \{0,1\}^{s_{i+1}}} \sum_{\vec{y} \in \{0,1\}^{s_{i+1}}} G_i(\vec{r}, \vec{x}, \vec{y}). \tag{1}$$

When $\vec{r}$ is a Boolean gate index, the indicators select the two inputs to that gate and apply its operation. The identity holds for every $\vec{r} \in \mathbb{K}^{s_i}$ because both sides are multilinear in $\vec{r}$ and agree on the Boolean hypercube.

Output randomization. The prover begins by claiming that the values in L_0 equal the stated circuit output out. The verifier samples $\vec{r}_0 \xleftarrow{\$} \mathbb{K}^{s_0}$ and reduces this table equality to

$$\widetilde{L}_0(\vec{r}_0) \stackrel{?}{=} \widetilde{\mathsf{out}}(\vec{r}_0).$$

The rest of GKR reduces this claim through the circuit one layer at a time.

Layer reduction. Suppose the verifier holds a claim

$$\widetilde{L}_i(\vec{r}_a) + \alpha \widetilde{L}_i(\vec{r}_b) \stackrel{?}{=} v_i, \tag{2}$$

where $\vec{r}_a, \vec{r}_b \in \mathbb{K}^{s_i}$ and $\alpha, v_i \in \mathbb{K}$. The first reduction is unbatched. It fits this form by taking $\vec{r}_a = \vec{r}_0$, $\alpha = 0$, and any $\vec{r}_b \in \mathbb{K}^{s_0}$. Using Eq. (1), the claim becomes

$$v_i \stackrel{?}{=} \sum_{\vec{x}\in\{0,1\}^{s_{i+1}}} \sum_{\vec{y}\in\{0,1\}^{s_{i+1}}} \big(G_i(\vec{r}_a, \vec{x}, \vec{y}) + \alpha G_i(\vec{r}_b, \vec{x}, \vec{y})\big).$$

This polynomial has individual degree at most two in the coordinates of $(\vec{x}, \vec{y})$. The parties apply sumcheck over these $2s_{i+1}$ variables. At the end of sumcheck, the verifier has sampled $\vec{\rho}, \vec{\phi} \in \mathbb{K}^{s_{i+1}}$. The remaining claim is

$$G_i(\vec{r}_a, \vec{\rho}, \vec{\phi}) + \alpha G_i(\vec{r}_b, \vec{\rho}, \vec{\phi}) \stackrel{?}{=} v_{\mathsf{sum}}.$$

The prover sends $u, w \in \mathbb{K}$, claiming

$$\widetilde{L}_{i+1}(\vec{\rho}) \stackrel{?}{=} u, \qquad \widetilde{L}_{i+1}(\vec{\phi}) \stackrel{?}{=} w.$$

The verifier evaluates or verifies the required gate-indicator values and forms

$$a_i := \tilde{\mathsf{add}}_i(\vec{r}_a, \vec{\rho}, \vec{\phi}) + \alpha \tilde{\mathsf{add}}_i(\vec{r}_b, \vec{\rho}, \vec{\phi}),$$
$$m_i := \tilde{\mathsf{mul}}_i(\vec{r}_a, \vec{\rho}, \vec{\phi}) + \alpha \tilde{\mathsf{mul}}_i(\vec{r}_b, \vec{\rho}, \vec{\phi}).$$

It then checks

$$v_{\mathsf{sum}} \stackrel{?}{=} a_i(u + w) + m_i uw. \tag{3}$$

If the check passes, the two remaining claims concern $\widetilde{L}_{i+1}(\vec{\rho})$ and $\widetilde{L}_{i+1}(\vec{\phi})$. The verifier samples $\beta \xleftarrow{\$} \mathbb{K}$ and batches them as

$$\widetilde{L}_{i+1}(\vec{\rho}) + \beta \widetilde{L}_{i+1}(\vec{\phi}) \stackrel{?}{=} u + \beta w.$$

If either evaluation claim is false, this equality holds for at most one value of β. The batching step therefore adds soundness error at most $1/|\mathbb{K}|$. This is a claim of the form in Eq. (2) for the next layer. After the last reduction, the verifier evaluates $\widetilde{L}_d$ from the input table and checks the final claim directly.

We discuss how circuit structure can improve GKR and explain efficient evaluation of gate polynomials in Appendix B.

Remark 4.1 (Layered Circuits vs. General Circuits)**.** The layer identity above assumes that every gate in L_i takes both inputs from L_{i+1}. A general arithmetic circuit may instead contain gates whose inputs lie in nonadjacent lower layers. Such a circuit can be made layered by inserting copy gates, although this conversion may increase its size and depth. Zhang et al. [79] give a GKR-style protocol that handles general arithmetic circuits directly. Their gate indicators record the input layer as well as the input wire within that layer. The resulting layer reduction still uses sumcheck, with a more involved layer polynomial and cost analysis [79].

4.2 BGV/BFV Leveled HE Protocol

BGV [17] and BFV [16, 37] are leveled homomorphic encryption schemes. We instantiate Laminate over BGV and give the BFV analogue of the packing well-formedness test in Appendix F.4.4. BGV's IND-CPA security is based on the Ring Learning with Errors (RLWE) assumption [17]. Throughout, we assume IND-CPA security with the published evaluation keys, including any additional circular-security assumption required by the chosen key-switching construction.

4.2.1 Algebraic Foundations

BGV operates over the cyclotomic ring $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$, where t is the plaintext modulus, Q is the ciphertext modulus, and N is a power-of-two ring dimension. For N plaintext slots over $\mathbb{F}_t$, we take t prime with $t \equiv 1 \pmod{2N}$. A BGV ciphertext encrypting a plaintext vector $\vec{m} \in \mathbb{Z}_t^N$ consists of two polynomials in $\mathcal{R}_Q$. All these choices are standard in practice [1, 68].

Centered representatives and the infinity norm. Write $\mathcal{R} := \mathbb{Z}[X]/(X^N + 1)$ for the integer cyclotomic ring, so that $\mathcal{R}_Q = \mathcal{R}/Q\mathcal{R}$. For any positive integer M and any $a \in \mathbb{Z}$, the *centered representative* $[a]_M \in \mathbb{Z}$ is the unique integer in $(-M/2, M/2]$ satisfying $[a]_M \equiv a \pmod{M}$. We extend this coefficient-wise: for $f \in \mathcal{R}$, define $[f]_M \in \mathcal{R}$ by applying $[\cdot]_M$ to each coefficient. For an element a of a quotient ring $\mathcal{R}_M = \mathcal{R}/M\mathcal{R}$, the notation $[a]_M$ denotes the unique representative of the coset a in $\mathcal{R}$ with all coefficients in $(-M/2, M/2]$.

The *infinity norm* $\|f\|_\infty$ of $f \in \mathcal{R}$ is the maximum absolute value of its integer coefficients. Note that if $f, g \in \mathcal{R}$ satisfy $\|f\|_\infty + \|g\|_\infty < M/2$, then $[f + g]_M = f + g$ (no coefficient wraps modulo M).

Lifting $\mathcal{R}_t$ to $\mathcal{R}_Q$. For any $m \in \mathcal{R}_t$, its centered representative $[m]_t \in \mathcal{R}$ has coefficients in $(-t/2, t/2]$. Since $t \ll Q$, these satisfy $\|[m]_t\|_\infty < t/2 < Q/2$, so $[m]_t$ is also a valid centered representative modulo Q, i.e. $[[m]_t]_Q = [m]_t$. The *canonical lift* sends $m \in \mathcal{R}_t$ to the element of $\mathcal{R}_Q$ represented by $[m]_t$. Plaintext–ciphertext multiplication uses this lift in both BGV and BFV. For BFV plaintext addition, the message is scaled to match the ciphertext's plaintext encoding.

Norm bounds on centered representatives. Although $\|\cdot\|_\infty$ is only defined on $\mathcal{R}$, we can still bound the effect of $\mathcal{R}_Q$-arithmetic on centered representatives. The key tool is the *centered-reduction property*: for any $f \in \mathcal{R}$,

$$\|[f]_Q\|_\infty \le \|f\|_\infty . \tag{4}$$

The following bounds will be used in the noise analysis of Section 4.2.3 and the well-formedness tests of Appendix F.3.

Claim 4.2 (Norm bounds). *Let $n \ge 1$, $a, b, a_1, \ldots, a_n \in \mathcal{R}_Q$, and $c \in \mathbb{Z}_Q$. Then*

$$\begin{aligned} \|[ca]_Q\|_\infty &\le \|[c]_Q\|_\infty \, \|[a]_Q\|_\infty , \\ \|[ab]_Q\|_\infty &\le N \, \|[a]_Q\|_\infty \, \|[b]_Q\|_\infty , \\ \left\|[\textstyle\sum_{i\in[n]} a_i]_Q\right\|_\infty &\le \displaystyle\sum_{i\in[n]} \|[a_i]_Q\|_\infty . \end{aligned}$$

Proof. For the centered lifts in $\mathcal{R}$, these bounds follow from the triangle inequality, since each coefficient of a polynomial product is a signed sum of at most N coefficient products. Centered reduction preserves the bounds by Eq. (4). □

4.2.2 Encryption, Decryption, and Homomorphic Operations

Plaintext encoding. A plaintext vector $\vec{m} \in \mathbb{Z}_t^N$ is encoded as a polynomial in $\mathcal{R}_t$ via the inverse Number Theoretic Transform (INTT); decoding applies the forward NTT. We treat encoding and decoding as blackbox linear transformations over $\mathbb{Z}_t$, and henceforth write $m \in \mathcal{R}_t$ for the encoded plaintext. Similarly, BGV.Dec denotes full decryption inclusive of decoding, outputting a plaintext vector in $\mathbb{Z}_t^N$.

Encryption and the decryption function. A BGV ciphertext encrypting encoded plaintext $m \in \mathcal{R}_t$ under secret key $\mathsf{sk} \in \mathcal{R}_Q$ is a pair $\mathsf{ct} = (c_0, c_1) \in \mathcal{R}_Q^2$ satisfying

$$c_0 + c_1 \cdot \mathsf{sk} \;\equiv\; [m]_t + t \cdot e \pmod{Q},$$

where $e \in \mathcal{R}$ is a small noise term (with $\|e\|_\infty \ll Q/(2t)$). Symmetric-key encryption samples $c_1 \in \mathcal{R}_Q$ uniformly and sets $c_0 := [m]_t + t \cdot e - c_1 \cdot \mathsf{sk}$ modulo Q. We refer to [17, 16, 37] for public-key variants.

Decryption proceeds in two stages. The first is the $\mathcal{R}_Q$-linear *decryption functional* $\mathsf{L}_{\mathsf{sk}} \colon \mathcal{R}_Q^2 \to \mathcal{R}_Q$, defined by $\mathsf{L}_{\mathsf{sk}}(\mathsf{ct}) := c_0 + c_1 \cdot \mathsf{sk}$. Writing $y := \mathsf{L}_{\mathsf{sk}}(\mathsf{ct}) \in \mathcal{R}_Q$, the centered representative $[y]_Q \in \mathcal{R}$ decomposes uniquely as

$$[y]_Q = m + t \cdot e \quad \text{in } \mathcal{R}, \tag{5}$$

where $m := [[y]_Q]_t \in \mathcal{R}$ is the encoded plaintext (with $\|m\|_\infty \le t/2$) and $e := ([y]_Q - m)/t \in \mathcal{R}$ is the noise. The second stage extracts m from $[y]_Q$ via centered reduction modulo t, then decodes (forward NTT) to recover $\vec{m} \in \mathbb{Z}_t^N$. In BFV, the decryption functional is the same, but $y \equiv \Delta m + e \pmod{Q}$ where $\Delta = \lfloor Q/t \rfloor \in \mathbb{Z}$, $\|m\|_\infty \le t/2$, and e is small. The second stage of BFV scales $[y]_Q$ by t/Q, rounds coefficient-wise, and reduces modulo t before decoding. In both schemes, correct decryption requires that the accumulated noise remain small.

Homomorphic operations. BGV supports addition (including scalar and plaintext variants), multiplication, and slot automorphisms. Provided the operands and result satisfy the scheme's correctness bounds, the arithmetic operations satisfy the following properties:

- (Ciphertext add) $\mathsf{BGV.Dec}(\mathsf{ct}_1 + \mathsf{ct}_2) = \mathsf{BGV.Dec}(\mathsf{ct}_1) + \mathsf{BGV.Dec}(\mathsf{ct}_2)$
- (Scalar add) $\mathsf{BGV.Dec}(c + \mathsf{ct}) = \mathsf{BGV.Dec}(\mathsf{ct}) + c$ for $c \in \mathbb{Z}_t$
- (Plaintext add) $\mathsf{BGV.Dec}(\vec{c} + \mathsf{ct}) = \mathsf{BGV.Dec}(\mathsf{ct}) + \vec{c}$ for $\vec{c} \in \mathbb{Z}_t^N$
- (Scalar mult) $c \times \mathsf{BGV.Dec}(\mathsf{ct}) = \mathsf{BGV.Dec}(c \times \mathsf{ct})$ for $c \in \mathbb{Z}_t$
- (Plaintext mult) $\vec{c} \times \mathsf{BGV.Dec}(\mathsf{ct}) = \mathsf{BGV.Dec}(\vec{c} \times \mathsf{ct})$ for $\vec{c} \in \mathbb{Z}_t^N$
- (Ciphertext mult) $\mathsf{BGV.Dec}(\mathsf{ct}_1 \times \mathsf{ct}_2) = \mathsf{BGV.Dec}(\mathsf{ct}_1) \times \mathsf{BGV.Dec}(\mathsf{ct}_2)$

Slot automorphisms. Set $H := N/2$ and view the decoded slots as two rows of length H. We index a slot by $(b, u) \in \mathbb{Z}_2 \times \mathbb{Z}_H$, where b selects the row and u selects the position within that row. The first H decoded coordinates form row 0, and the remaining H coordinates form row 1. A row rotation and a row swap satisfy

$$\mathsf{BGV.Dec}(\mathsf{rot}(\mathsf{ct}, j))[b, u] = \mathsf{BGV.Dec}(\mathsf{ct})[b, u + j \bmod H], \tag{6}$$

$$\mathsf{BGV.Dec}(\mathsf{swap}(\mathsf{ct}))[b, u] = \mathsf{BGV.Dec}(\mathsf{ct})[b \oplus 1, u]. \tag{7}$$

A row rotation applies the same cyclic shift to both rows. A row swap exchanges the two halves. For $\delta = (a, j) \in \mathcal{G}_N := \mathbb{Z}_2 \times \mathbb{Z}_H$, define the slot action

$$\rho_\delta(b, u) := (b \oplus a, u + j \bmod H). \tag{8}$$

The BGV slot automorphisms are these compositions, indexed by $\delta \in \mathcal{G}_N$.

Additions and multiplications act element-wise over the plaintext vector $m \in \mathbb{Z}_t^N$. The slot automorphisms permute its coordinates. Together, these operations give the Single Instruction Multiple Data (SIMD) [71] property of BGV. Scalar operations use a plaintext vector with the same value in every slot. We distinguish them because scalar multiplication is faster and causes less noise growth than multiplication by a general plaintext. We treat these operations as black boxes and refer the reader to [17, 16, 37, 55] for implementation details.

4.2.3 Noise Budget and Efficiency

Multiplicative depth. Homomorphic evaluation consumes the available noise budget, and excessive noise can cause decryption to fail. The ciphertext modulus Q must be chosen large enough to accommodate noise growth of the entire computation, and larger moduli increase the cost of ciphertext arithmetic.

Noise growth. We characterize the cost of homomorphic operations based on their impact on the noise budget. Additions and slot automorphisms have negligible noise growth compared to multiplications, so we treat them as free in terms of multiplicative depth. Under our noise-based depth convention, scalar-ciphertext multiplication costs 0.5 levels. Plaintext-ciphertext and ciphertext-ciphertext multiplications each cost 1 level. These fractional levels measure noise consumption rather than ordinary circuit depth. Scalar and plaintext-ciphertext multiplication act linearly on the decryption value $\mathsf{L}_{\mathsf{sk}}(\mathsf{ct})$. With centered plaintext lifts, the corresponding infinity-norm bounds have factors $t/2$ and $tN/2$, respectively (Claim 4.2). The noise bounds also include the carry from reducing the plaintext product modulo t. For ciphertext-ciphertext multiplication, multiplying the decryption values $m_1 + te_1$ and $m_2 + te_2$ gives a worst-case coefficient bound with a quadratic contribution $tN \|e_1\|_\infty \|e_2\|_\infty$, in addition to terms linear in the input noises, plaintext carry, and relinearization noise. Slot automorphisms preserve the coefficient norm of the existing noise but add key-switching noise, whose size depends on the encryption and key-switching parameters [55]. Bootstrapping can refresh the noise budget. We restrict attention to leveled computation and do not use it.

Runtime via the Residue Number System. Operations are performed over a large ciphertext modulus Q. To implement this efficiently, BGV implementations employ the Residue Number System (RNS) to represent each coefficient modulo the prime factors of Q. Each prime is chosen to fit within a standard machine word ($\lessapprox 60$ bits) and gives one RNS limb.

Larger noise budgets require larger ciphertext moduli and typically more RNS limbs, increasing the cost of ciphertext arithmetic. The number of limbs need not equal the multiplicative depth, since the noise consumed per multiplication depends on the plaintext modulus and the operand noises, and the primes can have different bit lengths. Modulus switching removes limbs as evaluation progresses, reducing the cost of subsequent operations while retaining enough noise budget for the remaining computation.

Coefficient and NTT forms. Coefficient form stores polynomial coefficients, while NTT form stores their evaluations separately for each active RNS prime. These ciphertext transforms are distinct from the plaintext transform used for SIMD encoding. The artifact describes plaintext conversion and caching.

5 BhIPs for Wide Payload Circuits

5.1 Overview and Prerequisites

We now specify the wide payload circuits supported by the base and folded BhIPs, using the FOLD operator in Fig. 4.

Definition 5.1 (Wide Payload Circuit)**.** An FHE payload circuit is *wide* if it operates on all N SIMD slots of each ciphertext. Wide payload circuits may include BGV slot automorphisms and other cross-slot interactions.

Remark 5.2. Every payload circuit can be viewed as a wide circuit by treating unused slots as carrying dummy zero values. The BhIPs presented in this section apply to the general wide setting.

For the specialized setting of independent lanes, Appendix E gives a BhIP that packs the lane values more densely during proof generation.

5.1.1 Class of Supported Payload Circuits

The payload circuit is computed using six types of FHE operations: *CT-CT addition*, *PT-CT addition*, *PT-CT multiplication*, *scalar-CT multiplication*, *CT slot automorphism*, and *CT-CT multiplication*.

Our arithmetization of such a circuit, used by the proving stage, represents it as an alternating layered circuit of depth $d = 2d_\times + 1$. Here $d_\times$ counts CT-CT multiplication depth. PT-CT multiplication may consume FHE noise, but it remains affine in the wire values and does not create a multiplication layer. Its non-input layers use two gate types: *affine gates* in even layers and *SIMD multiplication gates* in odd layers. Layer 0, the output layer, is affine.

Affine gates. Affine gates support public coefficients that depend on the SIMD slot. They express CT-CT and PT-CT addition, multiplication by public plaintexts, BGV slot automorphisms, and compositions of these operations. The plaintexts may be nonuniform across slots. We keep the affine computation and its shared intermediate values so that coefficient openings can use its sparse local equations.

Dense SIMD multiplication gates. SIMD multiplication gates perform CT-CT multiplication and use the following dense wiring convention. After padding an odd multiplication layer $2i-1$ to width S_{2i-1}, the even layer beneath it has width $S_{2i} = 2S_{2i-1}$ and is logically arranged as a left half followed by a right half. For every $j \in \{0, \dots, S_{2i-1} - 1\}$, multiplication gate j takes gates j and $j + S_{2i-1}$ as inputs and computes their slot-wise product. If several multiplication gates use the same operand in the original payload, that operand is logically duplicated in the appropriate positions of layer $2i$.

We use layered circuits for a cleaner presentation (Remark 4.1). The techniques of [79] extend the affine reductions to gates that draw inputs from any lower layer. This extension uses standard sumchecks with wiring predicates for the cross-layer connections.

5.1.2 Data Layout and Notation

The arithmetized payload circuit has depth d, with output layer L_0 and input layer L_d. Each layer has S_i SIMD gates, each representing N scalar wire values. We assume that S_i is a power of two and $s_i := \log_2(S_i)$. Let $T_i := S_i \cdot N$ denote the number of scalar wires in layer i of the $\mathbb{F}_q$-arithmetic circuit, and let $t_i := \log_2(T_i) = s_i + \log N$. Let n_{FHE} denote the total number of primitive FHE operations in these layer computations, counting the operations in each affine subcircuit and the CT-CT multiplications. Shared subcomputations within an affine layer are counted once. We assume $S_0 + S_d = O(n_{\mathsf{FHE}})$, counting input and output ciphertexts. In our alternating layered arithmetization, we let $m := \sum_{i=1}^{d_\times} S_{2i-1}$ denote the total number of SIMD multiplication gates and $p := \sum_{i=0}^{d_\times} S_{2i}$ denote the total number of affine gates (including the output layer). Here the widths include public-zero padding. The trace has $m + p + S_d = 3m + S_0 + S_d$ logical ciphertext positions, consisting of payload-derived values, repeated references to the same payload ciphertext, and padding. Padding increases each nonempty multiplication layer by less than a factor of two, so this size remains linear in n_{FHE}. The alternating trace is obtained from ciphertexts already computed during payload evaluation.

When obvious from context, we drop subscripts on S_i, T_i, s_i, t_i. We index a wire by $(\vec{x}_c, \vec{x}_s) \in \{0,1\}^{s_i} \times \{0,1\}^{\log N}$, where $\vec{x}_c$ identifies its ciphertext and $\vec{x}_s$ its SIMD slot. The most significant slot bit selects the BGV row, and the remaining $\log(N/2)$ bits select the position within that row.

5.1.3 The Folding Operator

The FOLD operator in Fig. 4 sums the N slots of a ciphertext or plaintext. Its output has the same type and stores the sum in every slot.

The plaintext version uses the same permutations and additions locally. For an extended ciphertext or plaintext, FOLD applies this procedure independently to each of its R base-field components.

FOLD on BGV Slots

Let $H = N/2$, let $h = \log H$, and set $\mathsf{ct}^{(0)} \leftarrow \mathsf{ct}$.

1. For $j = 0, \ldots, h-1$, compute $\mathsf{ct}^{(j+1)} \leftarrow \mathsf{ct}^{(j)} + \mathsf{rot}\big(\mathsf{ct}^{(j)}, 2^j\big)$. After this loop, each slot contains the sum of its input row.
2. Add the other row: $\mathsf{FOLD}(\mathsf{ct}) \leftarrow \mathsf{ct}^{(h)} + \mathsf{swap}\big(\mathsf{ct}^{(h)}\big)$.

Every slot of $\mathsf{FOLD}(\mathsf{ct})$ encrypts $\sum_{(b,u)\in\mathbb{Z}_2\times\mathbb{Z}_H} \mathsf{BGV.Dec}(\mathsf{ct})[b,u]$.

Figure 4: Implementation of FOLD using the slot automorphisms from Section 4.2.

Remark 5.3. For one base-field ciphertext, the protocol uses $\log(N/2)$ row rotations, one row swap, and $\log N$ additions, for $\log N$ slot automorphisms in total. These costs apply to each of the R ciphertexts representing an extended ciphertext and add no multiplicative depth under the convention of Section 4.2.

5.2 Base and Folded BhIPs

Fix an alternating layered payload circuit Ckt of depth $d = 2d_\times + 1$. The GKR protocol performs d layer reductions, reducing a claim about L_0 (the output layer) to L_d (the input layer). Even layers use affine reductions, while non-input odd layers use multiplication reductions. For the bounded-depth costs below, we treat $d_\times$ as independent of the payload size.

Both variants use the same multilinear layer identities. For $j \in \{0, \ldots, d\}$, let $\tilde{L}_j : \mathbb{K}^{s_j + \log N} \to \mathbb{K}$ denote the multilinear extension of the wire values in layer j.

Base and Folded sumchecks. Both variants compute ciphertext-index messages from the encrypted payload trace. Base then sends the per-slot values needed for the verifier to complete the reduction locally. Folded performs the remaining slot-index rounds homomorphically, using FOLD to aggregate across slots. It therefore has $d \log N$ additional rounds and sends more ciphertexts before transcript packing.

Affine layer reduction. Fix an even layer $2i$ for $i \in \{0, \ldots, d_\times\}$. Let $\tilde{\mathsf{const}}_{2i}$ and $\tilde{\mathsf{lin}}_{2i}$ denote the multilinear extensions of its public constant terms and source-wire coefficients, respectively. The first two arguments of $\tilde{\mathsf{lin}}_{2i}$ index the output ciphertext and slot, and the last two index the source ciphertext and slot. At a challenge $(\vec{\gamma}_c, \vec{\gamma}_s) \in \mathbb{K}^{s_{2i} + \log N}$, the affine layer identity is

$$\tilde{L}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s) - \tilde{\mathsf{const}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s) = \sum_{\vec{x}_c, \vec{x}_s} \tilde{\mathsf{lin}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s, \vec{x}_c, \vec{x}_s) \tilde{L}_{2i+1}(\vec{x}_c, \vec{x}_s), \tag{9}$$

where the sum ranges over $\vec{x}_c \in \{0,1\}^{s_{2i+1}}$ and $\vec{x}_s \in \{0,1\}^{\log N}$. A full sumcheck first processes the s_{2i+1} source-ciphertext variables, followed by the $\log N$ source-slot variables. Composing an affine layer can make its coefficient table dense. For a fixed output challenge, however, the output MLE is a weighted sum of the layer's outputs, hence an affine function of the source-slot values. The prover computes these coefficients by traversing the original affine computation backwards. The verifier checks them with sumcheck against the computation's committed sparse local equations. The resulting openings take $\widetilde{O}(N n_{\mathsf{FHE}})$ field operations across all affine layers at bounded depth, as shown

in Appendix C. The public descriptions of all affine layers share one preprocessing commitment, and the resulting public opening claims are checked together with one PCS proof. The construction is given in Appendix C. When an affine layer has considerable structure, exploiting it can reduce the opening work further, as in the laned construction in Appendix E.3.1.

Multiplication layer reduction. Fix an odd layer $2i-1$ for $i \in \{1, \dots, d_\times\}$. The leading bit of a layer-$2i$ ciphertext index selects the left or right operand; the remaining s_{2i-1} bits identify the multiplication gate. At a challenge $(\vec{\gamma}_c, \vec{\gamma}_s) \in \mathbb{K}^{s_{2i-1}+\log N}$, the dense wiring gives

$$\tilde{L}_{2i-1}(\vec{\gamma}_c, \vec{\gamma}_s) = \sum_{\vec{x}_c, \vec{\ell}} \tilde{\mathsf{eq}}(\vec{\gamma}_c, \vec{x}_c)\tilde{\mathsf{eq}}(\vec{\gamma}_s, \vec{\ell})\tilde{L}_{2i}(0, \vec{x}_c, \vec{\ell})\tilde{L}_{2i}(1, \vec{x}_c, \vec{\ell}), \tag{10}$$

where the sum ranges over $\vec{x}_c \in \{0,1\}^{s_{2i-1}}$ and $\vec{\ell} \in \{0,1\}^{\log N}$. A full sumcheck first processes the s_{2i-1} gate-index variables, followed by the $\log N$ slot-index variables. The same gate index selects both operands, so no multiplication wiring predicate is needed. We apply Gruen's equality optimization [49] to send quadratic round polynomials, accounting for the public equality factor in the verifier's check. Each round then communicates two field elements per slot in Base, or two field elements in Folded, as detailed in Remark D.1.

Appendix D gives the full protocols. The affine reductions appear in Figs. 10 and 11, and the multiplication reductions in Figs. 12 and 13.

5.2.1 Prover Complexity

Each round computes its encrypted messages from saved payload ciphertexts or a fixed source checkpoint, using the public challenges sampled so far. Consequently, ciphertext depth does not accumulate across rounds or layer reductions. The FHE work is quasi-linear in the number of payload operations. We fix the security parameters and extension degree R, and use the public affine index and polynomial commitments of Theorem C.1.

Theorem 5.4 (Base BhIP prover complexity)**.** *Let* Ckt *be an alternating layered payload circuit of depth* d *computed with* n_{FHE} *FHE operations. Suppose that multiplication in* $\mathbb{K}$ *can be computed by a multiplicative-depth-*1 *circuit over* $\mathbb{F}_q$*. The Base BhIP prover performs* $O(n_{\mathsf{FHE}} \log n_{\mathsf{FHE}})$ *FHE operations and requires one multiplicative level beyond the payload evaluation. Its additional public computation, including the affine coefficient openings, uses* $\widetilde{O}(N n_{\mathsf{FHE}})$ *field operations at bounded depth.*

Theorem 5.5 (Folded BhIP prover complexity)**.** *Let* Ckt *be an alternating layered payload circuit of depth* d *computed with* n_{FHE} *FHE operations. Suppose that multiplication in* $\mathbb{K}$ *can be computed by a multiplicative-depth-*1 *circuit over* $\mathbb{F}_q$*. The Folded BhIP prover performs*

$$O\big(n_{\mathsf{FHE}} \log n_{\mathsf{FHE}} + dR \log(N n_{\mathsf{FHE}}) \log N\big)$$

FHE operations and uses at most 2.5 *additional multiplicative levels. In particular, the Folded prover performs* $\widetilde{O}(n_{\mathsf{FHE}})$ *FHE operations. Its additional public computation, including the affine coefficient openings, uses* $\widetilde{O}(N n_{\mathsf{FHE}})$ *field operations at bounded depth.*

For proofs and arbitrary-depth costs, see Appendix D.4.

6 Achieving Efficient VCoED

Section 5 defined two public-coin BhIP protocols (base and folded) for wide payloads. This section describes how to use these to build efficient end-to-end VCoED protocols. The same approach applies to the specialized laned BhIP in Appendix E, whose affine and multiplication reductions appear in Appendices E.3.1 and E.3.2.

6.1 Transcript Packing Compiler

To obtain a non-interactive argument, we apply Fiat–Shamir to the full interaction, including the packing argument when transcript packing is used. Let r denote the number of rounds of the BhIP, which sends $M = 2R$ ciphertexts per round after public zero padding. For transcript packing, we require $M < N$ and $M \mid N$. In the base BhIP, the verifier uses all N slots of each ciphertext. In the folded and laned BhIPs, each ciphertext instead encrypts one field element replicated across all N plaintext slots, as described in Section 5.1.3. Their Mr ciphertexts therefore use MrN plaintext slots to carry Mr message coordinates. Our **transcript packing** compiler places each round's M message values in assigned slots of a single *aligned ciphertext*. The prover commits to this ciphertext before receiving the next challenge, as required for IP soundness. After GKR and its public gate openings, the prover sends packed ciphertexts encoding the messages from all rounds and proves their consistency with the round commitments using interactive Spartan. Spartan uses a VCoED-compatible PCS (Definition G.2).

Preserving the plaintext transcript. The proof of packing constrains the ciphertext arithmetic, but *does not constrain the committed ciphertexts to be well-formed*. Honest alignment leaves all slots outside the round's assigned block empty. A malicious prover may instead commit to a ciphertext with values outside that block, so the packing relation applies another mask intended to keep only the assigned slots. The construction and general packing layout are given in Appendix F.1.3.

Consider two aligned ciphertexts $\mathsf{ct}_A, \mathsf{ct}_B \in \mathcal{R}_Q^2$ at the final modulus Q, assigned to disjoint sets of output slots $\mathcal{S}_A, \mathcal{S}_B \subseteq [N]$. Let $p_A, p_B \in \mathcal{R}$ be the centered lifts of the plaintext masks selecting these sets. The logical packing equation is

$$\mathsf{ct}_{\text{target}} = p_A \mathsf{ct}_A + p_B \mathsf{ct}_B. \tag{11}$$

The intended guarantee is that the packed ciphertext preserves each input's assigned plaintext slots:

$$\mathsf{BGV.Dec}(\mathsf{ct}_{\text{target}})[s] = \begin{cases} \mathsf{BGV.Dec}(\mathsf{ct}_A)[s], & s \in \mathcal{S}_A, \\ \mathsf{BGV.Dec}(\mathsf{ct}_B)[s], & s \in \mathcal{S}_B. \end{cases} \tag{12}$$

This homomorphic implication is guaranteed only while the noise remains below the BGV correctness bound. If a masked partial decryption overflows the centered interval modulo Q, centered reduction subtracts a multiple of Q. Subsequent reduction modulo t can then change the plaintext. Consequently, Eq. (11) can hold over $\mathcal{R}_Q$ while Eq. (12) fails.

An attack using malformed ciphertexts. This permits a direct attack. Although p_B is zero outside $\mathcal{S}_B$ in plaintext slots, its coefficient lift can be invertible in $\mathcal{R}_Q$. Suppose the prover has already committed to ct_A. After learning the challenge, a cheating prover can choose an accepting target $\mathsf{ct}_{\text{target}} \in \mathcal{R}_Q^2$ and set

$$\mathsf{ct}_B = p_B^{-1}(\mathsf{ct}_{\text{target}} - p_A \mathsf{ct}_A).$$

The resulting $\mathsf{ct}_B \in \mathcal{R}_Q^2$ satisfies the packing equation exactly, but its large noise can invalidate the homomorphic guarantee. Its mask can therefore affect the slots in $\mathcal{S}_A$ as well as those in $\mathcal{S}_B$.

The transcript packing argument checks the ring equation, so it does not detect this failure. We additionally need to test that the aligned ciphertexts are sufficiently well formed.

Well-formedness testing. To address this, we augment the proof of packing with a randomized *well-formedness test*. If all operand ciphertexts have low noise, then a linear combination with small, mean-zero coefficients also has bounded noise with high probability. Conversely, an operand with sufficiently large noise cannot reliably be hidden by cancellation in a random combination. Such an operand causes the combination to exceed the test's noise bound with constant probability, and independent repetitions amplify the probability of detecting it.

The test threshold must also leave enough room for the subsequent packing operations. We therefore decompose each position mask into D signed-digit masks with small coefficients to limit noise growth. We pack the aligned ciphertexts separately for each mask digit and apply the radix powers only after decryption, using D ciphertexts for each logical packed output. The parameter bounds ensure that if digit packing and plaintext recomposition change any message, each independent test rejects the fixed ciphertext list with probability at least $1/2$.

After all r commitments are fixed, the verifier samples T independent linear combinations of the committed ciphertexts using small, mean-zero coefficients. The prover sends the corresponding test ciphertexts, and the interactive packing argument proves that they are the prescribed combinations of the same committed list. The client uses its secret key to check the centered partial-decryption coefficients against a public bound. Thus, the probability of accepting such a list after all T tests is at most 2^{-T}, without requiring the client to receive and decrypt every committed ciphertext.

The test is specified in Appendix F.3, with analysis in Appendix F.4.

Theorem 6.1 (Transcript packing compiler)**.** *Let $\mathcal{E}[\pi] = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a public-coin BhIP for an indexed language $\mathcal{L}$, defined over a BGV scheme $\mathcal{E}$ with power-of-two ring degree N and plaintext modulus t. Fix $M \in [N-1]$ such that $M \mid N$. Fix a signed-mask digit count $D \geq 1$, a test repetition count $T \geq 1$, and a target completeness error $\epsilon_{\mathsf{cmp}} \in (0,1)$. Suppose the BhIP has $r \geq 1$ rounds, with M ciphertexts in each round $i \in [r]$ after public zero padding if needed. Each ciphertext encodes one base-field coordinate of a plaintext message of π by replication across all N plaintext slots. Use a collision-resistant hash H and interactive Spartan with a VCoED-compatible PCS (Definition G.2). Assume that honest alignment and the final modulus switch preserve the encoded messages. Let δ_{noise} bound the probability that honest ciphertexts exceed the coefficient bound at the packing boundary, ϵ_{wf} the bound on test-rejection probability when the coefficient bound holds, and $\delta_{\mathsf{alg,cmp}}$ the packing argument's completeness error.*

Then, under the hypotheses of Corollary F.4, with $\delta_{\mathsf{noise}} + \epsilon_{\mathsf{wf}} + \delta_{\mathsf{alg,cmp}} \leq \epsilon_{\mathsf{cmp}}$, the compiler described in Appendix F.1.1 gives a designated-verifier interactive argument $\pi' = (\mathsf{Setup}', \mathsf{P}', \mathsf{V}')$ for $\mathcal{L}$ with the following properties:

- ***Correctness and completeness:*** *If $\mathcal{E}[\pi]$ is perfectly complete and the packing argument has completeness error at most $\delta_{\mathsf{alg,cmp}}$, the compiled verifier reconstructs the original plaintext messages and accepts with probability at least $1 - \epsilon_{\mathsf{cmp}}$.*
- ***State-restoration soundness (ROM):*** *If the component errors in Theorem G.9 are negligible, then π' has negligible state-restoration soundness error against classical polynomial-time adversaries.*
- ***FHE overhead:*** *The compiler adds two plaintext-ciphertext multiplication levels. Besides preserving correct decryption through these operations, the FHE parameters must leave enough noise headroom at the packing boundary. The precise coefficient and modulus requirements are given in Corollary F.4. Producing the well-formedness test ciphertexts uses only ciphertext additions and negations, so it adds no multiplicative depth.*

- ***Communication:*** *For a fixed FHE setup, in addition to unchanged public messages, the prover sends r hash digests, $D\lceil Mr/N\rceil + T$ ciphertexts, and one interactive packing argument.*

Integrity argument. We bound the probability that the actual verifier accepts an incorrect output by relating it to the soundness error of an oracle protocol. In this protocol, GKR messages are ciphertext oracles fixed before each challenge. For a fixed client key, their decryptions determine the claims carried between rounds. Each round is an interactive oracle reduction (IOR), and composing these reductions with the gate openings gives an interactive oracle proof (IOP). Statistical RBR soundness bounds the chance that a false claim becomes true at a fresh challenge.

Base sends the contents of these oracles directly. For packed variants, an intermediate protocol checks the packing and test equations against the original ciphertext oracles, then uses the same packed decryption checks as the actual verifier. Well-formedness tests bound the probability of accepting a changed plaintext transcript.

Replacing the intermediate protocol's ciphertext oracles by collision-resistant hash commitments, opened at the end, gives state-restoration soundness [26]. Spartan replaces these final ciphertext openings with a proof. Its straight-line extractor recovers a list that opens the commitments and satisfies the same equations, turning a successful attack into one on the committed intermediate protocol except when extraction fails. With consistent PCS extraction, independent PCS and Fiat–Shamir random oracles, and negligible component errors, applying Fiat–Shamir once to the actual interaction gives negligible soundness error. The construction is described in Appendix F.1.1 and proved in Appendix G.5.

6.2 The Laminate VCoED Protocols

An end-to-end VCoED protocol is determined by the choice of BhIP and whether the transcript packing compiler is applied before Fiat-Shamir compilation, as depicted in Fig. 5.

The server homomorphically evaluates the payload circuit and uses the encrypted computation trace for proof generation. Before applying the chosen GKR prover, it prepares this trace by modulus switching and, for Laned with $B < N$, witness packing, while retaining enough noise budget for the subsequent operations. The client decrypts the returned output and verifies the proof using its retained input.

Let θ_{pay} be the multiplicative depth of the FHE payload evaluation, including scalar-ciphertext and plaintext-ciphertext multiplications (under the convention of Section 4.2). If proof generation uses θ_π additional multiplicative levels, the total required depth is $\theta \le \theta_{\mathsf{pay}} + \theta_\pi$.

End-to-End VCoED Protocol

Let Π denote the designated-verifier SNARG obtained from the chosen BhIP for the claim $\mathsf{Ckt}(\mathsf{in}) = \mathsf{out}$.

1. **Setup:** The client runs $\Pi.\mathsf{Setup}(1^{\lambda_{\mathsf{int}}}, 1^{\lambda_{\mathsf{priv}}}, \mathsf{Ckt})$. This generates the HE keys $(\mathsf{sk}, \mathsf{evk})$ and the SNARG prover/verifier keys $(\mathsf{pk}, \mathsf{vk})$. The client keeps vk (which contains sk) and sends $(\mathsf{pk}, \mathsf{evk})$ to the server.
2. **Client Encryption:** The client encrypts its inputs with a modulus chain supporting θ levels of multiplicative depth. It retains the input and its encryption for verification.
3. **Server Computation:** The server homomorphically evaluates the payload circuit:
$$\{\mathsf{ct}[\mathsf{out}]\} \leftarrow \mathcal{E}.\mathsf{Eval}_{\mathsf{evk}}(\mathsf{Ckt}, \{\mathsf{ct}[\mathsf{in}]\})$$
4. **Server Witness Preparation:** The server prepares the ciphertext witness for the chosen BhIP. For $\mathsf{Laminate}_{\mathsf{laned}}$ with $B < N$, it packs groups of N/B ciphertexts within each assignment layer using plaintext masks and additions, following the prescribed padding and layout in Appendix E.2. It reduces the ciphertext modulus to make proof arithmetic cheaper, while leaving enough noise budget for witness packing and subsequent proof operations. Let $\mathcal{W}$ denote the prepared witness.
5. **Server Proof Generation:** The server follows the interaction in Appendix F.1, using Fiat–Shamir to derive each public challenge from all preceding messages. For $\mathsf{Laminate}_{\mathsf{fold}}$ and $\mathsf{Laminate}_{\mathsf{laned}}$, GKR and its gate openings are followed by well-formedness testing and the interactive Spartan argument for transcript packing, including its polynomial openings. The resulting noninteractive proof is:
$$\pi \leftarrow \Pi.\mathsf{P}^{[\![\rho]\!]}\big(\mathsf{pk},\ (\mathsf{ct}[\mathsf{in}], \mathsf{ct}[\mathsf{out}]),\ \mathcal{W}\big)$$
6. **Server Response:** The server sends $\{\mathsf{ct}[\mathsf{out}]\}$ and proof π to the client.
7. **Client Verification:** The client recomputes the Fiat–Shamir challenges from the transcript, including the encrypted request and returned encrypted output. It decrypts the returned output to obtain out, and uses its retained input in and verifier key vk to verify the claim, with $(\mathsf{ct}[\mathsf{in}], \mathsf{ct}[\mathsf{out}])$ retained as verifier state:
$$\{\mathsf{acc}/\mathsf{rej}\} \leftarrow \Pi.\mathsf{V}^{[\![\rho]\!]}\big(\mathsf{vk},\ (\mathsf{in}, \mathsf{out}),\ \pi\big)$$

   The client returns out on acceptance and $\perp$ otherwise.

Figure 5: End-to-End VCoED Protocol

Protocol guarantees. Let r denote the number of rounds of the chosen BhIP, including the terminal round of each layer reduction. Public coefficient openings and their auxiliary sumchecks are counted separately. For the packed variants, D and T are the signed-mask digit count and the number of well-formedness tests from Theorem 6.1. The proof-size bounds exclude the encrypted output and the preprocessed public index.

Theorem 6.2 ($\mathsf{Laminate}$ VCoED protocols)**.** *Assume BGV is IND-CPA secure with its published evaluation keys, and that honest homomorphic evaluation, alignment, and modulus switching preserve the encoded messages. Use perfectly complete gate-opening IOPs with negligible RBR error and, for packed variants, the compiler of Theorem 6.1, under its stated hypotheses. Suppose multiplication in $\mathbb{K}$ has multiplicative depth one over $\mathbb{F}_q$. Assume the polynomial commitments are VCoED-compatible (Definition G.2), with negligible setup and authentication/extraction error. Choose field sizes, hash lengths, and test count so that the algebraic soundness, Fiat–Shamir hash-*

domain, and packing completeness errors are negligible. Fiat–Shamir applied to these interactions (Appendix F.1) yields designated-verifier VCoED protocols with negligible soundness error against classical polynomial-time adversaries in the random oracle model. They satisfy client privacy without a verification oracle, with the client's output and verification decision kept private.

- $\mathsf{Laminate}_{\mathsf{base}}$ *applies Fiat-Shamir directly to the Base BhIP. It uses one additional multiplicative level, and its encrypted proof contains at most* $2Rr$ *ciphertexts.*
- $\mathsf{Laminate}_{\mathsf{fold}}$ *applies transcript packing to the Folded BhIP, followed by Fiat-Shamir. For* $2R \mid N$ *and* $2R < N$*, it uses at most* 4.5 *additional multiplicative levels, and its proof contains* $D\lceil 2Rr/N \rceil + T$ *ciphertexts,* r *hash outputs, and the packing proof.*
- $\mathsf{Laminate}_{\mathsf{laned}}$ *applies transcript packing to the specialized laned BhIP in Appendix E, followed by Fiat-Shamir. For* $2R \mid N$ *and* $2R < N$*, it uses at most* 5.5 *additional multiplicative levels, including witness packing. Its proof contains* $D\lceil 2Rr/N \rceil + T$ *ciphertexts,* r *hash outputs, and the packing proof.*

All three proofs additionally contain the public coefficient-opening arguments of their respective BhIPs. For $\mathsf{Laminate}_{\mathsf{base}}$*, each affine reduction also contributes a public table of* N *elements of* $\mathbb{K}$*. With polynomial openings of the size specified in Table 2, their total proof sizes are as shown there.* $\mathsf{Laminate}_{\mathsf{base}}$ *is perfectly complete under the stated correctness conditions, and the packed variants have completeness error at most* ϵ_{cmp}*.*

After circuit preprocessing, these protocols also satisfy outsourceability for bounded-depth payload families with sublinear input/output processing, provided the cryptographic parameter costs (including ciphertext sizes) and the proof sizes and verification costs of the coefficient openings and packing argument are polylogarithmic in payload size.

Proof. We prove Theorem 6.2 by composing the BhIPs with the transcript packing compiler and Fiat-Shamir.

Completeness and depth overhead. The client retains its input and decrypts the returned output, so the statement checked by the verifier is precisely $\mathsf{Ckt}(\mathsf{in}) = \mathsf{out}$. Correct homomorphic evaluation recovers the honest BhIP messages, and the public coefficient openings are complete. $\mathsf{Laminate}_{\mathsf{base}}$ therefore inherits the completeness of its underlying proof. For the packed variants, the verifier reconstructs these messages and accepts with probability at least $1 - \epsilon_{\mathsf{cmp}}$ by Corollary F.4 and Theorem 6.1. These correctness conditions apply to witness preparation (including Laned witness packing), packing alignment, and the final modulus switch as well as the payload and BhIP computations.

By Theorems 5.4, 5.5 and E.3, the BhIP provers use at most 1, 2.5, and 3.5 additional multiplicative levels, respectively. The laned bound includes one level for witness packing. Transcript packing adds two levels by Theorem 6.1, giving the stated overheads. Witness preparation leaves enough noise budget for later proof operations, including multiplication reductions that use ciphertexts from earlier payload levels.

Rounds and proof sizes. The Base BhIP has $r = O(d \log n_{\mathsf{FHE}})$ rounds, while the Folded and Laned BhIPs have $r = O(d \log n)$ rounds, including one terminal round per layer reduction. Each round sends at most two extended ciphertexts, so Base sends at most $2Rr$ ciphertexts. For each packed variant, Theorem 6.1 gives $D\lceil 2Rr/N \rceil + T$ ciphertexts, r hash outputs, and the packing argument. All variants also send their public coefficient-opening arguments, and Base sends an additional N-entry table over $\mathbb{K}$ for each affine reduction. The exact round counts and the derivation of the total communication bounds are given in Appendix H.5.

Client privacy. The client's messages consist of its encrypted input and the public setup data. All subsequent messages, hashes, and proofs are computed by the server from this information. Thus an adversary distinguishing the server's views for two inputs would also distinguish the corresponding HE encryptions. Privacy follows from IND-CPA security, including the published evaluation keys, as assumed in Section 4.2. The client keeps its output and verification result private, as required by Remark 3.1.

Interactive soundness. Fix the client request and secret key. The returned encrypted output is fixed before the first output challenge, so it determines the statement $\mathsf{Ckt}(\mathsf{in}) = \mathsf{out}$ on every explored execution. Base transmits its ciphertext oracles in full, fixing their decoded plaintexts before the following challenge. By Lemma G.5, its statistical oracle error is the maximum of the GKR and gate-opening RBR errors. PCS setup and authentication contribute additional error terms. For Base, the state-restoration error is negligible for a polynomial number of fresh challenges by Eq. (69) and the gate-PCS guarantees.

For the packed variants, Lemma G.5 gives the intermediate IOP's RBR bound, including the well-formedness tests. Together with the packing argument's bound in Lemma G.7, it meets the hypotheses of Theorem G.9. Collision resistance and negligible component errors therefore give negligible state-restoration soundness.

Fiat–Shamir soundness. Apply the classical state-restoration compiler [10, 28] to the entire ordered interaction in Appendix F.1. Each challenge binds the setup, encrypted request and output, and all preceding messages and commitments. Spartan and its PCS openings are part of this interaction. The verifier's secret key affects only its final predicate, so the simulation samples challenges without that secret key. A reduction that analyzes a false encrypted-output claim may generate the designated-verifier keys itself and use them to interpret its transcript. The state-restoration error and the Fiat–Shamir hash-domain error are negligible under the theorem's parameter choices. This proves the soundness assertion of Theorem 6.2.

Outsourceability. We analyze the online client cost after circuit preprocessing. For bounded-depth payloads, the round counts in Eq. (80) are logarithmic in the layer widths and N. The client encrypts the input, decrypts the output and proof, evaluates the input and output MLEs, checks the layer reductions and public coefficient openings, and verifies the packing argument when present. The coefficient tables for $\mathsf{Laminate}_{\mathsf{base}}$ require an additional $O(dN)$ field operations. Under the theorem's polylogarithmic cost assumptions, all proof communication and verification beyond input/output processing is polylogarithmic in payload size. If input/output processing is sublinear in the time to evaluate the payload, the total online client cost is therefore sublinear as well. This establishes both requirements of outsourceability. The succinctness requirement on the coefficient openings is separate from the support-efficient prover-work bounds in Appendix C. □

Remark 6.3 (Verifiable CKKS)**.** CKKS [25] supports SIMD operations over approximate real numbers, whereas our GKR-based approach requires exact finite-field arithmetic. Recent work [11] extends sumcheck to approximate computations, offering a starting point for an approximate GKR protocol. As those authors note, obtaining such a protocol requires further analysis. Combining it with our techniques could then give *verifiable CKKS*, which we leave to future work.

7 Evaluation

In our benchmarks, per-request VCoED client time for computations with $n = 2^{34}$ scalar gates is 1.2–9.1% of scalar evaluation time. We measure the server cost of obtaining these client savings, relative to both FHE evaluation without integrity and prior FHE-over-SNARK constructions.

7.1 Methodology

Payload computations. We use two synthetic arithmetic workloads, *balancedLaned* and *mulHeavyPyramid*. Both have ciphertext-ciphertext multiplicative depth $d_\times = 8$ and operate over $\mathbb{F}_t$, with $t = 2^{16} + 1$. Each circuit has $n_{\mathsf{FHE}} = 2^{20}$ payload FHE operations and uses ring dimension $N = 2^{14}$. Each payload has 64 input ciphertexts and one output ciphertext. At full SIMD width, this gives 2^{34} scalar operations, counting each slot permutation as N routing operations.

The balancedLaned payload performs the same computation independently in each active SIMD slot. Approximately 50% of its operations are ciphertext multiplications, 45% are ciphertext additions, and 5% are additions of public scalars. Each of its eight multiplication layers contains 65,536 gates. In mulHeavyPyramid, approximately 85% of the operations are ciphertext multiplications, 10% are ciphertext additions, 2.5% are slot permutations, and 2.5% are multiplications by plaintexts whose entries vary across slots. Its multiplication layers become narrower towards the output. Most of its multiplications therefore occur near the inputs, where ciphertexts still have large moduli. The slot permutations and varying plaintext entries exercise the Wide protocols' support for general affine operations. Both workloads admit the compact affine representation described in Appendix C.

Our seven benchmark configurations use $\mathsf{Laminate}_{\mathsf{base}}$ and $\mathsf{Laminate}_{\mathsf{fold}}$ on both payloads at full SIMD width, and $\mathsf{Laminate}_{\mathsf{laned}}$ on balancedLaned with $B \in \{1, 128, 16384\}$ active lanes. The three Laned configurations have $n = B \cdot n_{\mathsf{FHE}} \in \{2^{20}, 2^{27}, 2^{34}\}$ scalar gates at fixed $n_{\mathsf{FHE}} = 2^{20}$. Base and Folded use their Wide protocols even on the lane-independent balancedLaned payload.

Implementation and hardware. We implement Laminate in C++ using the BGV backend of Microsoft SEAL [68] with Intel HEXL acceleration [13].[5] The experiments run on a Google Cloud `c4-highmem-48` machine with an Intel Xeon Platinum 8581C processor and approximately 393 GB of memory. We use a 10,000 GB HyperDisk (cloud-based remote block storage) provisioned with 2400 MB/s throughput and 150,000 IOPS. Each timed process is single-threaded and pinned to one physical core. The artifact's benchmark runbook gives the software versions, build settings, and host configuration. The local baseline evaluates the same $\mathbb{F}_t$ arithmetic circuit as the FHE evaluation, but does so in plaintext.

Measurements. All timings are wall-clock times, and each reported time comes from one execution. Server timings include payload evaluation, proof generation, and storage and serialization, including writing the encrypted trace to HyperDisk and reading it during proving. They exclude peer waiting and one-time preprocessing generation. Reusable-state loading is included for Base and FHE-only, but excluded for Folded and Laned, where it takes less than two minutes.

Client time is the total time spent on input encryption, output decryption, and proof verification for one request. Verification includes transcript decryption, GKR, and public coefficient openings. For the packed variants, it also includes the packing proof and well-formedness tests. Client time excludes generating or loading preprocessing, transferring or serializing requests and responses, and waiting for the server.

Proof sizes are measured after serialization in MB (10^6 bytes), including public proof components and excluding the encrypted output.

Parameter selection. We choose modulus chains to keep ciphertext arithmetic inexpensive while leaving some spare noise budget. Unnecessarily large chains inflate the measured costs, and an oversized FHE-only baseline would understate the overhead of integrity. All four variants use the same rule during payload evaluation: relinearize and modulus-switch after each ciphertext–

[5] Our implementation is available at https://github.com/kabirp/LaminateArtifactPublic.

Payload	Protocol	Key modulus (bits)	Privacy estimate (bits)
`balancedLaned`	$\mathsf{Laminate}_{\mathsf{base}}$	463	118.8
	$\mathsf{Laminate}_{\mathsf{fold}}$	637	86.5
	$\mathsf{Laminate}_{\mathsf{laned}}$, $B = N$	637	86.5
	$\mathsf{Laminate}_{\mathsf{laned}}$, $B \in \{1, 128\}$	677	81.8
`mulHeavyPyramid`	$\mathsf{Laminate}_{\mathsf{base}}$	550	99.9
	$\mathsf{Laminate}_{\mathsf{fold}}$	686	80.8

Table 6: Initial key-modulus sizes and approximate classical privacy estimates from the lattice estimator [74], at $N = 2^{14}$ with a uniform ternary secret and centered-binomial error of parameter 21. Key moduli include the special key-switching prime. We report the smaller of the primal-uSVP and primal-BDD estimates under the classical MATZOV cost model, using the GSA model and unbounded samples. Exact inputs and software versions appear in the artifact's privacy-estimation guide.

ciphertext multiplication layer. We tune prime sizes and limb counts within this rule, using noise-growth estimates and SEAL tests to check the remaining noise margin.

The proof-bearing variants reserve budget for encrypted GKR, while Folded and Laned must also meet the noise limits for transcript packing and well-formedness. The tests check that the selected parameters support the sampled computations, but do not bound the probability that an honest run exceeds its noise limit (Appendix H.2). The artifact's BGV parameter guide details the RNS prime chains and modulus-switching schedules, the empirical noise estimates used to select them, and the corresponding validation checks.

For comparison with Phalanx [80], we fix the ring dimension at $N = 2^{14}$. Table 8 reports the initial data-modulus bit lengths, excluding the special key-switching prime. Privacy estimates include that prime. Table 6 gives the key-modulus sizes and resulting classical privacy estimates, which exceed 80 bits for all evaluated proof-bearing configurations.

Concrete integrity after Fiat–Shamir. In the random oracle model, we allow a classical server $q_{\mathsf{mv}} = 2^{60}$ fresh protocol challenges across all attempts and final verification (Appendix G.1), a budget also considered by Block et al. [12, full version, Appendix A.3].

GKR uses $R = 8$, and packed variants use $T = 120$ well-formedness tests. We use BaseFold [78] at inverse code rate 8, with generic foldable codes for Wide gates and Reed–Solomon codes for Laned gates and packing. Under the computational budgets and hash assumptions in Appendix H.2, Corollary H.1 bounds the probability that the client accepts an incorrect result below 2^{-50}. For the packed variants, we assume birthday-bound collision resistance for Poseidon with six field elements for both its capacity and digest. The bound includes the errors from oracle compilation, polynomial openings, shared setup, and Fiat–Shamir.

7.2 Client Time and Proof Size

Client time versus scalar evaluation. Table 7 compares per-request VCoED client time with local scalar evaluation. Client computation takes approximately 4.3–9.1% of scalar evaluation time for $\mathsf{Laminate}_{\mathsf{base}}$ and 1.2–2.7% for $\mathsf{Laminate}_{\mathsf{fold}}$. These times change little between the two payloads despite mulHeavyPyramid's larger fraction of multiplications. For $\mathsf{Laminate}_{\mathsf{fold}}$, client time remains below 1.7 seconds for either computation.

Communication. $\mathsf{Laminate}_{\mathsf{fold}}$ gives 21.5–21.6× smaller proofs than $\mathsf{Laminate}_{\mathsf{base}}$. Its encrypted transcript has 124 ciphertexts, each using one 60-bit modulus limb: four packed ciphertexts and 120

Payload	Protocol	Local evaluation (s)	Client (s)	Client / local	Proof ciphertexts	Proof (MB)
`balancedLaned`	$\mathsf{Laminate}_{\mathsf{base}}$	61.94	5.660	0.0914	4,392	1162.6
	$\mathsf{Laminate}_{\mathsf{fold}}$	61.94	1.663	0.0269	124	54.1
`mulHeavyPyramid`	$\mathsf{Laminate}_{\mathsf{base}}$	132.00	5.612	0.0425	4,264	1127.5
	$\mathsf{Laminate}_{\mathsf{fold}}$	132.00	1.597	0.0121	124	52.1

Table 7: Client time and proof size for the two $n = 2^{34}$-gate circuits, evaluated using $n_{\mathsf{FHE}} = 2^{20}$ full-SIMD payload operations. Client time includes per-request input encryption, output decryption, and proof verification; client/local divides it by single-core local evaluation time (Section 7.1). Proof size includes all proof components and excludes the encrypted output.

Payload	Protocol	Initial modulus (bits)	Payload (h)	GKR (h)	Transcript packing (h)	Server total (h)	Total / no integrity
`balancedLaned`	No integrity	379	1.61	—	—	1.61	1.0×
	$\mathsf{Laminate}_{\mathsf{base}}$	430	2.90	19.04	—	21.94	13.6×
	$\mathsf{Laminate}_{\mathsf{fold}}$	577	4.68	38.07	10.37	53.12	32.9×
`mulHeavyPyramid`	No integrity	398	5.48	—	—	5.49	1.0×
	$\mathsf{Laminate}_{\mathsf{base}}$	490	7.50	47.48	—	54.98	10.0×
	$\mathsf{Laminate}_{\mathsf{fold}}$	626	10.31	92.99	10.41	113.71	20.7×

Table 8: Measured server wall time for the two $n = 2^{34}$-gate circuits (Section 7.1). No integrity denotes FHE evaluation without a proof. The initial ciphertext modulus excludes the special key-switching prime. Overhead is relative to FHE evaluation without integrity. Server totals include storage and serialization as well as the displayed phases, and exclude one-time preprocessing generation.

well-formedness tests (Appendix H.2). Commitments, public coefficient openings, and the packing proof account for the remaining bytes.

7.3 Server Overhead over FHE Evaluation

Table 8 measures the server cost of adding integrity to the same FHE payload computation. $\mathsf{Laminate}_{\mathsf{base}}$ takes 10.0–13.6 times as long as FHE evaluation without integrity, while $\mathsf{Laminate}_{\mathsf{fold}}$ takes 20.7–32.9 times as long.

Reserving noise budget for proof generation. Payload ciphertexts must retain noise budget for proving. On balancedLaned, $\mathsf{Laminate}_{\mathsf{base}}$ uses 51 additional initial modulus bits and $\mathsf{Laminate}_{\mathsf{fold}}$ uses 198, relative to FHE evaluation without integrity. Each variant's modulus-switching schedule also determines the budget retained for proving. Retaining larger ciphertexts increases payload arithmetic and trace-storage costs, particularly for multiplications near the inputs before modulus switching has shortened the ciphertexts. The measured payload times include both costs.

$\mathsf{Laminate}_{\mathsf{base}}$ versus $\mathsf{Laminate}_{\mathsf{fold}}$. GKR accounts for most of the server time, while Folded's packing takes about 10.4 hours on either payload. Across both payloads, $\mathsf{Laminate}_{\mathsf{fold}}$ uses 2.1–2.4× $\mathsf{Laminate}_{\mathsf{base}}$'s server time for 21.5–21.6× smaller proofs and 3.4–3.5× faster client computation, reflecting the tradeoff in Section 2.2.

7.4 Comparison with Prior Work

We compare our measured server totals with estimates for selected prover stages of Phalanx [80] and Blind Fractal [40], using each paper's hardware and primitive timings. The balancedLaned

Scalar gates n	Protocol	Server time (h)	Proof (MB)	Client (s)	Additional modulus bits
2^{20}	Laminate$_{\text{laned}}$	11.10	90.6	2.165	238
	Phalanx [80]	$\geq 0.28 + \mathsf{W}(1)$	61.4	≥ 0.20	≥ 300
	Blind Fractal [40]	$\geq 8.5 + \mathsf{W}(1)$	7.2	≥ 0.020	≥ 250
2^{27}	Laminate$_{\text{laned}}$	30.37	91.1	2.246	238
	Phalanx [80]	$\geq 36.7 + \mathsf{W}(128)$	62.1	≥ 0.20	≥ 300
	Blind Fractal [40]	$\geq 1479 + \mathsf{W}(128)$	10.4	≥ 0.029	≥ 250
2^{34}	Laminate$_{\text{laned}}$	54.49	91.5	2.387	198
	Phalanx [80]	$\geq 4705 + \mathsf{W}(2^{14})$	110.1	≥ 0.36	≥ 300
	Blind Fractal [40]	$\geq 265{,}804 + \mathsf{W}(2^{14})$	13.7	≥ 0.038	≥ 250

Table 9: Comparison on the `balancedLaned` payload, with $n_{\mathsf{FHE}} = 2^{20}$ for Laminate. Laminate server times are measured totals for payload evaluation, proof generation, and the associated storage and serialization. The numerical prior-work server bounds count only Phalanx's matrix products, or Blind Fractal's unpacking, matrix products, and initial NTTs. Laminate client times include per-request input encryption, output decryption, and proof verification. Prior-work client times are estimated lower bounds counting only proof-ciphertext decryption. Their proof sizes use favorable output assumptions (Appendix H.4). $\mathsf{W}(B)$ denotes the omitted time, in hours, for payload evaluation and witness reduction on $B = n/2^{20}$ lanes, separately for each prior scheme. The prior-work server bounds are rounded down and use each paper's hardware and primitive timings. Additional modulus bits exclude the special key-switching prime. For Laminate, they are measured relative to the 379-bit FHE-only `balancedLaned` chain. Prior-work values are the reported proving reserves at $n = 2^{20}$, kept fixed at larger sizes and excluding extra bits for witness reduction (Appendix H.4).

workloads have $n = 2^{20}$, 2^{27}, and 2^{34} scalar gates, each evaluated by Laminate using $n_{\mathsf{FHE}} = 2^{20}$ FHE operations. The prior-work estimates assume a single proof for an encrypted R1CS witness of size n, already in each scheme's required field and input layout.[6] For Blind Fractal, the larger instances fail the source's noise checks, and at $n = 2^{34}$ its original field is also too small for the code domain and stated integrity target. The estimates keep these parameters fixed and omit the cost of enlarging them.

Server cost at larger sizes. At $n = 2^{27}$, Phalanx's matrix products alone cost approximately 37 hours, while our complete laned computation takes 30.37 hours. At $n = 2^{34}$, Laminate$_{\text{laned}}$'s complete server time is less than 1/86 of Phalanx's matrix-product estimate and 1/4800 of the counted Blind Fractal stages. We do not establish a server advantage over Phalanx at $n = 2^{20}$. Our one-lane total includes 5.45 hours of payload evaluation and 0.99 hours of witness packing, both absent from the Phalanx bound.

Accounting for server time. The Phalanx estimate counts only its three R1CS matrix-vector products [80, Section 5.2]. The Blind Fractal estimate additionally counts input unpacking and the initial Reed–Solomon encodings [40, Section 6.2]. Both use the papers' small-instance primitive timings and omit later prover stages, parameter enlargement, memory and storage costs, and the payload evaluation and witness reduction denoted by $\mathsf{W}(B)$. Laminate server times include the complete computation, storage, and serialization. Appendix H.4 states the counting assumptions, and the artifact's comparison guide gives the calculations and reproduction script.

Communication estimates. For communication, we use Phalanx's reported average ciphertext

[6] Proving the B lanes separately, each with $n/B = 2^{20}$ gates, could reduce server time at the expense of larger total proofs and more client work. A single proof favors communication and client costs, while our server estimates count only selected stages at the papers' measured operation costs.

size and Blind Fractal's smallest published proof format for a server with no secret inputs [40, Remark 3]. The Blind Fractal server estimate excludes the conversion and repacking needed to produce that format. Appendix H.4 gives the counts and assumptions.

Witness reduction and storage. Delegation begins with the client's encrypted inputs, so the server must first evaluate the payload and construct the witness expected by the proof system (Section 1.1). This may require auxiliary values beyond the payload trace, non-native arithmetic if the proof system uses a different field from $\mathbb{F}_t$, and conversion from the payload's SIMD layout. The estimates also omit any resulting increase in witness size. $\mathsf{Laminate}_{\mathsf{fold}}$ uses the payload ciphertexts directly, over their original plaintext field and in their existing SIMD layout. The witness packing needed by $\mathsf{Laminate}_{\mathsf{laned}}$ is included in its measured server time.

The payload ciphertexts must retain enough noise budget for witness reduction and proving. Reserving this budget can require larger moduli during payload evaluation, making the payload itself more expensive, as Table 8 illustrates for our schemes. Simply adding a prover estimate to an FHE-only payload runtime would miss this cost.

In Phalanx's input layout, each witness value occupies a separate ciphertext. Materializing this unpacked encrypted witness would occupy approximately 211 TB at $n = 2^{27}$ and 27 PB at $n = 2^{34}$, before storing the matrix products. Streaming or recomputation could reduce the amount held in memory, but their costs are also omitted from the Phalanx estimate.

8 Open Questions

Several questions remain open. Can the overhead of proving be reduced even further? Can the approach be adapted to the CKKS [25] FHE scheme ([11] provides potential building blocks)? Can one obtain a publicly-verifiable scheme, or avoid the one-bit leakage, with comparable performance?

Acknowledgments. Zeyu Liu is supported by Yale CADMY.

References

[1] A. Al Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca. OpenFHE: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. Association for Computing Machinery, 2022. Updated version at https://eprint.iacr.org/2022/915. doi:10.1145/3560827.3563379.

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

[3] S. Angel, H. Chen, K. Laine, and S. T. V. Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00062.

[4] B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *Automata, Languages and Programming - ICALP 2010*, pages 152–163. Springer, 2010. doi:10.1007/978-3-642-14165-2_14.

[5] D. F. Aranha, A. Costache, A. Guimarães, and E. Soria-Vazquez. HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In *ASIACRYPT 2024, Part V*. Springer, Singapore, 2024. URL: https://eprint.iacr.org/2023/1949, doi:10.1007/978-981-96-0935-2_10.

[6] S. Atapoor, K. Baghery, H. V. L. Pereira, and J. Spiessens. Verifiable FHE via lattice-based SNARKs. *IACR Commun. Cryptol.*, 2024. doi:10.62056/a6ksdkp10.

[7] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. ACM, 1991. doi:10.1145/103418.103428.

[8] E. Ben-Sasson, D. Carmon, U. Haböck, S. Kopparty, and S. Saraf. On proximity gaps of Reed–Solomon codes. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*. ACM, 2026. Numbering follows the November 11, 2025 full version at https://eprint.iacr.org/2025/2055. URL: https://www.math.toronto.edu/swastik/rs-proximity-gaps-2025.pdf, doi:10.1145/3798129.3800827.

[9] E. Ben-Sasson, D. Carmon, Y. Ishai, S. Kopparty, and S. Saraf. Proximity gaps for Reed–Solomon codes. *J. ACM*, 2023. Theorem numbering follows the full version at https://eccc.weizmann.ac.il/report/2020/083. URL: https://eccc.weizmann.ac.il/report/2020/083/download, doi:10.1145/3614423.

[10] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC 2016-B, Part II*. Springer, Berlin, Heidelberg, 2016. URL: https://eprint.iacr.org/2016/116, doi:10.1007/978-3-662-53644-5_2.

[11] D. Bitan, Z. DeStefano, S. Goldwasser, Y. Ishai, Y. T. Kalai, and J. Thaler. Sum-check protocol for approximate computations. In *Advances in Cryptology – EUROCRYPT*

2026, Part VII. Springer, 2026. URL: https://eprint.iacr.org/2025/2152, doi:10.1007/978-3-032-25336-1_8.

[12] A. R. Block, A. Garreta, J. Katz, J. Thaler, P. R. Tiwari, and M. Zając. Fiat-Shamir security of FRI and related SNARKs. In *Advances in Cryptology - ASIACRYPT 2023*. Springer, 2023. Full version: https://eprint.iacr.org/2023/1071. doi:10.1007/978-981-99-8724-5_1.

[13] F. Boemer, S. Kim, G. Seifu, F. D. M. de Souza, and V. Gopal. Intel HEXL: Accelerating homomorphic encryption with Intel AVX512-IFMA52. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC '21, pages 57–62, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3474366.3486926.

[14] A. Bois, I. Cascudo, D. Fiore, and D. Kim. Flexible and efficient verifiable computation on encrypted data. In J. A. Garay, editor, *PKC 2021, Part II*, 2021. doi:10.1007/978-3-030-75248-4_19.

[15] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco. Single secret leader election. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, pages 12–24, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3419614.3423258.

[16] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO 2012*. Springer-Verlag, 2012. doi:10.1007/978-3-642-32009-5_50.

[17] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. doi:10.1145/2633600.

[18] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In R. Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, Oct. 2011. doi:10.1109/FOCS.2011.12.

[19] B. Bünz, P. Mishra, W. Nguyen, and W. Wang. Arc: Accumulation for Reed–Solomon codes. In *CRYPTO 2025, Part VII*. Springer, Cham, 2025. Full version. URL: https://eprint.iacr.org/2024/1731, doi:10.1007/978-3-032-01907-3_5.

[20] R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, and R. D. Rothblum. Fiat–Shamir from simpler assumptions. Cryptology ePrint Archive, Paper 2018/1004, 2018. URL: https://eprint.iacr.org/2018/1004.

[21] I. Cascudo, A. Costache, D. Cozzo, D. Fiore, A. Guimarães, and E. Soria-Vazquez. Verifiable computation for approximate homomorphic encryption schemes. In Y. T. Kalai and S. F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VII*, Lecture Notes in Computer Science, pages 643–677. Springer, 2025. doi:10.1007/978-3-032-01907-3_21.

[22] I. Cascudo, A. Costache, D. Cozzo, D. Fiore, A. Guimarães, and E. Soria-Vazquez. Batch, Pack, and Prove: More efficient verifiable computation for CKKS. Presentation at the FHE.org Conference 2026, Mar. 2026. Work in progress. Presentation slides. URL: https://fhe.org/conferences/conference-2026/resources/slides/1440_Guimaraes.pdf.

[23] H. Chen, Z. Huang, K. Laine, and P. Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In *ACM CCS 2018*. ACM Press, Oct. 2018. doi:10.1145/3243734.3243836.

[24] H. Chen, K. Laine, and P. Rindal. Fast private set intersection from homomorphic encryption. In *ACM CCS 2017*. ACM Press, Oct. / Nov. 2017. doi:10.1145/3133956.3134061.

[25] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017. doi:10.1007/978-3-319-70694-8_15.

[26] A. Chiesa, Z. Guan, C. Knabenhans, and Z. Yu. On the Fiat–Shamir security of succinct arguments from functional commitments. In *Advances in Cryptology – CRYPTO 2026, Part IX*. Springer, 2026. Numbering follows the June 2, 2026 full version. URL: https://eprint.iacr.org/2025/902, doi:10.1007/978-3-032-35424-2_7.

[27] A. Chiesa, D. Ojha, and N. Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *Advances in Cryptology - EUROCRYPT 2020*. Springer, 2020. doi:10.1007/978-3-030-45721-1_27.

[28] A. Chiesa and E. Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024. Version 1.2, March 25, 2026. URL: https://snargsbook.org/.

[29] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33. Springer, 2016. doi:10.1007/978-3-662-53887-6_1.

[30] K.-M. Chung, Y. T. Kalai, and S. P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In T. Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, Lecture Notes in Computer Science, pages 483–501. Springer, 2010. doi:10.1007/978-3-642-14623-7_26.

[31] K. Cong, R. C. Moreno, M. B. da Gama, W. Dai, I. Iliashenko, K. Laine, and M. Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In *ACM CCS 2021*. ACM Press, Nov. 2021. doi:10.1145/3460120.3484760.

[32] G. Cormode, M. Mitzenmacher, and J. Thaler. Practical verified computation with streaming interactive proofs. In *Innovations in Theoretical Computer Science 2012*. ACM, 2012. doi:10.1145/2090236.2090245.

[33] G. Cormode, J. Thaler, and K. Yi. Verifying computations with streaming interactive proofs. *Proc. VLDB Endow.*, 2011. URL: http://www.vldb.org/pvldb/vol5/p025_grahamcormode_vldb2012.pdf, doi:10.14778/2047485.2047488.

[34] A. Dalvi, A. Jain, S. Moradiya, R. Nirmal, J. Sanghavi, and I. Siddavatam. Securing neural networks using homomorphic encryption. In *2021 International Conference on Intelligent Technologies (CONIT)*, pages 1–7, 2021. doi:10.1109/CONIT51480.2021.9498376.

[35] B. E. Diamond and J. Posen. Succinct arguments over towers of binary fields. In *EUROCRYPT 2025, Part IV*. Springer, Cham, 2025. URL: https://eprint.iacr.org/2023/1784, doi:10.1007/978-3-031-91134-7_4.

[36] L. Ducas and D. Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015. doi:10.1007/978-3-662-46800-5_24.

[37] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://ia.cr/2012/144.

[38] D. Fiore, A. Nitulescu, and D. Pointcheval. Boosting verifiable computation on encrypted data. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, 2020. doi:10.1007/978-3-030-45388-6_5.

[39] B. Fisch, A. Lazzaretti, Z. Liu, and C. Papamanthou. ThorPIR: Single server PIR via homomorphic Thorp shuffles. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2024. doi:10.1145/3658644.3690326.

[40] M. Gama, E. H. Beni, J. Kang, J. Spiessens, and F. Vercauteren. Blind zkSNARKs for private proof delegation and verifiable computation over encrypted data. *IACR Commun. Cryptol.*, 2025. doi:10.62056/akgyl8n4e.

[41] M. Gama, E. H. Beni, J. Kang, J. Spiessens, and F. Vercauteren. Blind zkSNARKs: Performance estimation artifact, 2025. Commit `006d5cd`. URL: https://github.com/KULeuven-COSIC/blind_zkSNARKs.

[42] C. Ganesh, A. Nitulescu, and E. Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *J. Cryptol.*, 2023. doi:10.1007/s00145-023-09481-3.

[43] S. Garg, A. Goel, and M. Wang. How to prove statements obliviously? In *Advances in Cryptology - CRYPTO 2024*. Springer, 2024. doi:10.1007/978-3-031-68403-6_14.

[44] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology - CRYPTO 2010*. Springer, 2010. doi:10.1007/978-3-642-14623-7_25.

[45] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*. ACM, 2009. doi:10.1145/1536414.1536440.

[46] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: Interactive proofs for muggles. *Journal of the ACM*, 2015. Full version at https://eccc.weizmann.ac.il/report/2017/108. doi:10.1145/2699436.

[47] A. Golovnev, J. Lee, S. T. V. Setty, J. Thaler, and R. S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023*, volume 14082 of *Lecture Notes in Computer Science*, pages 193–226. Springer, 2023. doi:10.1007/978-3-031-38545-2_7.

[48] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A new hash function for Zero-Knowledge proof systems. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 519–535. USENIX Association, Aug. 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi.

[49] A. Gruen. Some improvements for the PIOP for ZeroCheck. Cryptology ePrint Archive, Paper 2024/108, 2024. URL: https://eprint.iacr.org/2024/108.

[50] U. Haböck. Basefold in the list decoding regime. Cryptology ePrint Archive, Paper 2024/1571, 2024. Revised September 4, 2025. URL: https://eprint.iacr.org/2024/1571.

[51] A. Henzinger, E. Dauterman, H. Corrigan-Gibbs, and N. Zeldovich. Private web search with Tiptoe. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pages 396–416, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3600006.3613134.

[52] K. Huang, Y. Guo, W. Qu, and J. Zhang. Linear-time, constant-depth blind polynomial commitments from generalized RAA codes, with an end-to-end blind SNARK implementation. Cryptology ePrint Archive, Paper 2026/487, 2026. URL: https://eprint.iacr.org/2026/487.

[53] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security 2018*. USENIX Association, Aug. 2018. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar.

[54] J. Kang, E. Pohle, and J. Spiessens. Verifiable computation on encrypted data from homomorphic encryption over random encodings. Cryptology ePrint Archive, Paper 2026/2422, 2026. URL: https://eprint.iacr.org/2026/2422.

[55] A. Kim, Y. Polyakov, and V. Zucca. Revisiting homomorphic encryption schemes for finite fields. In *ASIACRYPT 2021*. Springer, 2021. doi:10.1007/978-3-030-92078-4_21.

[56] C. Knabenhans, A. Viand, A. Merino-Gallardo, and A. Hithnawi. vFHE: Verifiable fully homomorphic encryption. In *Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. Association for Computing Machinery, 2024. Earlier version at https://arxiv.org/abs/2301.07041. doi:10.1145/3689945.3694806.

[57] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10:30039–30054, 2022. doi:10.1109/ACCESS.2022.3159694.

[58] J.-W. Lee, E. Lee, Y.-S. Kim, and J.-S. No. Rotation key reduction for client-server systems of deep neural network on fully homomorphic encryption. In *Advances in Cryptology – ASIACRYPT 2023*, pages 36–68. Springer Nature Singapore, 2023. doi:10.1007/978-981-99-8736-8_2.

[59] K. Lee and Y. Yeo. SophOMR: Improved oblivious message retrieval from SIMD-Aware homomorphic compression. In *35th USENIX Security Symposium (USENIX Security 26)*. USENIX Association, Aug. 2026. Full version at https://eprint.iacr.org/2024/1814. URL: https://www.usenix.org/conference/usenixsecurity26/presentation/lee-keewoo.

[60] F. Liu, H. Liang, X. Xie, Y. Yu, W. Zheng, and Y. Hu. HasteBoots: Proving TFHE programmable bootstrapping in seconds. In *35th USENIX Security Symposium (USENIX Security 26)*. USENIX Association, 2026. URL: https://www.usenix.org/conference/usenixsecurity26/presentation/liu-fengrun.

[61] Z. Liu, K. Sotiraki, E. Tromer, and Y. Wang. Snake-eye resistant PKE from LWE for oblivious message retrieval and robust encryption. In S. Fehr and P.-A. Fouque, editors, *EUROCRYPT 2025, Part III*, volume 15603 of *LNCS*, pages 126–156. Springer, Cham, May 2025. doi:10.1007/978-3-031-91131-6_5.

[62] Z. Liu and E. Tromer. Oblivious message retrieval. In *CRYPTO 2022, Part I*. Springer, Cham, Aug. 2022. doi:10.1007/978-3-031-15802-5_26.

[63] Z. Liu, E. Tromer, and Y. Wang. Group oblivious message retrieval. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 4367–4385, 2024. doi:10.1109/SP54263.2024.00115.

[64] Z. Liu, E. Tromer, and Y. Wang. PerfOMR: Oblivious message retrieval with reduced communication and computation. In *USENIX Security 2024*. USENIX Association, 2024. Full version at https://eprint.iacr.org/2024/204. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/liu-zeyu.

[65] W.-j. Lu, Z. Huang, C. Hong, Y. Ma, and H. Qu. PEGASUS: Bridging polynomial and non-polynomial evaluations in homomorphic encryption. In *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021. doi:10.1109/SP40001.2021.00043.

[66] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 1992. doi:10.1145/146585.146605.

[67] S. J. Menon and D. J. Wu. SPIRAL: Fast, high-rate single-server PIR via FHE composition. In *2022 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2022. URL: https://www.cs.utexas.edu/~dwu4/papers/Spiral.pdf.

[68] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, Jan. 2023. Microsoft Research, Redmond, WA.

[69] S. T. V. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Advances in Cryptology - CRYPTO 2020*. Springer, 2020. doi:10.1007/978-3-030-56877-1_25.

[70] S. T. V. Setty, J. Thaler, and R. S. Wahby. Unlocking the lookup singularity with Lasso. In M. Joye and G. Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 180–209. Springer, 2024. doi:10.1007/978-3-031-58751-1_7.

[71] N. P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *DCC*, 71(1):57–81, 2014. doi:10.1007/s10623-012-9720-4.

[72] J. Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology - CRYPTO 2013*. Springer, 2013. doi:10.1007/978-3-642-40084-1_5.

[73] J. Thaler. A note on the GKR protocol, 2015. URL: https://people.cs.georgetown.edu/jthaler/GKRNote.pdf.

[74] The Lattice Estimator developers. Lattice estimator. GitHub, 2026. Commit 3e48ef4. URL: https://github.com/malb/lattice-estimator/tree/3e48ef421ec256afddb3e7d2249a77eab6e9ba12.

[75] L. T. Thibault and M. Walter. Towards verifiable FHE in practice: Proving correct execution of TFHE's bootstrapping using plonky2. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2025. URL: https://eprint.iacr.org/2024/451, doi:10.1145/3719027.3765097.

[76] I. Tyurin. New estimates of the convergence rate in the Lyapunov theorem. *arXiv preprint arXiv:0912.0726*, 2009. URL: https://arxiv.org/abs/0912.0726.

[77] Y. Wang and F. Zhang. Qelect: Lattice-based single secret leader election made practical. In *USENIX Security 2025*. USENIX Association, 2025. Full version at https://eprint.iacr.org/2025/122. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/wang-yunhao.

[78] H. Zeilberger, B. Chen, and B. Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In *CRYPTO 2024, Part X*. Springer, 2024. Full version revised July 10, 2026. URL: https://eprint.iacr.org/2023/1705, doi:10.1007/978-3-031-68403-6_5.

[79] J. Zhang, T. Liu, W. Wang, Y. Zhang, D. Song, X. Xie, and Y. Zhang. Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. In *ACM CCS 2021*. ACM Press, 2021. URL: https://eprint.iacr.org/2020/1247, doi:10.1145/3460120.3484767.

[80] X. Zhang, R. Wang, Z. Liu, B. Xiang, Y. Deng, B. Fisch, and X. Lu. Phalanx: An FHE-friendly SNARK for verifiable computation on encrypted data. In *ACM CCS 2025*. ACM Press, 2025. Section and figure references use the October 19, 2025 full version. URL: https://eprint.iacr.org/2025/302, doi:10.1145/3719027.3765226.

[81] X. Zhang, R. Wang, Q. Niu, P. Liu, X. Lu, L. Zhao, R. Hou, and Y. Deng. Lasagne: Practical verifiable computation over encrypted data. Cryptology ePrint Archive, Paper 2026/857, 2026. URL: https://eprint.iacr.org/2026/857.

[82] M. Zhou, W.-K. Lin, Y. Tselekounis, and E. Shi. Optimal single-server private information retrieval. In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 395–425. Springer, Cham, Apr. 2023. doi:10.1007/978-3-031-30545-0_14.

[83] Z. Zhou, Y. Li, Y. Wang, Z. Yang, B. Zhang, C. Hong, T. Wei, and W. Chen. ZHE: Efficient zero-knowledge proofs for HE evaluations. In *2025 IEEE SP*, 2025. Corrected full version at https://eprint.iacr.org/2025/770. doi:10.1109/SP61157.2025.00199.

AI Acknowledgements

Our implementation and benchmarking of Laminate were assisted by Claude (Opus 4.8) and Codex (5.6 Sol). We started with a simple manual implementation without optimizations or transcript packing, implementing Laminate's core proof system under a mock SEAL backend, written without AI assistance. We then instructed the AI to incrementally flesh out the additional integrations and components: integrating the real SEAL library, adding Spartan with a mock PCS, BaseFold, Poseidon hash, etc. The coding agents were given this PDF writeup (in essentially full form except the benchmark results), Microsoft's Spartan repository, and earlier proof system code from one of the authors as a reference for style. The implementation took several months, with frequent feedback and revisions from the authors. We also used agents to generate the benchmark harness: measurement code, logging and progress tracking.

We used GPT 6 Astra to improve the readability of Fig. 3, starting with a manually-prepared full draft and preserving the content and layout.

A Other Approaches to VCoED

We give further details of the approaches summarized in Section 1.4 and their limitations for outsourcing computation from a weak client to a powerful server.

A.1 SNARK-over-FHE

An alternative is to prove directly that the server performed the ciphertext operations correctly. Proof systems tailored to FHE [38, 14, 42, 6] reduce the cost of expressing these operations, and subsequent work supports approximate arithmetic [21]. These constructions can provide succinct verification, but proving ciphertext operations such as relinearization remains expensive for the server [56]. An unpublished follow-up by Cascudo et al., *Batch, Pack, and Prove* [22], substantially reduces the server overhead of their earlier construction [21]. The remaining server overhead is still higher than that reported in existing FHE-over-SNARK evaluations [5, 40, 80], and the experiments do not establish concrete outsourceability.

Thibault and Walter [75] and HasteBoots [60] specialize SNARK-over-FHE to TFHE programmable bootstrapping. Proving a single bootstrapped operation still takes seconds to minutes, leaving substantial server overhead.

A.2 VOLE-over-FHE

ZHE [83] uses proofs based on Vector Oblivious Linear Evaluation (VOLE) to verify ciphertext operations while protecting private server inputs. It improves proving efficiency over earlier SNARK-over-FHE implementations, but its proof size and verifier work are linear in the ciphertext computation. In the workloads reported in its corrected benchmarks, even online verification takes over two orders of magnitude longer than the underlying FHE evaluation (which is already a few orders of magnitude slower than the payload computation).[7] Such a tradeoff can be useful for two-party computation, where the client cannot compute the result locally without access to the server's private inputs. It does not meet the sublinear client requirement of our outsourcing setting.

[7] We refer to the corrected benchmarks in Section 7 of the September 2026 extended version of [83].

A.3 Composing Garbled Circuits and FHE

The original VCoED construction of Gennaro et al. [44] achieves outsourceability by combining garbled circuits with FHE. However, for every Boolean gate of the payload, the server must homomorphically evaluate the symmetric decryption and validity checks used to evaluate the garbled gate. This replaces each payload gate with a cryptographic computation under FHE, introducing substantial server work beyond the payload's arithmetic.

A.4 Hidden Challenges

Chung et al. [30] obtain efficient online verification using hidden challenges and expensive circuit-dependent preprocessing. Their reusable construction requires nested homomorphic evaluation, and delegating the preprocessing adds further homomorphic work and a universal argument. Applebaum et al. [4] give a related approach based on randomized encodings and MACs, using garbled circuits and FHE.

B Additional Background on GKR

We give further details of the GKR protocol, using the notation from Section 4.1. The standard reduction for layer i uses $2s_{i+1}$ sumcheck rounds. The prover can compute all of its sumcheck messages in time quasi-linear in the size of the circuit [32, 73].

B.1 Leveraging Circuit Structure

Circuit structure can improve both the layer reduction and the evaluation of the gate indicators in Eq. (3).

Data-parallel circuits. A computation is data parallel when it applies the same subcircuit independently to many inputs. A gate address then separates into a copy index and an index within one copy. Since the copies do not interact, the gate indicators enforce equality of the copy indices and use the indicators of one copy for the remaining coordinates. Thaler [72] uses this factorization so that circuit preprocessing stores the wiring data for a single copy. The resulting layer polynomial has one copy index shared by the output gate and both input wires. The two input indices within a copy may remain independent. Thus data parallelism changes the layer reduction even when the wiring within one copy is irregular. The specialized laned construction in Appendix E has this form, with the lane coordinate serving as the copy index. Its dense wiring gives an additional reduction in the number of sumcheck variables.

MLE-structured functions. Evaluating the multilinear extension of a general function on $\{0,1\}^k$ takes $O(2^k)$ field operations. Following [70], a function is *MLE-structured* when its multilinear extension has a succinct evaluation algorithm. For example, define $\mathsf{OR} : \{0,1\}^k \to \{0,1\} \subseteq \mathbb{K}$ by

$$\mathsf{OR}(\vec{z}) = \begin{cases} 0 & \text{if } \vec{z} = \vec{0}, \\ 1 & \text{otherwise.} \end{cases}$$

For $\vec{Z} \in \mathbb{K}^k$, its multilinear extension is

$$\widetilde{\mathsf{OR}}(\vec{Z}) = 1 - \prod_{j=1}^{k} (1 - Z_j).$$

This expression can be evaluated using $O(k)$ field operations.

MLE-structured gate indicators. If add_i and mul_i are MLE-structured, the indexed circuit description includes succinct algorithms for evaluating $\tilde{\mathsf{add}}_i$ and $\tilde{\mathsf{mul}}_i$. The verifier runs these algorithms whenever Eq. (3) requires a gate-indicator evaluation. These indicators require no polynomial commitment.

Sparse gate indicators. For an arbitrary circuit, the Boolean tables for add_i and mul_i need not have a useful evaluation formula. They are still sparse because the two tables contain at most one nonzero entry per gate. Spark [69, 70] is a standard method for committing to these sparse gate-indicator polynomials and proving their evaluations.

Virtual gate polynomials. A gate polynomial may also be represented implicitly through related polynomials. When the verifier requests an evaluation of the gate polynomial, the prover reduces the claim to evaluations of these related polynomials. The reduction may use public MLE-structured functions, short sumchecks, and additional prover commitments. The prover then opens the resulting evaluations. Following [35], we call the gate polynomial a *virtual polynomial* because the protocol uses it without committing to it directly.

For wide payloads (Appendix C), the prover answers coefficient queries by certifying a public reverse traversal against commitments to the sparse local affine equations. In the laned case (Appendix E.3.1), the verifier computes the lane-equality factor locally, so the prover only needs to open the smaller scalar coefficient polynomial.

Support-efficient commitments. We call a multilinear PCS *support-efficient* if preprocessing and the opening prover each use $\widetilde{O}(Q)$ field operations for a Boolean table with Q nonzero entries. We use such commitments for the local affine equations in Appendix C and the scalar coefficient tables in Appendix E.3.1. Additional structure in a payload may give a more efficient virtual representation or a direct MLE evaluation.

C Affine Coefficients and Openings

A short affine computation can have a large coefficient table. For example, computing all prefix sums $y_j = x_1 + \cdots + x_j$ of k input ciphertexts takes only $k-1$ additions, but the resulting coefficient table has $\Theta(k^2 N)$ nonzero entries. We only need the coefficients of one weighted sum of outputs, which can be computed by working backwards. In each slot, output weights $\chi_1, \ldots, \chi_k$ give input x_i the weight $\chi_i + \cdots + \chi_k$. Computing these suffix sums takes $O(k)$ work per slot, or $O(kN)$ in total.

Write affine layer $2i$ as a map on scalar slots:

$$F(\mathbf{x}) = A\mathbf{x} + \mathbf{b}, \tag{13}$$

where $\mathbf{x} \in \mathbb{F}_q^{NS_{2i+1}}$ and $F(\mathbf{x}) \in \mathbb{F}_q^{NS_{2i}}$. The multilinear extensions of the entries of A and $\mathbf{b}$ are $\tilde{\mathsf{lin}}_{2i}$ and $\tilde{\mathsf{const}}_{2i}$, evaluated over $\mathbb{K}$.

Computing the requested coefficients. Evaluating the output MLE at $\vec{\gamma}$ gives a weighted sum of outputs, with weights $\chi_{\vec{\gamma}}(\vec{v}) = \tilde{\mathsf{eq}}(\vec{\gamma}, \vec{v})$ at Boolean output addresses $\vec{v}$. For a scalar wire u, its *adjoint* $z_u \in \mathbb{K}$ describes its effect on this weighted output: adding $\delta \in \mathbb{K}$ to the value at u and recomputing downstream operations changes the weighted output by $z_u \delta$. The input adjoints are therefore the coefficients $A^{\mathsf{T}} \chi_{\vec{\gamma}}$. All adjoints depend only on the circuit and $\vec{\gamma}$, and can be computed in reverse topological order using the local affine equations below.

The requested coefficient and constant are

$$\tilde{\mathsf{lin}}_{2i}(\vec{\gamma}, \vec{r}) = \langle A^{\mathsf{T}} \chi_{\vec{\gamma}}, e_{\vec{r}} \rangle, \qquad \tilde{\mathsf{const}}_{2i}(\vec{\gamma}) = \chi_{\vec{\gamma}}^{\mathsf{T}} \mathbf{b}, \tag{14}$$

where $e_{\vec{r}}$ is the Boolean equality vector for the source point $\vec{r}$. This traversal visits each original operation once, even when many paths share a wire.

Certifying the computation. For each computed scalar wire v, write its local affine equation as $x_v = b_v + \sum_u a_{v,u} x_u$, where the sum runs over its inputs. Let χ_u be the requested weight of wire u if it is an output, and zero otherwise. The adjoints satisfy

$$z_u = \chi_u + \sum_{v:\, u \text{ feeds } v} a_{v,u} z_v. \tag{15}$$

Since the circuit is acyclic, these equations determine all weights uniquely. The input weights give $A^{\mathsf{T}} \chi_{\vec{\gamma}}$, and the sum $\sum_v b_v z_v$ over computed wires gives $\chi_{\vec{\gamma}}^{\mathsf{T}} \mathbf{b}$. The *internal adjoints* are the adjoints of computed wires, including those that serve as outputs. In the general construction, the prover commits to these adjoints and claims the coefficient and constant evaluations in Eq. (14). The input coefficients and direct output contributions follow from these adjoints and the public local equations. Using sumcheck, the verifier reduces these identities and the weight equations to opening claims about the committed weights and the public circuit description. These openings authenticate the coefficient values used during GKR.

The public index. Each public sparse table lists its nonzero entries, with one column for the values and one for each address bit. The addresses determine how each stored value contributes to a requested gate evaluation. Using sumcheck, the gate claims reduce to opening claims about these columns.

All affine layers share one public index for their wiring, slot actions, constants, and plaintext masks. During preprocessing, we pack these arrays into equal-length columns, encode each column honestly, and commit to them together. All public opening claims left by the local checks are batched into one PCS opening against this commitment. The verifier checks the public equality and selection weights itself.

Any internal adjoints require additional online commitments. The prover may concatenate several adjoint tables in a fixed block layout and encode them together. It commits to the result before the verifier checks any of the tables. A block evaluation is an inner product with a public equality polynomial that selects the block and its evaluation point, so these claims share one PCS opening. We require the PCS guarantees of Definition G.2 without assuming that the prover encoded the adjoint tables correctly. Every claimed evaluation of an adjoint or public table remains pending until its corresponding opening succeeds.

Theorem C.1 (Wide affine coefficient openings)**.** *Assume honest preprocessing and a perfectly complete multilinear PCS satisfying the fixed-polynomial and RBR-opening requirements of Definition G.2, with opening error ϵ_{PCS}. Let d_{gate} bound the total degree of each check in its fresh verifier challenges, which can reach $\log N$ for Base's slot table. Then the gate-opening IOP is perfectly complete and, conditioned on good setup, has RBR error at most*

$$\epsilon_{\mathsf{gate\text{-}batch\text{-}PCS}} \le \max\{d_{\mathsf{gate}}/|\mathbb{K}|, \epsilon_{\mathrm{PCS}}\}. \tag{16}$$

Authentication and setup failures are accounted for separately.

Proof. If an adjoint commitment has interpretation $\perp$, we keep the state bad until its required PCS opening, which starts in a bad state by the PCS assumption. Otherwise, the commitments fix the adjoint polynomials, while honest preprocessing fixes the public columns. Incorrect weights violate a local equation, and incorrect coefficient values violate one of the requested identities. Each claim is fixed before its checking challenge, so a false identity passes its polynomial test with probability at most $d_{\mathsf{gate}}/|\mathbb{K}|$.

For the public batch, let $C_1, \ldots, C_w \in \mathbb{K}^m$ be the column vectors and $\phi_1, \ldots, \phi_h \in \mathbb{K}^m$ the public weights for the requested openings. Each ϕ_t selects a block and applies equality weights within it, with zero weights elsewhere. The prover supplies the complete matrix $M \in \mathbb{K}^{h \times w}$ of claimed inner products $M_{t,j} \stackrel{?}{=} \langle C_j, \phi_t \rangle$. This matrix and all its operands, including those derived from local requests, are fixed before the verifier samples fresh independent uniform column weights $\lambda_1, \ldots, \lambda_w \in \mathbb{K}$. The verifier checks the operands and consistency of M with the sparse-table terminal claims. The column batch replaces these claims by

$$C_\lambda := \sum_{j=1}^{w} \lambda_j C_j, \qquad \langle C_\lambda, \phi_t \rangle \stackrel{?}{=} \sum_{j=1}^{w} \lambda_j M_{t,j} \quad (t = 1, \ldots, h). \tag{17}$$

If M is incorrect, it has a nonzero row of errors whose inner product with the column weights is zero with probability at most $1/|\mathbb{K}|$. With C_λ and these claims fixed, the verifier combines the request vectors ϕ_t and their claimed inner products using a fresh set of independent uniform field weights. This gives one PCS opening against the original joint commitment. Each batching round has RBR error at most $1/|\mathbb{K}|$, and therefore satisfies the stated $d_{\mathsf{gate}}/|\mathbb{K}|$ transition bound. Any remaining false claim is reduced by sumcheck to a false opening claim, which the verifier checks with the PCS. Keeping each claim until it is checked gives the stated maximum. □

Base coefficient tables. Base fixes and sends its N slot coefficients before the verifier samples a slot point and checks the table's MLE against one coefficient opening. A wrong table gives a nonzero multilinear discrepancy of degree at most $\log N$, included in d_{gate}. The client still processes all N entries.

Costs. The shared mask dictionary contains $D_{\text{mask}} = O(Nn_{\mathsf{FHE}})$ slot entries, revisited across the d layers. Fix the security parameters and extension degree R. With a constant number of requests per layer and quasilinear PCS work, the public computation takes $\widetilde{O}(Nn_{\mathsf{FHE}} + dD_{\text{mask}})$ field operations, and hence $\widetilde{O}(Nn_{\mathsf{FHE}})$ at bounded depth.

Compact representations. When an affine map has a compact sparse description, the prover can authenticate its coefficients and constants without committing to internal adjoints. Our implementation finds such descriptions for every affine layer of both evaluated Wide payloads. When preprocessing cannot find one, the prover uses the reverse-trace construction with the cost bound above. The artifact describes how preprocessing chooses between these representations.

D Layer Reductions for Wide Payload Circuits

We give the Base and Folded reductions and analyze their prover costs using the circuit model and data layout from Section 5. The affine coefficient polynomials and their opening protocols are defined in Appendix C.

D.1 Ciphertext and Plaintext Notation

For $x : \{0,1\}^{\log N} \to \mathbb{F}_q$, we use $\mathsf{pt}[x]$ to denote the plaintext whose $\vec{\ell}$th slot stores $x(\vec{\ell})$, and $\mathsf{ct}[x]$ for a ciphertext encrypting it. For $x : \{0,1\}^{\log N} \to \mathbb{K}$, we use $\mathsf{ept}[x]$ and $\mathsf{ect}[x]$ for the corresponding extended plaintext and ciphertext. Each is represented by R base-field plaintexts or ciphertexts, respectively, as in Section 3.1.

The layer-reduction figures describe witness-dependent messages by their decrypted values unless a ciphertext is written explicitly. The prover sends these messages encrypted, while public gate coefficients and verifier challenges remain in the clear.

Folded Affine Layer Reduction

1. **Full Sumcheck:** P supplies $a \stackrel{?}{=} \tilde{\mathsf{const}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s)$, which V records for the final gate-opening batch. V subtracts a from the current claim. P and V run sumcheck on the right-hand side of Eq. (9) for $s_{2i+1} + \log N$ rounds. At the end, V has challenges $\vec{r}_c \in \mathbb{K}^{s_{2i+1}}$ and $\vec{r}_s \in \mathbb{K}^{\log N}$ and a claim

$$c \stackrel{?}{=} \tilde{\mathsf{lin}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s, \vec{r}_c, \vec{r}_s)\tilde{L}_{2i+1}(\vec{r}_c, \vec{r}_s). \tag{18}$$

2. **Evaluation Claims and Check:** P supplies

$$g \stackrel{?}{=} \tilde{\mathsf{lin}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s, \vec{r}_c, \vec{r}_s), \qquad v_x \stackrel{?}{=} \tilde{L}_{2i+1}(\vec{r}_c, \vec{r}_s).$$

V records g, its layer, and both evaluation points for authentication by Appendix C. V checks $c \stackrel{?}{=} g v_x$. If the check passes, the claim for v_x becomes the starting claim for layer $2i + 1$.

Figure 10: Specification of an affine layer $2i \to 2i + 1$ reduction for the Folded hIP.

D.2 Affine Layer Reduction

Fix an even layer $2i$ for $i \in \{0, \ldots, d_\times\}$. Its gates may use additions, BGV slot automorphisms, and multiplication by public plaintexts, including nonuniform ones. Taking the MLE of Eq. (13) gives the layer identity in Eq. (9). The protocols below use the affine index from Appendix C. Each affine reduction starts with one layer-evaluation claim and returns one source-layer claim. Every supplied constant or coefficient value is fixed when used and recorded with its layer and evaluation point. These values are authenticated after GKR using Appendix C. These public computations are performed outside FHE.

Base hIP. The Base protocol runs only the s_{2i+1} ciphertext-index rounds. Fixing $(\vec{\gamma}_c, \vec{\gamma}_s, \vec{r}_c)$ in $\tilde{\mathsf{lin}}_{2i}$ leaves a multilinear polynomial in the $\log N$ source-slot variables. The prover sends its values at all N Boolean assignments. These values are checked together in the final gate-opening batch, as described in Appendix C.

If the verifier stores the explicit affine index, it may compute the coefficient values locally. The figures give the commitment-based protocol used with a succinct index. The concrete ciphertext counts use one terminal message $\mathsf{ect}[w]$ per affine reduction.

D.3 Multiplication Layer Reduction

Fix an odd layer $2i - 1$ for $i \in \{1, \ldots, d_\times\}$. The dense layout gives $s_{2i} = s_{2i-1} + 1$, with the leading source-index bit selecting the left (0) or right (1) operand. For every $\vec{x}_c \in \{0,1\}^{s_{2i-1}}$ and slot $\vec{x}_s \in \{0,1\}^{\log N}$,

$$L_{2i-1}(\vec{x}_c, \vec{x}_s) = L_{2i}(0, \vec{x}_c, \vec{x}_s) \cdot L_{2i}(1, \vec{x}_c, \vec{x}_s). \tag{22}$$

Taking the MLE of Eq. (22) gives the layer identity in Eq. (10). The terminal source evaluations are taken at points that differ only in the leading left/right selector. Since the source MLE is linear in this coordinate, a fresh selector challenge combines them into one source-layer evaluation.

Remark D.1 (Gruen's equality optimization)**.** Although the summand in Eq. (10) has individual degree three, its equality factor is public. Following Gruen [49, Section 3], we account for this factor in the verifier's check and send evaluations of a quadratic round polynomial. Write $\vec{\gamma} = (\vec{\gamma}_c, \vec{\gamma}_s)$ and

Base Affine Layer Reduction

1. **Sumcheck (Early Termination):** P supplies

$$a \stackrel{?}{=} \tilde{\mathsf{const}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s),$$

which V records for the final gate-opening batch. V subtracts a from the current claim and calls the result c_0. In round $j \in \{1, \ldots, s_{2i+1}\}$, after challenges $\vec{r}_{<j} = (r_1, \ldots, r_{j-1})$, define

$$R_{j,\vec{\ell}}(X) := \sum_{\vec{u} \in \{0,1\}^{s_{2i+1}-j}} \tilde{\mathsf{lin}}_{2i}\big(\vec{\gamma}_c, \vec{\gamma}_s, (\vec{r}_{<j}, X, \vec{u}), \vec{\ell}\big) \cdot \tilde{L}_{2i+1}\big((\vec{r}_{<j}, X, \vec{u}), \vec{\ell}\big) \tag{19}$$

for every $\vec{\ell} \in \{0,1\}^{\log N}$. For each $t \in \{1, 2\}$, P sends an extended ciphertext whose $\vec{\ell}$th slot encrypts $R_{j,\vec{\ell}}(t)$. After decryption, V computes

$$R_j(t) = \sum_{\vec{\ell} \in \{0,1\}^{\log N}} R_{j,\vec{\ell}}(t) \qquad \text{for } t \in \{1, 2\}.$$

It sets $R_j(0) = c_{j-1} - R_j(1)$, interpolates the quadratic R_j, samples r_j, and sets $c_j = R_j(r_j)$.

2. **Remaining Slot Claim:** After these rounds, let $\vec{r}_c = (r_1, \ldots, r_{s_{2i+1}})$. The remaining claim is

$$c_{s_{2i+1}} \stackrel{?}{=} \sum_{\vec{\ell} \in \{0,1\}^{\log N}} h_{\vec{\ell}} w_{\vec{\ell}}, \tag{20}$$

where

$$h_{\vec{\ell}} := \tilde{\mathsf{lin}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s, \vec{r}_c, \vec{\ell}), \qquad w_{\vec{\ell}} := \tilde{L}_{2i+1}(\vec{r}_c, \vec{\ell}). \tag{21}$$

3. **Terminal Vectors:** P sends the N claimed coefficient evaluations $\mathsf{ept}[h] = (h_{\vec{\ell}})_{\vec{\ell} \in \{0,1\}^{\log N}}$ in plaintext. It also sends $\mathsf{ect}[w]$, which V decrypts to $\mathsf{ept}[w] = (w_{\vec{\ell}})_{\vec{\ell} \in \{0,1\}^{\log N}}$. Both messages are sent before V samples the slot challenge. V checks that the inner product in Eq. (20) equals $c_{s_{2i+1}}$.

4. **Coefficient Openings:** V samples a fresh challenge $\vec{r}_s \in \mathbb{K}^{\log N}$ and records the coefficient claim obtained from $\mathsf{ept}[h]$ as in Appendix C. This claim is authenticated in the final gate-opening batch. V sets

$$v_x := \sum_{\vec{\ell} \in \{0,1\}^{\log N}} \tilde{\mathsf{eq}}(\vec{r}_s, \vec{\ell}) w_{\vec{\ell}}.$$

5. **Reduction:** The reduction outputs the claim

$$v_x \stackrel{?}{=} \tilde{L}_{2i+1}(\vec{r}_c, \vec{r}_s).$$

Figure 11: Specification of an affine layer $2i \to 2i+1$ reduction for the Base hIP.

$k = s_{2i-1} + \log N$. After challenges $\vec{r}_{<j}$, the honest round polynomial in round $j \in \{1, \ldots, k\}$ is

$$h_j(X) = \sum_{\vec{b} \in \{0,1\}^{k-j}} \tilde{\mathsf{eq}}(\vec{\gamma}_{>j}, \vec{b}) \tilde{L}_{2i}(0, \vec{r}_{<j}, X, \vec{b}) \tilde{L}_{2i}(1, \vec{r}_{<j}, X, \vec{b}).$$

The current claim c satisfies $c = (1 - \gamma_j) h_j(0) + \gamma_j h_j(1)$. The omitted-evaluation optimization from Remark 3.2 applies to this weighted check as well. If $\gamma_j \neq 0$, the prover sends $h_j(0), h_j(2)$, and the

Folded Dense Multiplication Layer Reduction

1. **Full Sumcheck:** P and V engage in sumcheck for $s_{2i-1} + \log N$ rounds to prove the claim in Eq. (10), using Gruen's equality optimization from Remark D.1. At the end, V has challenges $\vec{r}_c \in \mathbb{K}^{s_{2i-1}}$ and $\vec{r}_s \in \mathbb{K}^{\log N}$ and a final claim $c \in \mathbb{K}$.
2. **Evaluation Claims:** P sends
$$v_0 \stackrel{?}{=} \tilde{L}_{2i}(0, \vec{r}_c, \vec{r}_s), \qquad v_1 \stackrel{?}{=} \tilde{L}_{2i}(1, \vec{r}_c, \vec{r}_s).$$
3. **Consistency Check:** V checks
$$c \stackrel{?}{=} v_0 v_1. \tag{23}$$
4. **Selector Reduction:** After both values are fixed, V samples $\alpha \in \mathbb{K}$ and forms the single claim
$$(1-\alpha)v_0 + \alpha v_1 \stackrel{?}{=} \tilde{L}_{2i}(\alpha, \vec{r}_c, \vec{r}_s). \tag{24}$$

Figure 12: Specification of a dense multiplication layer $2i-1 \to 2i$ reduction for the Folded hIP.

verifier recovers $h_j(1) = (c - (1-\gamma_j)h_j(0))/\gamma_j$. If $\gamma_j = 0$, the prover instead sends $h_j(1), h_j(2)$, and the verifier recovers $h_j(0) = c$. The three evaluations determine h_j by interpolation, since $\mathbb{K}$ has odd characteristic. After sampling r_j, the verifier continues with claim $h_j(r_j)$. No division by a vanishing equality weight is needed. A false current claim forces the reconstructed quadratic to differ from the honest round polynomial, so the next claim can become true with probability at most $2/|\mathbb{K}|$. After all k rounds, the claim is the product of the two source evaluations, with no equality factor remaining.

In Base, we use the same optimization only in the ciphertext-index rounds. The prover sends two evaluations for each slot, packed as vectors. The verifier aggregates these evaluations using $\tilde{\mathsf{eq}}(\vec{\gamma}_s, \vec{\ell})$ and recovers the omitted evaluation of the aggregate polynomial from its current claim.

Base hIP. In Base, the parties run only the s_{2i-1} ciphertext-index rounds, leaving the N slot instances SIMD-packed. In every round, the prover sends two vectors of evaluations of the per-slot quadratic round polynomials. The verifier performs the remaining slot aggregation after decryption.

D.4 Prover Complexity Analysis

We fix the security parameters and extension degree R throughout this analysis.

Computing affine messages. Consider an affine layer with S_{in} source ciphertexts, S_{out} output ciphertexts, and an underlying computation of g_i affine operations. After fixing the output challenge $(\vec{\gamma}_c, \vec{\gamma}_s)$, the prover needs the public table
$$k_i(\vec{x}_c, \vec{x}_s) := \tilde{\mathsf{lin}}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s, \vec{x}_c, \vec{x}_s)$$
on Boolean source indices. By Eq. (14), this table is obtained by traversing the original affine computation in reverse. This takes $O(N(g_i + S_{\text{in}} + S_{\text{out}}))$ field operations, without expanding the paths through the affine computation.

The affine sumcheck is then the product of the MLE of this public table and the source-wire MLE. In each ciphertext-index round, expand the latter into the original source ciphertexts. Writing

Base Dense Multiplication Layer Reduction

1. **Sumcheck (Early Termination):** P and V engage in sumcheck for s_{2i-1} rounds, using Gruen's equality optimization from Remark D.1. In round j, P sends two evaluations of each per-slot quadratic round polynomial $R_{j,\vec{\ell}}(X)$, packed as vectors. For a fixed $\vec{\ell} \in \{0,1\}^{\log N}$, these are the round polynomials for

$$\tilde{L}_{2i-1}(\vec{\gamma}_c, \vec{\ell}) = \sum_{\vec{x}_c \in \{0,1\}^{s_{2i-1}}} \tilde{\mathsf{eq}}(\vec{\gamma}_c, \vec{x}_c) \tilde{L}_{2i}(0, \vec{x}_c, \vec{\ell}) \tilde{L}_{2i}(1, \vec{x}_c, \vec{\ell}).$$

   V aggregates the received evaluations to obtain two evaluations of

$$R_j(X) = \sum_{\vec{\ell} \in \{0,1\}^{\log N}} \tilde{\mathsf{eq}}(\vec{\gamma}_s, \vec{\ell}) R_{j,\vec{\ell}}(X).$$

   It recovers the omitted evaluation from the current claim as in Remark D.1 and interpolates R_j.

2. **Final Sumcheck Claim:** After the s_{2i-1} rounds, with challenge $\vec{r}_c \in \mathbb{K}^{s_{2i-1}}$, V holds

$$c \stackrel{?}{=} \sum_{\vec{\ell} \in \{0,1\}^{\log N}} \tilde{\mathsf{eq}}(\vec{\gamma}_s, \vec{\ell}) \tilde{L}_{2i}(0, \vec{r}_c, \vec{\ell}) \tilde{L}_{2i}(1, \vec{r}_c, \vec{\ell}). \tag{25}$$

3. **Vector Evaluation Claims:** P sends two vectors $\mathsf{ept}[w_0]$ and $\mathsf{ept}[w_1]$ such that, for every slot $\vec{\ell}$,

$$w_{0,\vec{\ell}} \stackrel{?}{=} \tilde{L}_{2i}(0, \vec{r}_c, \vec{\ell}), \qquad w_{1,\vec{\ell}} \stackrel{?}{=} \tilde{L}_{2i}(1, \vec{r}_c, \vec{\ell}).$$

4. **Consistency Check:** V substitutes $\mathsf{ept}[w_0]$ and $\mathsf{ept}[w_1]$ into the right-hand side of Eq. (25) and checks the result against c.

5. **Slot Reduction:** V samples $\vec{r}_s \in \mathbb{K}^{\log N}$ and computes

$$v_0 \leftarrow \widetilde{w}_0(\vec{r}_s), \qquad v_1 \leftarrow \widetilde{w}_1(\vec{r}_s).$$

6. **Selector Reduction:** After the slot reduction, V samples a fresh $\alpha \in \mathbb{K}$ and forms the single claim

$$(1-\alpha)v_0 + \alpha v_1 \stackrel{?}{=} \tilde{L}_{2i}(\alpha, \vec{r}_c, \vec{r}_s). \tag{26}$$

Figure 13: Specification of a dense multiplication layer $2i-1 \to 2i$ reduction for the Base hIP.

$\vec{\alpha} = (\vec{r}_{<j}, t)$, the slot values in Eq. (19) become

$$R_{j,\vec{\ell}}(t) = \sum_{\vec{b} \in \{0,1\}^j,\, \vec{u} \in \{0,1\}^{s_{2i+1}-j}} \tilde{\mathsf{eq}}(\vec{\alpha}, \vec{b})\, \widetilde{k}_i(\vec{\alpha}, \vec{u}, \vec{\ell}) \cdot L_{2i+1}(\vec{b}, \vec{u}, \vec{\ell}).$$

All interpolation factors and entries of the partially evaluated coefficient table are public, so their products can be computed before applying them as plaintext masks. For each evaluation of the round polynomial, each original source ciphertext is multiplied by one extended plaintext mask and the results are added. Thus each round uses $O(S_{\text{in}})$ FHE operations for fixed R, at one plaintext-ciphertext multiplication level. The slot values of the public masks take $O(NS_{\text{in}})$ field operations per round to compute. Encoding them as plaintexts adds at most a logarithmic factor in N. The same observation applies to the terminal source vector. In particular, computing the public coefficients

does not add FHE depth, and the interpolations need not be applied to the ciphertexts in separate multiplication stages.

The coefficient and constant requests of each affine reduction share one output point, so one set of weights suffices for Theorem C.1. The Folded prover uses the same table through the remaining slot-index rounds. Across all layers, the affine computations and their input and output wires have total size $O(n_{\mathsf{FHE}})$. The shared plaintext-mask dictionary of size D_{mask} is read for each layer, as accounted for in Appendix C. Hence the public sumcheck arithmetic and affine coefficient openings together use $\widetilde{O}(Nn_{\mathsf{FHE}} + dD_{\mathrm{mask}})$ field operations. Since $D_{\mathrm{mask}} = O(Nn_{\mathsf{FHE}})$, this is $\widetilde{O}(Nn_{\mathsf{FHE}})$ for bounded depth. The public index is constructed once in preprocessing using $\widetilde{O}(Nn_{\mathsf{FHE}})$ field operations.

Proof of Theorem 5.4. Padding and the dense left/right layout enlarge the logical trace by at most a constant factor. Across all layer reductions, each encrypted wire value or alias contributes to $O(\log n_{\mathsf{FHE}})$ sumcheck messages. For affine reductions, the computation above uses $O(S_{\mathrm{in}} \log S_{\mathrm{in}})$ FHE operations, independently of the support of the composed affine coefficients. For multiplication reductions, each round forms the two operand interpolations using scalar-ciphertext multiplications and additions, then multiplies the paired interpolated operands. Each round uses work linear in the source width. Summing over the trace gives $O(n_{\mathsf{FHE}} \log n_{\mathsf{FHE}})$ FHE operations. An affine layer reduction consumes one level from the deepest payload ciphertexts, while a multiplication layer reduction consumes 1.5 levels but starts from ciphertexts at least one payload level earlier. These bounds include the terminal source evaluations. □

Proof of Theorem 5.5. As in the Base prover, the dense trace has linear logical size and the sumcheck arithmetic contributes $O(n_{\mathsf{FHE}} \log n_{\mathsf{FHE}})$ FHE operations. The Folded prover additionally applies $O(dR \log(Nn_{\mathsf{FHE}}))$ slot-folding operations, each implemented with $O(\log N)$ slot automorphisms and additions. In the ciphertext-index rounds, slot weighting adds at most one plaintext-ciphertext multiplication level to the Base computations, giving a bound of 2.5 levels from the source ciphertexts.

After the ciphertext-index rounds, the prover fixes their challenges in the source-wire MLEs and stores the resulting encrypted slot vectors as a checkpoint. Forming the checkpoint uses scalar-ciphertext multiplications and additions, consuming 0.5 levels. Each slot-index round starts from this same checkpoint. Evaluating the slot challenges sampled so far uses one plaintext-ciphertext multiplication level, followed by slot automorphisms and additions. Applying the public coefficient or equality weights uses one further plaintext-ciphertext multiplication level. An affine slot-index round therefore uses at most $0.5 + 1 + 1 = 2.5$ levels from its source ciphertexts. A multiplication slot-index round evaluates both operands in parallel and then multiplies them, using at most 3.5 levels. These operands start at least one payload multiplication level earlier, so both reductions use at most 2.5 levels beyond the payload evaluation. Terminal evaluations use the same checkpoint and one level for evaluating the slot variables, and fit within this bound. The folding operations use only slot automorphisms and additions, so they add no multiplicative depth. The public slot interpolations and weights contribute at most a logarithmic factor to the field work above, giving $\widetilde{O}(Nn_{\mathsf{FHE}} + dD_{\mathrm{mask}})$ field operations including the affine coefficient openings. □

E Specialized BhIP for Laned Payload Circuits

E.1 Overview and Prerequisites

This appendix gives a specialized BhIP for payload circuits consisting of independent SIMD lanes, including the single-slot payloads considered in prior work. When the number of lanes is small, witness packing reduces the number of ciphertexts on which the BhIP runs.

Definition E.1 (B-Laned Payload Circuit)**.** A payload circuit is *B-laned* (where B is a power of two dividing N) if: (i) it evaluates $B \leq N$ logical SIMD lanes, (ii) it involves no cross-slot interactions (no slot automorphisms), and (iii) all B lanes execute the same single-slot arithmetic circuit independently, in SIMD parallel.

The specialized BhIP. The wide BhIP from Section 5 could be applied directly to a B-laned trace, with its N/B copies of each lane value. However, we can do better by exploiting the independence of the lanes. We first use *witness packing* to combine the intermediate assignments in fewer ciphertexts. The Laned BhIP then follows the Folded protocol on this packed trace, using the FOLD operator from Section 5.1.3 to aggregate its round messages across slots. We give the resulting prover bounds in Theorem E.3. Applying transcript packing after the BhIP gives $\mathsf{Laminate}_{\mathsf{laned}}$, as described in Theorem 6.2.

E.1.1 Payload Representation

During payload evaluation, each ciphertext stores N/B copies of its B logical lane values across all N SIMD slots. Concretely, a ciphertext encoding $(x_0, \ldots, x_{B-1}) \in \mathbb{F}_q^B$ stores

$$(\underbrace{x_0, \ldots, x_{B-1}}_{B}, \underbrace{x_0, \ldots, x_{B-1}}_{B}, \ldots, \underbrace{x_0, \ldots, x_{B-1}}_{B}).$$

The supported operations are CT-CT addition, scalar-CT addition, scalar-CT multiplication, and CT-CT multiplication. All of these act slot-wise and preserve this B-periodic structure.

E.1.2 Dense Alternating Layered Arithmetization

We use the dense alternating arithmetization and notation from Sections 5.1.1 and 5.1.2. The circuit has depth $d = 2d_\times + 1$, with output layer L_0 and input layer L_d. Non-input odd layers contain SIMD multiplication gates, and even layers contain affine gates. Here the affine gates use scalar coefficients and no slot automorphisms. The multiplicative depth $d_\times$ counts CT-CT multiplications in the payload.

The dense wiring convention gives $S_{2i} = 2S_{2i-1}$, with a left operand half followed by a right operand half. The trace has $S_d + S_0 + 3m$ logical ciphertext positions, where $m = \sum_{i=1}^{d_\times} S_{2i-1} \leq 2n_{\mathsf{FHE}}$. These are ciphertexts already computed during payload evaluation, their aliases, and public zeros. Witness packing therefore requires no further evaluation of the payload circuit.

E.2 Witness Packing

After evaluating the payload circuit, the server packs every assignment layer before beginning the BhIP. The packing masks depend only on the data layout and are independent of the proof transcript.

E.2.1 Packing Procedure

Write $K := N/B$ for the number of original ciphertexts that fit in one packed ciphertext. For a group of K ciphertexts, the jth ciphertext ($j \in \{0, \ldots, K-1\}$) is multiplied by a plaintext mask that preserves positions jB through $(j+1)B - 1$ and zeroes out all other slots. Since the original ciphertexts are B-periodic, the target chunk already contains the correct values. The masked ciphertexts are summed to produce one packed ciphertext, without slot automorphisms.

Padding the dense layout. We pack the left and right operand halves of each even operand layer separately, padding both halves in the same way. For a multiplication layer $2i-1$, define

$$H_{2i-1} := K\lceil S_{2i-1}/K \rceil, \qquad H_{2i} := 2H_{2i-1}.$$

Gate $j < S_{2i-1}$ remains at position j, and the remaining multiplication gates are public zeros. Its left operand remains at position j, while its right operand moves from position $S_{2i-1}+j$ to position $H_{2i-1}+j$. The input and output layers are padded at the end to widths $H_d := K\lceil S_d/K \rceil$ and $H_0 := K\lceil S_0/K \rceil$. The affine coefficient tables and constant terms use these same public maps on their source and output indices, with zero entries at padding positions. This preserves the circuit computation and the dense left/right layout, including when a multiplication layer has fewer than K gates.

After packing, layer i has $S'_i := H_i/K$ ciphertexts. These widths are powers of two under the convention from Section 5.1.2, and $S'_{2i} = 2S'_{2i-1}$. Packing both operand halves separately leaves at most two partially filled ciphertexts per assignment layer.

E.2.2 Data Layout After Packing

Each packed ciphertext contains K chunks of B slots. We index a packed wire value by $(\vec{x}_c, \vec{x}_q, \vec{x}_\ell)$, where

- $\vec{x}_c \in \{0,1\}^{s'_i}$ is the packed ciphertext index, with $s'_i := \log S'_i$.
- $\vec{x}_q \in \{0,1\}^{\log K}$ is the chunk index.
- $\vec{x}_\ell \in \{0,1\}^{\log B}$ is the lane within the chunk.

The slot index is $\vec{x}_s = (\vec{x}_q, \vec{x}_\ell)$, and the full wire index has $t'_i := s'_i + \log N$ bits. The concatenation $(\vec{x}_c, \vec{x}_q)$ identifies a ciphertext position in the padded layout of width H_i. We use L_i and $\tilde{L}_i$ for the packed assignment and its MLE from this point onward.

E.2.3 Witness Packing Cost

Witness packing uses at most one base-field PT-CT multiplication for masking and one CT-CT addition per original ciphertext position. Public-zero positions require no FHE operations. Across all layers, packing therefore uses $O(S_d + S_0 + n_{\mathsf{FHE}})$ FHE operations and produces $O((S_d + S_0 + n_{\mathsf{FHE}})B/N + d)$ ciphertexts. This packing requires at most one multiplicative level beyond the payload evaluation.

E.3 Layer Reductions

The prover and verifier run the Folded protocol on the packed trace, working from the output layer to the input layer through d reductions. For affine layers, they use the slot-independent coefficients of the laned payload, and for multiplication layers, the same dense layer identity as in Section 5.2. In

both cases, they process the packed ciphertext-index variables first, followed by the chunk variables and then the lane variables. The prover uses FOLD to aggregate across slots, so each transmitted field element is replicated in every slot. After each reduction, the verifier has one evaluation claim for the next layer.

E.3.1 Affine Layer Reduction

Fix an even layer $2i$ for $i \in \{0, \ldots, d_\times\}$. Each affine gate computes the same scalar affine map in all B lanes. Write $\vec{x}_u := (\vec{x}_c, \vec{x}_q)$ for a source position in the padded layout, and $\vec{z}_u := (\vec{z}_c, \vec{z}_q)$ for an output position. Let $\mathsf{lin}^{\mathsf{ct}}_{2i} : \{0,1\}^{s'_{2i}+\log K} \times \{0,1\}^{s'_{2i+1}+\log K} \to \mathbb{F}_q$ give the scalar coefficients after the public padding and reindexing from Appendix E.2.1. Let $\mathsf{const}^{\mathsf{ct}}_{2i} : \{0,1\}^{s'_{2i}+\log K} \to \mathbb{F}_q$ give the additive constants.

Affine coefficient polynomials. The packed coefficient function factors as

$$\mathsf{lin}^{\mathsf{pack}}_{2i}(\vec{z}_u, \vec{z}_\ell, \vec{x}_u, \vec{x}_\ell) = \mathsf{lin}^{\mathsf{ct}}_{2i}(\vec{z}_u, \vec{x}_u)\mathbf{1}[\vec{z}_\ell = \vec{x}_\ell]. \tag{27}$$

The two factors use disjoint variables, so the MLE factors as

$$\tilde{\mathsf{lin}}^{\mathsf{pack}}_{2i}(\vec{Z}_u, \vec{Z}_\ell, \vec{X}_u, \vec{X}_\ell) = \tilde{\mathsf{lin}}^{\mathsf{ct}}_{2i}(\vec{Z}_u, \vec{X}_u)\tilde{\mathsf{eq}}(\vec{Z}_\ell, \vec{X}_\ell). \tag{28}$$

We treat $\tilde{\mathsf{lin}}^{\mathsf{pack}}_{2i}$ as a virtual polynomial, committing only to $\tilde{\mathsf{lin}}^{\mathsf{ct}}_{2i}$ and $\tilde{\mathsf{const}}^{\mathsf{ct}}_{2i}$ during preprocessing. The terminal coefficient check opens $\tilde{\mathsf{lin}}^{\mathsf{ct}}_{2i}$, and the verifier computes the public lane-equality factor locally.

Affine layer identity. At a challenge $(\vec{\gamma}_c, \vec{\gamma}_s) \in \mathbb{K}^{s'_{2i}+\log N}$, write $\vec{\gamma}_s = (\vec{\gamma}_q, \vec{\gamma}_\ell)$ and $\vec{\gamma}_u = (\vec{\gamma}_c, \vec{\gamma}_q)$. To form the affine messages, we include the Boolean lane-equality weights in the plaintext masks applied to the source layer. For Boolean source positions, define

$$H_{\vec{\gamma}_\ell}(\vec{x}_u, \vec{x}_\ell) := \tilde{\mathsf{eq}}(\vec{\gamma}_\ell, \vec{x}_\ell)L_{2i+1}(\vec{x}_c, \vec{x}_q, \vec{x}_\ell).$$

Here $\vec{x}_c \in \{0,1\}^{s'_{2i+1}}$, $\vec{x}_q \in \{0,1\}^{\log K}$, $\vec{x}_\ell \in \{0,1\}^{\log B}$, and $\vec{x}_u = (\vec{x}_c, \vec{x}_q)$. Let $\widetilde{H}_{\vec{\gamma}_\ell}$ be this table's MLE. The layer identity is

$$\tilde{L}_{2i}(\vec{\gamma}_c, \vec{\gamma}_s) - \tilde{\mathsf{const}}^{\mathsf{ct}}_{2i}(\vec{\gamma}_u) = \sum_{\vec{x}_c, \vec{x}_q, \vec{x}_\ell} \tilde{\mathsf{lin}}^{\mathsf{ct}}_{2i}(\vec{\gamma}_u, \vec{x}_u)\widetilde{H}_{\vec{\gamma}_\ell}(\vec{x}_u, \vec{x}_\ell). \tag{29}$$

The summand has individual degree at most two, and its coefficient factor is independent of the lane variables. The terminal claim concerns $\widetilde{H}_{\vec{\gamma}_\ell}$, which generally differs from the product of the lane-equality polynomial and the source-layer MLE. By normalizing the Boolean interpolation weights, we recover a source-layer evaluation at a different lane point.

For each $j \in [\log B]$, set $a_j = (1 - \gamma_{\ell,j})(1 - r_{\ell,j})$, $b_j = \gamma_{\ell,j} r_{\ell,j}$, $c_j = a_j + b_j$, and $s_{\ell,j} = b_j / c_j$ when $c_j \neq 0$. Expanding the Boolean interpolation weights gives

$$\widetilde{H}_{\vec{\gamma}_\ell}(\vec{r}_u, \vec{r}_\ell) = \left(\prod_{j \in [\log B]} c_j \right) \widetilde{L}_{2i+1}(\vec{r}_u, \vec{s}_\ell). \tag{30}$$

The challenge for lane coordinate j is sampled uniformly subject to $c_j \neq 0$. There is at most one forbidden value, since c_j is affine in that coordinate and is never identically zero. For $B = 1$, the product is empty and equals one.

Affine Layer Reduction (Laned Payload)

1. **Full Sumcheck:** P supplies $a \stackrel{?}{=} \widetilde{\mathsf{const}}^{\mathsf{ct}}_{2i}(\vec{\gamma}_u)$. V records this value for the final gate-opening batch and subtracts it from the current claim. The parties run $s'_{2i+1} + \log N$ sumcheck rounds for Eq. (29). Each round polynomial has degree at most two. Ciphertext and chunk challenges are uniform in $\mathbb{K}$, and lane challenges use the distribution above. The terminal point is $(\vec{r}_c, \vec{r}_q, \vec{r}_\ell)$, with claim $c \in \mathbb{K}$.
2. **Terminal Claims:** P supplies $g \stackrel{?}{=} \widetilde{\mathsf{lin}}^{\mathsf{ct}}_{2i}(\vec{\gamma}_u, \vec{r}_u)$ and $h \stackrel{?}{=} \widetilde{H}_{\vec{\gamma}_\ell}(\vec{r}_u, \vec{r}_\ell)$, where $\vec{r}_u = (\vec{r}_c, \vec{r}_q)$. V records g with its layer and points for the final gate-opening batch.
3. **Consistency Check:** V checks
$$c \stackrel{?}{=} gh. \tag{31}$$
4. **Reduction:** The next claim is $h / \prod_{j \in [\log B]} c_j \stackrel{?}{=} \widetilde{L}_{2i+1}(\vec{r}_u, \vec{s}_\ell)$.

Figure 14: Specification of an affine layer $2i \to 2i+1$ reduction for the laned hIP.

The recorded constant and coefficient claims are opened after GKR against their honestly preprocessed public tables.

Coefficient opening cost. Let Q_{ct} and Q_{const} be the total numbers of nonzero entries in the scalar coefficient and constant tables across all affine layers. Public padding and reindexing preserve these counts. The virtual representation avoids the B repeated intra-chunk copies that a direct commitment to the packed coefficient tables would contain. With a support-efficient PCS as defined in Appendix B.1, all affine coefficient opening proofs take $\widetilde{O}(Q_{\mathsf{ct}} + Q_{\mathsf{const}})$ field operations. When $Q_{\mathsf{ct}} + Q_{\mathsf{const}} = O(n_{\mathsf{FHE}})$, these opening proofs take $\widetilde{O}(n_{\mathsf{FHE}})$ field operations. This public opening work is separate from the FHE operation counts below.

Lemma E.2 (Laned gate-opening soundness)**.** *Suppose preprocessing honestly commits the sparse address and value columns for the scalar coefficient and constant tables. For each committed table, all claimed column values and their evaluation points are fixed before column batching. The verifier checks every public operand in the terminal equations, including equality, selection, and column-combination weights. Assume the PCS satisfies the fixed-polynomial and RBR-opening requirements of Definition G.2, with opening error at most $\epsilon_{\mathsf{PCS,rbr}}$. It checks each opening against these columns or their prescribed linear combination. When the opening protocol batches its input claims, such as column evaluations, let d_{gate} bound the total degree in each challenge block, including its column and operand weights. Assuming a good setup, the gate-opening IOP has RBR error at most*

$$\epsilon_{\mathsf{gate,rbr}} \le \max\left\{ \frac{\max\{3, d_{\mathsf{gate}}\}}{|\mathbb{K}|}, \epsilon_{\mathsf{PCS,rbr}} \right\}.$$

Proof. For a fixed sparse table, its evaluation is a weighted sum of its stored values, with equality weights determined by their addresses. We check each product of a stored value and its address-equality factors through a binary multiplication tree, keeping the degree independent of the address length. The sumcheck identity is the same as in Fig. 12: two multilinear child tables are multiplied by a multilinear equality factor, giving individual degree at most three. At the leaves, the remaining claims are affine combinations of the committed address and value columns. Thus a false identity leaves a false sumcheck or column-evaluation claim, except with probability at most $3/|\mathbb{K}|$ at each fresh scalar challenge. The selector interpolation has degree one and satisfies the same bound. Batching contributes at most $d_{\mathsf{gate}}/|\mathbb{K}|$, since the column values and their evaluation points are

fixed before its challenges. Any remaining false evaluation of the prescribed column combination starts the PCS argument in a bad state. All public operands are checked by the verifier. Following a false claim until it is checked gives the displayed maximum, with separate fresh challenges for different committed tables. □

E.3.2 Multiplication Layer Reduction

Fix an odd layer $2i-1$ for $i \in \{1, \ldots, d_\times\}$. The packed layout preserves $S'_{2i} = 2S'_{2i-1}$. The left and right operands share a ciphertext-index suffix and slot index, differing only in the leading selector bit. Consequently, the packed trace satisfies the same multiplication identity as Eq. (22).

We apply the Folded reduction from Fig. 12 with s_{2i-1} replaced by s'_{2i-1}. This gives $s'_{2i-1}+\log N$ sumcheck rounds, each sending two elements of $\mathbb{K}$, encoded in $2R$ ciphertexts after folding.

E.4 Prover Complexity

The laned prover first packs the encrypted computation trace and then runs the BhIP on the smaller packed trace. The subsequent BhIP work shrinks with the ratio B/N of logical lanes to physical SIMD slots.

Theorem E.3 (Laned BhIP prover complexity)**.** *Let* Ckt *be a dense alternating layered* B*-laned payload circuit of depth* d *computed with* n_{FHE} *FHE operations, where* $B \le N$*. Suppose that* $S_0+S_d = O(n_{\mathsf{FHE}})$*. Suppose that multiplication in* $\mathbb{K}$ *can be computed by a multiplicative-depth-*1 *circuit over* $\mathbb{F}_q$*. For constant extension degree* R*, the BhIP prover, including witness packing, performs*

$$O\left(n_{\mathsf{FHE}} + \left(\frac{n_{\mathsf{FHE}}B}{N} + d\right)\log n_{\mathsf{FHE}} + d\log(Nn_{\mathsf{FHE}})\log N\right) = \widetilde{O}(n_{\mathsf{FHE}})$$

FHE operations and requires at most 3.5 *multiplicative levels beyond the payload evaluation.*

Public coefficient-opening work is bounded separately in Appendix E.3.1.

Justification. The bounds in Appendix E.2.3 give the packing work and packed trace size. As in the Folded analysis in Appendix D.4, each packed ciphertext contributes to $O(\log n_{\mathsf{FHE}})$ round-message computations. The remaining term accounts for folding the round messages across slots.

The BhIP follows the Folded checkpoint schedule from Appendix D.4, using at most $0.5+1+1 = 2.5$ levels after witness packing. In an affine reduction, the source-interpolation masks include the Boolean lane-equality weights from $H_{\vec{\gamma}_\ell}$ in Eq. (29). Including these weights before interpolation gives the required table without an extra multiplication level and leaves the final multiplication for the public coefficient weights. The total overhead is therefore at most $1+0.5+1+1 = 3.5$ levels, including witness packing. A multiplication reduction additionally multiplies two source values, but they start one payload level earlier and meet the same bound. The ciphertext-index rounds and terminal evaluations fit within this budget, as in the Folded analysis. Computing the two evaluations of the quadratic round polynomial uses the same bilinear products and public weights, so the equality optimization requires no additional levels. □

F Transcript Packing

We give the construction of the transcript packing compiler from Section 6.1, followed by its well-formedness tests and their guarantees. Appendix G uses these guarantees to prove integrity of the VCoED protocols.

F.1 Protocol and Packing Construction

The initial transcript contains the public parameters, payload index, encrypted request, and returned encrypted output. The protocol proceeds as follows, with steps 3–5 applying only to the packed variants, Folded and Laned.

1. During GKR, the prover sends either a ciphertext message or an aligned-ciphertext commitment before receiving the next verifier challenge.
2. After the last GKR message, both parties authenticate the evaluation claims about the MLEs of the affine gate wiring and constants.
3. The verifier samples the well-formedness coefficients. The prover then sends the digit and test ciphertexts.
4. The prover commits to its packing witness as a multilinear polynomial, and both parties run interactive Spartan on the packing statement.
5. Both parties authenticate the resulting evaluation claim about the packing witness.

All verifier challenges are public coins, so the client can defer decrypting and checking the GKR messages until the interaction ends. For the packed variants, it also checks the well-formedness tests at that point. We apply Fiat–Shamir to this whole interaction. Each challenge is derived from a domain-separated hash of the full preceding transcript, using the same sampling rule as the interactive verifier.

F.1.1 Compiler Description

The compiler acts on the r rounds of the BhIP, with public zero padding to M ciphertexts when needed. All public auxiliary messages and original verifier challenges are transmitted unchanged in their original order.

Setup assigns consecutive rounds to disjoint blocks of M consecutive logical slots, filling each logical packed output with N/M rounds before starting the next. A fixed public permutation maps logical slots to physical SIMD slots. For every output $\ell \in [\lceil Mr/N \rceil]$, let $\mathcal{I}_\ell$ be its set of rounds. For every round $i \in [r]$ and message $j \in [M]$, let $\mu_{i,j}$ be the plaintext mask selecting that message's destination slot, and let p_i select the whole block for round i.

Setup decomposes each mask p_i into D public signed-digit masks $p_i^{\langle d \rangle}$ for $d \in [D]$, as specified in Eq. (37). Setup fixes the final ciphertext modulus Q and test threshold τ according to Corollary F.4 and Eq. (51). The digit decomposition and packing costs are detailed below, and the parameter bounds appear in Appendix F.3.

Abusing notation slightly, $H(\mathsf{ct})$ denotes hashing a fixed, unambiguous representation of the ciphertext in the input domain of H.

1. **Round commitment.** In round $i \in [r]$, let $\mathsf{ct}_{\mathrm{raw}}^{(i,j)}$ be the jth BhIP ciphertext, whose slots all contain the same value. The prover aligns the M messages into their assigned slots:

$$\mathsf{ct}_{\mathrm{align},i} := \sum_{j \in [M]} \mu_{i,j} \mathsf{ct}_{\mathrm{raw}}^{(i,j)}. \tag{32}$$

   It switches this ciphertext to modulus Q, preserving its plaintext, and sends $C_i := H(\mathsf{ct}_{\mathrm{align},i})$. The prover and verifier then continue the original interaction, including any challenge following the last BhIP round.

2. **Well-formedness challenge.** After the original interaction is complete and all round commitments are fixed, the verifier samples and sends independent coefficients $x_{a,i} \in \{-1, 0, 1\}$ for every $(a, i) \in [T] \times [r]$. Each coefficient is 0 with probability 1/2, and each of -1 and 1 with probability 1/4.

3. **Packed transcript.** For every output $\ell \in [\lceil Mr/N \rceil]$ and digit $d \in [D]$, the prover computes

$$\mathsf{ct}_{\mathrm{pack},\ell}^{\langle d \rangle} := \sum_{i \in \mathcal{I}_\ell} p_i^{\langle d \rangle} \mathsf{ct}_{\mathrm{align},i} \in \mathcal{R}_Q^2. \tag{33}$$

   It also computes, for every $a \in [T]$,

$$\mathsf{ct}_a^{\mathsf{test}} := \sum_{i \in [r]} x_{a,i} \mathsf{ct}_{\mathrm{align},i} \in \mathcal{R}_Q^2. \tag{34}$$

   The prover sends the digit and test ciphertexts. The ordered commitments, test coefficients, revealed ciphertexts, and setup description fix the public packing statement.

4. **Interactive packing proof.** The prover commits to a multilinear encoding of the aligned ciphertexts and auxiliary arithmetic values. The parties then run interactive Spartan, including its polynomial openings, to show that this ciphertext list opens $(C_i)_{i \in [r]}$ and produces the revealed digit and test ciphertexts.

5. **Verifier decision.** The verifier checks the interactive packing transcript and uses its secret key to check

$$\left\| [\mathsf{L}_{\mathsf{sk}}(\mathsf{ct}_a^{\mathsf{test}})]_Q \right\|_\infty \le \tau \tag{35}$$

   for every $a \in [T]$. For each logical output, it decrypts the D digit ciphertexts separately and recomposes the plaintext with the powers $1, \zeta, \ldots, \zeta^{D-1}$. It reads the assigned slots, discards any padding, and runs the underlying plaintext verifier on the recovered values, unchanged public messages, and original challenges.

F.1.2 The Packing Argument

The public input to the packing relation $\mathcal{R}_{\mathsf{pack}}$ consists of the ordered round commitments, well-formedness coefficients, and revealed digit and test ciphertexts. Given these values, the prover commits to a witness containing the aligned ciphertexts and the auxiliary values for the hash circuit and packing equations, before receiving Spartan's first challenge. The constraints require this ciphertext list to open the round commitments and produce the revealed ciphertexts. They check the representations used in these computations, including the ciphertext representation supplied to H and any Boolean representation of the test coefficients.

The packing prover works directly with the visible ciphertext coefficients outside FHE. It uses a base field $\mathbb{F}$ and a challenge field $\mathbb{K}_{\mathsf{pack}} \supseteq \mathbb{F}$. We use the same prime for the final ciphertext modulus and the packing base field (Appendix H.2), so ordinary field operations perform the required modular arithmetic. Other choices require explicit constraints for arithmetic modulo Q. In either case, the hash and the packing equations must use the same coefficient representation.

Spartan and public-index checks. Using Spartan, the parties reduce the claim that the assignment satisfies $\mathcal{R}_{\mathsf{pack}}$ to evaluation claims about the witness and the R1CS matrices A, B, C [69]. The verifier checks these evaluations without expanding the matrices. It computes the regular wiring directly and uses PCS openings to check the contributions from digit-mask coefficients against tables

committed during setup. The prover supplies each table's contribution before the checking challenges, and the verifier checks that these contributions give the claimed matrix evaluation. Using sumcheck, the parties reduce these claims to openings of the table columns and evaluations that the verifier computes from the public layout and Spartan's challenges. For each PCS opening, the verifier computes and checks the public weights, including the equality weights for the evaluation point and any coefficients combining table columns. In Spartan, the public and witness halves of the assignment are combined using a random selector. To recover the witness evaluation, the verifier subtracts the public contribution and divides by this selector, so it rejects when the selector is zero. With a uniform selector and perfectly complete PCS openings, this gives completeness error $\delta_{\mathsf{alg,cmp}} = 1/|\mathbb{K}_{\mathsf{pack}}|$.

F.1.3 Packing Layout and Cost

Each round has $M \mid N$ ciphertexts, each encrypting one value replicated across all slots. Under the block assignment above, each logical output contains at most N/M rounds, with

$$|\mathcal{I}_\ell| \le \frac{N}{M}, \qquad r_{\max} := \max_{\ell \in [\lceil Mr/N \rceil]} |\mathcal{I}_\ell| = \min\left\{ r, \frac{N}{M} \right\}. \tag{36}$$

The public permutation can be chosen to reduce mask coefficient norms. The artifact documents the slot-index formulas and implemented placement.

Alignment and commitment. The mask $\mu_{i,j} \in \mathcal{R}_t$ selects message j's assigned slot in round i. Multiplying each raw ciphertext by this mask and summing gives Eq. (32) without rotations. Across all rounds, alignment uses Mr plaintext-ciphertext multiplications and $(M-1)r$ additions, consuming one level because the multiplications run in parallel. The prover then switches to the final modulus Q and commits to the aligned ciphertext. Every removed coefficient-modulus factor is congruent to 1 modulo t, preserving the BGV plaintext representation under the assumed correctness bound. All subsequent packing and test operations remain in $\mathcal{R}_Q^2$, without further modulus switching, relinearization, or rotations.

Digit packing. Let $p_i \in \mathcal{R}_t$ select round i's M destination slots, and use the same symbol for its centered lift in $\mathcal{R}$. At the plaintext level, this discards any values a malicious prover may have placed outside the assigned block. A centered coefficient of p_i can have magnitude almost $t/2$, so using the full mask can greatly increase ciphertext noise. For digit count $D \ge 1$, choose the smallest odd $\zeta \ge 3$ with $\zeta^D \ge t$. Then $\zeta = O(t^{1/D})$, and the public signed-digit decomposition satisfies

$$p_i = \sum_{d \in [D]} \zeta^{d-1} p_i^{\langle d \rangle} \quad \text{in } \mathcal{R}, \qquad \left\| p_i^{\langle d \rangle} \right\|_\infty \le \frac{\zeta - 1}{2} \tag{37}$$

for $i \in [r]$ and $d \in [D]$. Equation (33) uses at most Dr plaintext-ciphertext multiplications and $D(r - \lceil Mr/N \rceil)$ additions, consuming one more level and producing $D\lceil Mr/N \rceil$ ciphertexts. The verifier decrypts the digits separately and applies $1, \zeta, \ldots, \zeta^{D-1}$ in $\mathcal{R}_t$, so radix powers do not amplify ciphertext noise. The well-formedness tests add T ciphertexts and use only additions and negations.

F.2 Well-Formedness and Digit Packing

The signed-digit masks from Section 6.1 keep the public plaintext multipliers small. Correct packing also requires a bound on the aligned ciphertexts' partial decryptions. We express this bound using coefficient columns, then give a randomized test for it.

Definition F.1 (Partial-decryption coefficient columns)**.** For the analysis, write the centered partial decryption of each ciphertext in the list $\mathsf{ct}_1, \ldots, \mathsf{ct}_r \in \mathcal{R}_Q^2$ as

$$y_u := [\mathsf{L}_{\mathsf{sk}}(\mathsf{ct}_u)]_Q = \sum_{j\in[N]} y_{u,j} X^{j-1} \in \mathcal{R}, \qquad y_{u,j} \in \mathbb{Z} \quad (u \in [r]). \tag{38}$$

The matrix $Y := (y_{u,j}) \in \mathbb{Z}^{r\times N}$ has one row per ciphertext. Its *coefficient columns* are

$$\boldsymbol{y}^{(j)} := (y_{u,j})_{u\in[r]} \in \mathbb{Z}^r \qquad (j \in [N]). \tag{39}$$

For every operand $u \in [r]$, let $\mu_u := y_u \bmod t \in \mathcal{R}_t$ be its encoded plaintext and let $m_u := [\mu_u]_t \in \mathcal{R}$ be the centered coefficient lift. There is a unique noise polynomial $e_u \in \mathcal{R}$ such that

$$y_u = m_u + te_u \in \mathcal{R}, \qquad m_u, e_u \in \mathcal{R}. \tag{40}$$

Fix a public coefficient bound $P > 0$. Because $\|m_u\|_\infty \le (t-1)/2$, the condition

$$t\,\|e_u\|_\infty + \frac{t-1}{2} \le P \qquad \text{for every } u \in [r] \tag{41}$$

implies $|y_{u,j}| \le P$ for every $u \in [r]$ and $j \in [N]$. It follows that

$$\left\|\boldsymbol{y}^{(j)}\right\|_2 \le P\sqrt{r} \qquad \text{for every } j \in [N]. \tag{42}$$

Plaintext recomposition. If each digit ciphertext for output $\ell \in [\lceil Mr/N\rceil]$ decrypts to

$$\mu_\ell^{\langle d\rangle} := \sum_{u\in\mathcal{I}_\ell} p_u^{\langle d\rangle}\mu_u \quad \text{in } \mathcal{R}_t \qquad (d \in [D]), \tag{43}$$

then, by the signed-digit decomposition,

$$\sum_{d\in[D]} \zeta^{d-1}\mu_\ell^{\langle d\rangle} = \sum_{u\in\mathcal{I}_\ell} p_u\mu_u \quad \text{in } \mathcal{R}_t. \tag{44}$$

Sufficient condition for correct digit packing. A uniform bound on the column norms prevents coefficient wraparound during digit packing.

For every output $\ell \in [\lceil Mr/N\rceil]$ and digit $d \in [D]$, define the unreduced integer partial decryption

$$\widetilde{y}_\ell^{\langle d\rangle} := \sum_{u\in\mathcal{I}_\ell} p_u^{\langle d\rangle} y_u \in \mathcal{R}. \tag{45}$$

Reducing this polynomial modulo Q gives the partial decryption of $\mathsf{ct}_{\mathrm{pack},\ell}^{\langle d\rangle}$. Hence, it is enough to keep every coefficient of $\widetilde{y}_\ell^{\langle d\rangle}$ below $Q/2$ in absolute value.

For every $(u, d, h) \in [r] \times [D] \times [N]$, write $p_{u,h}^{\langle d\rangle} \in \mathbb{Z}$ for coefficient h of the public digit mask $p_u^{\langle d\rangle}$. Define the *packing weight*

$$W := \max_{\ell\in[\lceil Mr/N\rceil], d\in[D], h\in[N]} \left\|(p_{u,h}^{\langle d\rangle})_{u\in\mathcal{I}_\ell}\right\|_2. \tag{46}$$

Thus, W is the largest norm of a public mask-coefficient vector across all outputs, digits, and coefficient offsets. Every entry has magnitude less than $\zeta/2$, and every output contains at most $r_{\max}$ operands. Therefore,

$$W \le \frac{\zeta}{2}\sqrt{r_{\max}}. \tag{47}$$

Summing the actual mask norms at each offset gives a tighter bound than using W at every offset. Define

$$W_\Sigma := \max_{\ell\in[\lceil Mr/N\rceil],d\in[D]} \sum_{h\in[N]} \left\| (p_{u,h}^{\langle d\rangle})_{u\in\mathcal{I}_\ell} \right\|_2 \le NW. \tag{48}$$

This accounts for the small or zero coefficients in the encoded masks, which is useful for the concrete parameter choices in Appendix H.2.

Lemma F.2 (Coefficient-column bounds imply correct BGV digit packing)**.** *Let $B_{\mathsf{col}} \ge 0$, and suppose $\left\|\boldsymbol{y}^{(j)}\right\|_2 \le B_{\mathsf{col}}$ for every $j \in [N]$. Then*

$$\left|\widetilde{y}_{\ell,j}^{\langle d\rangle}\right| \le B_{\mathsf{col}} W_\Sigma \le B_{\mathsf{col}} WN \tag{49}$$

for every $\ell \in [\lceil Mr/N\rceil]$, $d \in [D]$, and $j \in [N]$. If $B_{\mathsf{col}} W_\Sigma < Q/2$, every digit ciphertext decrypts to the plaintext in Eq. (43), and the verifier recovers the intended logical packing through Eq. (44).

The proof applies Cauchy–Schwarz to each coefficient of the packed partial decryption, as shown in Appendix F.4.1. It remains to test the column-norm bound.

F.3 Well-Formedness Tests

The tests detect ciphertext lists whose partial-decryption coefficients are too large for safe packing. In each test, the verifier samples a random linear combination of the ciphertexts and checks that its centered partial-decryption coefficients lie below a threshold. The test requires the ordered ciphertext list to be fixed before these random coefficients are sampled.

Centered-ternary combinations. With ordinary 0/1 subset sums, honest coefficients of the same sign can add to Pr. Honest sums concentrate at scale $P\sqrt{r}$ with centered random coefficients, so a smaller acceptance threshold suffices. We use $T \ge 1$ tests and sample independent coefficients

$$x_{a,u} \in \{-1,0,1\}, \qquad \mathbf{Pr}[x_{a,u}=0] = \frac{1}{2}, \qquad \mathbf{Pr}[x_{a,u}=1] = \mathbf{Pr}[x_{a,u}=-1] = \frac{1}{4} \tag{50}$$

for every $(a,u) \in [T]\times[r]$. For every repetition $a \in [T]$, write $\boldsymbol{x}_a := (x_{a,u})_{u\in[r]} \in \{-1,0,1\}^r$. For every coefficient column $\boldsymbol{y}^{(j)}$, these vectors satisfy

$$\mathbf{E}\left[\left\langle \boldsymbol{x}_a, \boldsymbol{y}^{(j)}\right\rangle^2\right] = \frac{1}{2}\left\|\boldsymbol{y}^{(j)}\right\|_2^2.$$

Thus, with the same random combinations, we can improve completeness and test the coefficient-column norm needed for packing.

Fix a target test-completeness error $\epsilon_{\mathsf{wf}} \in (0,1)$, and define

$$c := \sqrt{\ln\left(\frac{2TN}{\epsilon_{\mathsf{wf}}}\right)}, \qquad \tau := \left\lfloor cP\sqrt{r}\right\rfloor. \tag{51}$$

For every repetition $a \in [T]$, the prover publishes the test ciphertext in Eq. (34), with $\mathsf{ct}_u = \mathsf{ct}_{\mathrm{align},u}$. The verifier checks in the packing argument that these are the prescribed linear combinations. The tests pass when Eq. (35) holds for every $a \in [T]$.

Theorem F.3 (BGV well-formedness guarantee)**.** *Fix an ordered ciphertext list* $\mathsf{ct}_1, \ldots, \mathsf{ct}_r \in \mathcal{R}_Q^2$ *before the test coefficients are sampled. Let* y_u *and* $\boldsymbol{y}^{(j)}$ *be its centered partial-decryption lifts and coefficient columns from Definition F.1. If*

$$Q \geq 18cPr, \tag{52}$$

then the centered-ternary test has the following guarantees.

1. ***Completeness.*** *If* $|y_{u,j}| \leq P$ *for every* $u \in [r]$ *and* $j \in [N]$*, all* T *tests accept with probability at least* $1 - \epsilon_{\mathsf{wf}}$*.*
2. ***Soundness.*** *If* $\left\|\boldsymbol{y}^{(j)}\right\|_2 \geq 6cP\sqrt{r}$ *for some* $j \in [N]$*, each independent test rejects with probability at least* $1/2$*. Such a fixed list passes all* T *tests with probability at most* 2^{-T}*.*
3. ***Correct packing.*** *Let* BadPack *be the event that all* T *tests accept but decrypting and recomposing the digit ciphertexts from Eq. (33) does not give the intended packed plaintext of the fixed list. If*

$$6cP\sqrt{rr_{\max}}\left(\frac{\zeta N}{2}\right) < Q/2, \tag{53}$$

then

$$\mathbf{Pr}[\mathsf{BadPack}] \leq 2^{-T}. \tag{54}$$

The honest coefficient bound gives column norms at most $P\sqrt{r}$. The packing condition Eq. (53) allows the larger bound $6cP\sqrt{r}$, so it is enough to detect columns that reach this threshold.

By Hoeffding's inequality, honest test sums remain at scale $P\sqrt{r}$. For soundness, we pair the two choices of one challenge bit when a coefficient column has one dominant coordinate. When its norm is spread across many coordinates, we use a Berry–Esseen anti-concentration bound. The full sampling proof appears in Appendix F.4.2. The BGV packing consequence is proved in Appendix F.4.3. The BFV extension appears in Theorem F.6.

Completeness at the packing boundary. Choose the test bound $P > 0$ so that

$$P < \min\left\{\frac{Q}{18cr}, \frac{Q}{6c\zeta N\sqrt{rr_{\max}}}\right\}. \tag{55}$$

The first upper bound is the modulus condition required for test soundness. The second prevents wraparound during digit packing. Using the actual mask norms, it suffices to require

$$P < \min\left\{\frac{Q}{18cr}, \frac{Q}{12cW_\Sigma\sqrt{r}}\right\}, \tag{56}$$

where W_Σ is defined in Eq. (48). For a fixed list with a column of norm at least $6cP\sqrt{r}$, all tests pass with probability at most 2^{-T} by Theorem F.3. When every column has smaller norm, packing is correct whenever $6cP\sqrt{r}W_\Sigma < Q/2$ by Lemma F.2. The honest coefficient bound in Eq. (41) applies after the final modulus switch, before digit packing.

Corollary F.4 (Completeness at the packing boundary)**.** *Suppose either Eq. (55) or Eq. (56) holds. Let* δ_{noise} *bound the probability that any honest aligned ciphertext violates Eq. (41). Then the honest tests accept and digit packing reconstructs the intended logical outputs with probability at least* $1 - \delta_{\mathsf{noise}} - \epsilon_{\mathsf{wf}}$*. In particular, any choice with* $\delta_{\mathsf{noise}} + \epsilon_{\mathsf{wf}} \leq \epsilon_{\mathsf{cmp}}$ *gives completeness error at most* ϵ_{cmp}*.*

Proof. Fix the secret key and the aligned ciphertext list before sampling the fresh test coefficients. If the list satisfies the coefficient bound, the tests reject with probability at most ϵ_{wf} by Theorem F.3, and the packing bounds ensure correct recomposition. Adding the probability that the coefficient bound fails gives the claim. □

Completeness and efficiency of the compiler. In Theorem 6.1, honest alignment and the final modulus switch preserve the messages by hypothesis. With perfect BhIP completeness, Corollary F.4 and the packing argument's completeness error $\delta_{\mathsf{alg,cmp}}$ give total error at most $\delta_{\mathsf{noise}}+\epsilon_{\mathsf{wf}}+\delta_{\mathsf{alg,cmp}} \le \epsilon_{\mathsf{cmp}}$. The operation and communication counts follow from Appendix F.1.3.

F.4 Proofs of the Well-Formedness Guarantees

We prove the packing bound and test guarantees from Appendices F.2 and F.3.

F.4.1 Coefficient proof of the packing bound

Proof of Lemma F.2. For every offset $h \in [N]$, define the signed cyclic shift

$$S_h(y) := X^{h-1} y \bmod (X^N + 1) \qquad \text{for } y \in \mathcal{R}.$$

Writing $p_u^{\langle d \rangle} = \sum_{h\in[N]} p_{u,h}^{\langle d \rangle} X^{h-1}$ gives

$$\widetilde{y}_{\ell,j}^{\langle d \rangle} = \sum_{h\in[N]} \left\langle (p_{u,h}^{\langle d \rangle})_{u\in\mathcal{I}_\ell}, ((S_h y_u)_j)_{u\in\mathcal{I}_\ell} \right\rangle \tag{57}$$

for every output $\ell \in [\lceil Mr/N \rceil]$, digit $d \in [D]$, and coefficient $j \in [N]$.

For fixed $j, h \in [N]$, there are a coefficient index $k_{j,h} \in [N]$ and a sign $\sigma_{j,h} \in \{-1, 1\}$, both independent of u, such that $(S_h y_u)_j = \sigma_{j,h} y_{u,k_{j,h}}$ for every $u \in \mathcal{I}_\ell$. Hence,

$$\|((S_h y_u)_j)_{u\in\mathcal{I}_\ell}\|_2 \le \left\| \boldsymbol{y}^{(k_{j,h})} \right\|_2 \le B_{\mathsf{col}}.$$

By Cauchy–Schwarz, the contribution of offset h is at most $B_{\mathsf{col}} \left\| (p_{u,h}^{\langle d \rangle})_{u\in\mathcal{I}_\ell} \right\|_2$. Summing over $h \in [N]$ gives the bound $B_{\mathsf{col}} W_\Sigma \le B_{\mathsf{col}} WN$.

Suppose now that $B_{\mathsf{col}} W_\Sigma < Q/2$. Then no coefficient of any $\widetilde{y}_\ell^{\langle d \rangle}$ wraps when reduced modulo Q. By definition, $y_u \bmod t = \mu_u$. Reducing the packed sum modulo t therefore gives

$$\widetilde{y}_\ell^{\langle d \rangle} \bmod t = \sum_{u\in\mathcal{I}_\ell} p_u^{\langle d \rangle} \mu_u \quad \text{in } \mathcal{R}_t.$$

Every digit therefore decrypts to the required plaintext. Applying Eq. (44) completes the proof. □

F.4.2 The centered-ternary sampling bound

The sampling argument is independent of the encryption scheme. Let $z_1, \ldots, z_r \in \mathcal{R}$ be fixed centered representatives of elements of $\mathcal{R}_Q$. For every coefficient $j \in [N]$, define

$$\boldsymbol{z}^{(j)} := (z_{u,j})_{u\in[r]} \in \mathbb{Z}^r.$$

Fix $\epsilon_{\mathsf{wf}} \in (0,1)$ and a real threshold scale $\theta > 0$, and define

$$c := \sqrt{\ln\left(\frac{2TN}{\epsilon_{\mathsf{wf}}}\right)}.$$

For every repetition $a \in [T]$, accept when

$$\left\| [\sum_{u\in[r]} x_{a,u} z_u]_Q \right\|_\infty \le \lfloor \theta \rfloor. \tag{58}$$

Lemma F.5 (Centered-ternary sampling)**.** *The test in Eq. (58) has the following guarantees.*

1. ***Completeness.*** *If $\left\|\boldsymbol{z}^{(j)}\right\|_2 \le \theta/c$ for every $j \in [N]$, the probability that any of the T tests rejects is at most ϵ_{wf}.*
2. ***Soundness.*** *Suppose $Q \ge 18\theta\sqrt{r}$. If $\left\|\boldsymbol{z}^{(j)}\right\|_2 \ge 6\theta$ for some $j \in [N]$, one test rejects with probability at least $1/2$. All T tests accept with probability at most 2^{-T}.*

Completeness. Write each challenge as $x_{a,u} = b^+_{a,u} - b^-_{a,u}$ for independent fair bits $b^+_{a,u}, b^-_{a,u} \xleftarrow{\$} \{0,1\}$. For fixed $(a,j) \in [T] \times [N]$, centering the bits expresses the test sum $\sum_{u\in[r]} x_{a,u} z_{u,j}$ as $2r$ independent mean-zero terms. Their squared interval lengths sum to $2\sum_{u\in[r]} z^2_{u,j} = 2\left\|\boldsymbol{z}^{(j)}\right\|_2^2$. By Hoeffding,

$$\mathbf{Pr}\left[\left|\sum_{u\in[r]} x_{a,u} z_{u,j}\right| \ge v\right] \le 2\exp\left(-\frac{v^2}{\left\|\boldsymbol{z}^{(j)}\right\|_2^2}\right) \tag{59}$$

for every $v > 0$ whenever $\left\|\boldsymbol{z}^{(j)}\right\|_2 > 0$. If $\left\|\boldsymbol{z}^{(j)}\right\|_2 = 0$, the corresponding test sum is always zero.

Centered reduction modulo Q cannot increase absolute value. Because the test sums are integers, rejection at threshold $\lfloor\theta\rfloor$ implies that the unreduced sum has magnitude greater than θ. Under the completeness hypothesis, applying Eq. (59) with $v = \theta$ bounds rejection for each $(a,j) \in [T] \times [N]$ by

$$2e^{-c^2} = \frac{\epsilon_{\mathsf{wf}}}{TN}.$$

Completeness follows by a union bound over the TN pairs.

Soundness for one test. Fix a violating coefficient column $j \in [N]$ before sampling the test, and write

$$L := \left\|\boldsymbol{z}^{(j)}\right\|_2, \qquad A := \left\|\boldsymbol{z}^{(j)}\right\|_\infty, \qquad S := \sum_{u\in[r]} x_{a,u} z_{u,j}, \qquad \tau_0 := \lfloor\theta\rfloor.$$

We show that the centered reduction of S lies outside the radius-τ_0 acceptance interval with probability at least $1/2$.

Suppose first that $A > 2\tau_0$, and let $u^* \in [r]$ attain A. Condition on every challenge bit except b^+_{a,u^*}. Toggling this bit shifts S by $z_{u^*,j}$ modulo Q. Because $z_{u^*,j}$ is centered and attains A, we have $2\tau_0 < |z_{u^*,j}| \le Q/2$. The modulus bound also gives $Q > 4\tau_0$, so the two possible sums are too far apart modulo Q to both lie in the acceptance interval. At most one of the two equiprobable choices accepts.

It remains to consider $A \le 2\tau_0$. Here no individual summand dominates, so we use the Berry–Esseen inequality (a quantitative central limit theorem). Apply it to the $2r$ centered Bernoulli terms used in the completeness proof. Their total variance is $L^2/2$, while the sum of their third absolute moments is at most $AL^2/4$. By Tyurin's general-summand Berry–Esseen inequality [76, Eq. (1) and Thm. 7], we may take $C_{\mathsf{BE}} = 0.561$. The distribution function of $S/(L/\sqrt{2})$ therefore differs from the standard normal distribution function by at most

$$\epsilon_{\mathsf{BE}} \le C_{\mathsf{BE}} \frac{A}{\sqrt{2}L}.$$

A standard normal variable G satisfies $\mathbf{Pr}[|G| \le d] \le d\sqrt{2/\pi}$ for every $d \ge 0$. Applying the distribution-function comparison at both endpoints and using $A \le 2\tau_0$ and $L \ge 6\theta \ge 6\tau_0$ gives

$$\mathbf{Pr}[|S| \le \tau_0] \le \left(\frac{2}{\sqrt{\pi}} + 2\sqrt{2}C_{\mathsf{BE}}\right)\frac{\tau_0}{L} < 0.453. \tag{60}$$

Finally, an integer outside the ordinary acceptance interval can wrap into the centered modular interval only if $|S| \ge Q - \tau_0 > Q/2$. Since $L \le A\sqrt{r} \le 2\theta\sqrt{r}$ and $Q \ge 18\theta\sqrt{r}$, Eq. (59) gives

$$\mathbf{Pr}[|S| \ge Q/2] \le 2\exp\left(-\frac{Q^2}{4L^2}\right) \le 2e^{-18^2/16} < 4 \cdot 10^{-9}.$$

Adding this wraparound probability to Eq. (60) leaves the one-test acceptance probability below $1/2$.

Amplification. Fix a violating coefficient column. Each repetition accepts it with probability at most $1/2$, so independence across $a \in [T]$ bounds acceptance of all T tests by 2^{-T}.

F.4.3 Proof of the BGV guarantee

Proof of Theorem F.3. For completeness and soundness, apply Lemma F.5 with $z_u = y_u$ and $\theta = cP\sqrt{r}$. Honest columns have norm at most $P\sqrt{r} = \theta/c$, and the modulus condition is $Q \ge 18cPr = 18\theta\sqrt{r}$.

It remains to prove the packing consequence. Let BadCol be the event that all tests accept while some coefficient column has norm at least $6cP\sqrt{r}$. If the fixed list has a violating column, fix one before sampling the tests. Acceptance requires this column to pass every repetition, so $\mathbf{Pr}[\mathsf{BadCol}] \le 2^{-T}$ by the sampling lemma. Outside this event, acceptance implies

$$\left\|\boldsymbol{y}^{(j)}\right\|_2 < 6cP\sqrt{r} \qquad \text{for every } j \in [N].$$

Using $W \le (\zeta/2)\sqrt{r_{\max}}$ and the theorem's packing inequality,

$$(6cP\sqrt{r})WN \le 6cP\sqrt{rr_{\max}}\left(\frac{\zeta N}{2}\right) < Q/2.$$

Under this inequality, the prescribed digit ciphertexts decrypt correctly and recompose to the intended logical packing by Lemma F.2. Therefore, $\mathsf{BadPack} \subseteq \mathsf{BadCol}$, and $\mathbf{Pr}[\mathsf{BadPack}] \le 2^{-T}$. □

F.4.4 BFV instantiation

In BFV, the message term can be on the order of Q even for an honest ciphertext. We therefore test the rounding residual, obtained by multiplying the partial decryption by t before centered reduction.

Throughout this subsection, assume that the BFV plaintext modulus t and ciphertext modulus Q are odd. Define

$$\Delta := \left\lfloor \frac{Q}{t} \right\rfloor, \qquad \rho := Q - t\Delta \in \{0, \dots, t-1\}.$$

For every operand $u \in [r]$, write

$$v_u := [\mathsf{L}_{\mathsf{sk}}(\mathsf{ct}_u)]_Q \in \mathcal{R}.$$

For every coefficient $j \in [N]$, let

$$m_{u,j} := \lceil t v_{u,j}/Q \rfloor$$

rounding to the nearest integer, and collect these coefficients in $m_u \in \mathcal{R}$. Since Q is odd, there are no rounding ties. Thus, $m_u \bmod t$ is the plaintext returned by BFV decryption. Define $e_u \in \mathcal{R}$ through

$$v_u = \Delta m_u + e_u. \tag{61}$$

Since $m_{u,j}$ is the nearest integer to $tv_{u,j}/Q$, we have $|tv_{u,j} - Qm_{u,j}| < Q/2$. Thus $tv_u - Qm_u$ is already centered modulo Q, and $t\Delta = Q - \rho$ gives

$$z_u := [t\,\mathsf{L}_{\mathsf{sk}}(\mathsf{ct}_u)]_Q = te_u - \rho m_u \in \mathcal{R}. \tag{62}$$

For every coefficient $j \in [N]$, define $\boldsymbol{z}^{(j)} := (z_{1,j}, \dots, z_{r,j}) \in \mathbb{Z}^r$.

Let $\beta_{\mathsf{BFV}} \geq 0$ be the honest BFV noise bound, and set

$$P_{\mathsf{BFV}} := t\beta_{\mathsf{BFV}} + \rho\frac{t-1}{2}, \qquad \tau_{\mathsf{BFV}} := \left\lfloor cP_{\mathsf{BFV}}\sqrt{r} \right\rfloor. \tag{63}$$

We require $P_{\mathsf{BFV}} > 0$. If $\|e_u\|_\infty \leq \beta_{\mathsf{BFV}}$, then $\|z_u\|_\infty \leq P_{\mathsf{BFV}}$. For every repetition $a \in [T]$, the BFV verifier accepts when

$$\left\| [t\,\mathsf{L}_{\mathsf{sk}}(\mathsf{ct}_a^{\mathsf{test}})]_Q \right\|_\infty \leq \tau_{\mathsf{BFV}}. \tag{64}$$

Theorem F.6 (BFV well-formedness guarantee)**.** *Fix an ordered list $\mathsf{ct}_1, \dots, \mathsf{ct}_r \in \mathcal{R}_Q^2$ before the test coefficients are sampled. If*

$$Q \geq 18cP_{\mathsf{BFV}}r, \tag{65}$$

then the BFV test has the following guarantees.

1. ***Completeness.*** *If $\|e_u\|_\infty \leq \beta_{\mathsf{BFV}}$ for every $u \in [r]$, all tests accept with probability at least $1 - \epsilon_{\mathsf{wf}}$.*
2. ***Soundness.*** *If $\left\|\boldsymbol{z}^{(j)}\right\|_2 \geq 6cP_{\mathsf{BFV}}\sqrt{r}$ for some $j \in [N]$, all T tests accept with probability at most 2^{-T}.*
3. ***Correct packing.*** *If*

$$6cP_{\mathsf{BFV}}\sqrt{rr_{\max}}\left(\frac{\zeta N}{2}\right) < Q/2, \tag{66}$$

then the probability that all tests accept while some prescribed packing-digit ciphertext from Eq. (33) decrypts incorrectly is at most 2^{-T}. Outside this event, if the tests accept, recomposition gives every intended logical packed plaintext.

Proof. By linearity of L_{sk},

$$[t\,\mathsf{L}_{\mathsf{sk}}(\mathsf{ct}_a^{\mathsf{test}})]_Q = [\sum_{u\in[r]} x_{a,u}z_u]_Q.$$

Apply Lemma F.5 with $\theta = cP_{\mathsf{BFV}}\sqrt{r}$. The honest noise bound implies $\left\|\boldsymbol{z}^{(j)}\right\|_2 \le P_{\mathsf{BFV}}\sqrt{r}$ for every $j \in [N]$, so completeness and soundness follow from the sampling lemma.

For packing correctness, Eq. (62) gives the integer identity

$$tv_u = Qm_u + z_u. \tag{67}$$

Fix an output $\ell \in [\lceil Mr/N \rceil]$ and digit $d \in [D]$. Let $\widetilde{v}, \widetilde{m}, \widetilde{z} \in \mathcal{R}$ denote the corresponding sums over $u \in \mathcal{I}_\ell$, weighted by the digit masks $p_u^{\langle d \rangle}$. Then

$$t\widetilde{v} = Q\widetilde{m} + \widetilde{z}.$$

Except with probability 2^{-T}, acceptance implies that every residual column has norm less than $6cP_{\mathsf{BFV}}\sqrt{r}$. By the coefficient calculation in Appendix F.4.1 and Eq. (47),

$$\|\widetilde{z}\|_\infty < 6cP_{\mathsf{BFV}}\sqrt{rr_{\max}}\left(\frac{\zeta N}{2}\right) < Q/2.$$

The centered representative of $\widetilde{v}$ is $\widetilde{v} - Qk$ for some $k \in \mathcal{R}$. Thus

$$\frac{t(\widetilde{v} - Qk)}{Q} = \widetilde{m} - tk + \frac{\widetilde{z}}{Q}.$$

Since every coefficient of $\widetilde{z}$ has magnitude less than $Q/2$, the rounded result is $\widetilde{m} - tk$. Reducing modulo t recovers $\widetilde{m} \bmod t = \sum_{u\in\mathcal{I}_\ell} p_u^{\langle d \rangle}(m_u \bmod t)$, the intended digit plaintext. The verifier combines these digits according to Eq. (44) to obtain the intended logical packed plaintext. □

G Integrity Analysis

We prove the integrity guarantees from Section 6.1 and Section 6.2. The proof begins with the oracle protocols, then accounts for hash commitments and the packing argument before applying Fiat–Shamir. For the packed variants, Theorem F.3 bounds the probability that the tests accept while packing changes the messages from a fixed ciphertext list.

G.1 Security Model and PCS Assumptions

We give the formal definition underlying the soundness analysis in Section 3. The analysis identifies partial transcripts in which a false claim remains to be checked. Any preprocessing and private verifier input, such as the client's secret key, are fixed throughout the definition.

Definition G.1 (RBR soundness, adapted from [28, Definitions 31.1.1–31.1.2])**.** Let $\pi = (P, V)$ be a public-coin IOP for a language $\mathcal{L}$, with $r \ge 1$ verifier challenges. A partial transcript τ contains the explicit messages and full prover oracle strings sent so far, and $\tau \| m$ denotes appending the next message m. A *state function* is a deterministic, possibly inefficient function $\mathsf{State}(x, \tau) \in \{0, 1\}$, where 0 denotes a bad state and 1 a good state. For every instance x, it must satisfy:

1. **Initial state.** For the empty transcript, $\mathsf{State}(x, \varnothing) = 0$.

2. **Prover messages.** For every prefix τ at which the prover is to speak and every next message Π of the prescribed format,

$$\mathsf{State}(x, \tau) = 0 \quad \Longrightarrow \quad \mathsf{State}(x, \tau \| \Pi) = 0.$$

   This includes any final prover response after the last challenge.

3. **Final verification.** For every complete transcript τ, if $\mathsf{State}(x, \tau) = 0$, then V rejects that transcript.

The protocol has *RBR soundness error* ϵ_{rbr} if there is such a state function for which the following bound holds. For every $x \notin \mathcal{L}$, round $i \in [r]$, and prefix τ immediately before challenge i with $\mathsf{State}(x, \tau) = 0$,

$$\Pr_{\rho \leftarrow \mathcal{D}_i(x,\tau)} \big[\mathsf{State}(x, \tau \| \rho) = 1\big] \le \epsilon_{\mathsf{rbr}}, \tag{68}$$

where $\mathcal{D}_i(x, \tau)$ is the protocol's prescribed distribution of the fresh verifier challenge. The prefix is fixed before sampling ρ, and need not come from an honest execution.

We count one random verifier message as one challenge, even when that message contains several field elements or query positions. The RBR error must then account for the entire message. For example, testing a nonzero multilinear polynomial at a random point in $\mathbb{F}^m$ has error at most $m/|\mathbb{F}|$. Our PCS query positions and well-formedness test coefficients are each sampled in one verifier message. For a single execution, the soundness error is at most the sum of the round bounds, or $r\epsilon_{\mathsf{rbr}}$ for r challenge blocks.

In the state-restoration game, the prover may return to earlier transcript prefixes and request further challenges. It wins by producing an accepting transcript for a false statement, with every challenge consistent with the recorded replies. When a query reveals several new prefixes, we process them in transcript order. Every accepting execution for a false statement must leave a bad state at some challenge. With at most q_{mv} fresh challenges, including any sampled to reconstruct a prefix or verify the final output, a union bound gives

$$\varepsilon^{\mathrm{sr}}(q_{\mathsf{mv}}, n) \le q_{\mathsf{mv}} \epsilon_{\mathsf{rbr}}(n). \tag{69}$$

Here $\epsilon_{\mathsf{rbr}}(n)$ is a uniform bound for false instances of size at most n, and the prover may choose the statement adaptively before requesting its first challenge.

Fiat–Shamir and query budgets. The Fiat–Shamir reduction bounds noninteractive soundness by the state-restoration soundness of the interactive protocol [10, 28]. When the transcript is summarized by an ℓ_{FS}-bit hash, collisions contribute an additional error of $O(q_{\mathsf{FS}}^2/2^{\ell_{\mathsf{FS}}})$, where q_{FS} counts hash queries by the adversary and during final verification. The number of protocol challenges obtained in this reduction determines the move budget q_{mv} in Eq. (69). The RBR parameter thus bounds the error at each fresh challenge, while the final soundness bound also depends on how many challenges the adversary tries.

Commitment assumptions. The ciphertext hash H is assumed collision resistant and is instantiated with Poseidon [48]. For the PCS and Fiat–Shamir hashes, we use independent random oracles with separate, unambiguously encoded domains, also independent of H.

In the oracle model, each prover word is fixed before the challenges that test it. To transfer these proofs to hash commitments, we need to reconstruct a word that agrees with the authenticated answers used in accepted openings, except for a bounded authentication error. This word must remain fixed when the prover revisits a transcript prefix, so that the same prefix has the same interpretation throughout the analysis. For the packing commitment, we also need to decode the word into the complete candidate assignment.

Definition G.2 (VCoED-compatible PCS)**.** Fix the fields and dimensions of the committed tables. A complete interactive PCS is *VCoED-compatible* if it has the following properties, conditioned on a good setup.

1. **A fixed polynomial.** In the oracle model, the prover supplies a representation W before subsequent challenges. Its interpretation f_W is a polynomial of the prescribed shape over its specified coefficient field, or $\perp$. This interpretation is fixed by W and need not be efficiently computable. The public-index polynomials come from honest preprocessing and may share a joint commitment.

2. **RBR openings.** The opening's state predicate is a deterministic, possibly inefficient function of W, the opening claim, and the partial transcript. It is bad initially whenever the claimed evaluation or inner product is false for f_W, or $f_W = \perp$. This predicate satisfies the prover-message and final-verification conditions of Definition G.1. At every bad prefix, it satisfies Eq. (68) with error $\epsilon_{\mathsf{PCS},\mathsf{rbr}}$, including when the initial opening claim was true. The verifier checks the public operands, and all claims and batching weights are fixed before the challenges used to test them.

3. **Consistent straight-line extraction.** Given any chronological execution record, an extractor supplies a word of the prescribed length for each commitment at its prefix, including the words used in the opening IOPs. Fix the extractor's randomness. The representation must remain the same if the record is extended or its prefix is revisited. Except with probability δ_{auth}, every authenticated answer used by an accepted opening equals its value in the reconstructed representation. The ideal verifier still performs the same algebraic checks, so false evaluation claims and $f_W = \perp$ are subject to the RBR error above.

   The guarantees cover adaptive statements and state restoration, including final verification and public-index authentication. The extractor uses only the retained record and authorized setup information, without making new oracle queries or changing the statement, transcript, or auxiliary output.

When the adversary revisits a protocol prefix, the extractor keeps the record of all queries made so far. When the security reduction restarts a saved simulation to sample a continuation, it restores the extraction record to that saved state as well. For use as the packing PCS, compatibility additionally requires a deterministic polynomial-time decoder D that computes f_W for the initial witness word and recovers its full candidate assignment, including every constrained coordinate, or returns $\perp$. The extractor and decoder run in worst-case polynomial time. Let δ_{setup} bound the probability that setup is not good.

G.2 Soundness in the Oracle Model

Blind messages as oracles. Each round of GKR is an interactive oracle reduction (IOR): it reduces the current claim to a new claim about the same computation [19, Section 5.1]. The new claim becomes the input to the next round, and composing these IORs with the final checks gives an IOP. The claims refer to both explicit data and oracle strings, including the prover's messages from earlier rounds. Each oracle is fixed when the prover supplies it, before the next verifier challenge. Public auxiliary messages remain explicit, and the verifier may defer reading and decrypting the ciphertext oracles until its final check. For the analysis, fix the client's input and secret key. These determine the true circuit computation and the interpretation of every ciphertext message, including maliciously chosen ciphertexts.

Base's ciphertext vectors. In a Base sumcheck round, the oracle contains two groups of R ciphertexts, all of which the verifier reads. Decrypting each group and combining corresponding slots in the fixed extension-field basis gives one vector in $\mathbb{K}^N$. The verifier aggregates these vectors and recovers the omitted evaluation using the rule for the current layer reduction (Figs. 11 and 13). Shorter terminal messages retain their prescribed formats, including the explicit public coefficient vector in Base's affine terminal message.

For Folded and Laned, the oracle instead contains the single aligned ciphertext from Eq. (32). Its assigned slots hold the base-field coordinates of the round's extension-field values. Reading this one oracle entry gives the entire ciphertext, from which the verifier recovers those coordinates by decryption. In the actual packed protocols, the prover sends a hash commitment to this ciphertext and later proves the packing relation with Spartan.

Claims between rounds. At layer i, the verifier holds a claim $v \stackrel{?}{=} \widetilde{L}_i(\vec{\gamma})$ about the true layer MLE. The layer identity expresses this value as a sum involving the source layer. Using sumcheck, the verifier replaces this sum claim with evaluation claims about the source layer, to be checked in the next layer reduction.

In an ordinary sumcheck round, let $h_j^\star$ be the true round polynomial and h_j the prover's polynomial [20, Sections 2.1 and 6.2]. If the current claim is false, passing the consistency check forces $h_j \neq h_j^\star$. A fresh uniform challenge in $\mathbb{K}$ makes the next claim true only at a zero of this nonzero discrepancy, giving error at most $\deg(h_j - h_j^\star)/|\mathbb{K}|$.

For our blind messages, h_j is determined by decoding the oracle and recovering the omitted evaluation. In Base, this is the aggregate polynomial obtained from the two decoded slot vectors. The earlier message oracles and public transcript determine the current claim. For multiplication, we use the weighted consistency check in Remark D.1, following Gruen's optimization. The same bound on the number of polynomial zeros applies.

At the end of sumcheck, the layer identity relates the remaining claim to source evaluations and any gate values awaiting authentication. For Base, the prover first supplies whole slot vectors, and a fresh slot point reduces their correctness to scalar evaluation claims. The two source claims left by multiplication can be combined into one claim about the source layer by selector interpolation.

Definition G.3 (Satisfying partial GKR transcript)**.** Fix the client's input and secret key. A partial GKR transcript is *satisfying* if every consistency check determined so far passes and every pending claim is correct for the true circuit computation. Pending claims include the current layer or sumcheck claim, the source values or slot tables replacing it at the end of a reduction, and any gate claims awaiting authentication. Their claimed values are determined by the ciphertext oracles and public transcript as above.

False claims through GKR. After the first verifier challenge, a GKR state is bad when its partial transcript is not satisfying. A failed consistency check forces rejection. Otherwise, acceptance requires a future challenge to make all pending claims true. RBR soundness bounds the probability of this event at each round. For instance, a false sumcheck claim may become true at a zero of $h_j - h_j^\star$, but a false gate claim still awaiting authentication keeps the state bad. We track the claims still to be checked, so an earlier false claim that has already been reduced to a true one does not itself keep the state bad.

Polynomial openings. Hash-based multilinear opening proofs admit the same view [50, Sections 2.2 and 5]. For an evaluation claim $f(\vec{z}) \stackrel{?}{=} v$, the point and claimed value are explicit, while the implicit input is an oracle word intended to encode f. For code-based openings, the relations also require proximity to an encoding of a polynomial satisfying the evaluation constraints. Over the opening rounds, these claims are reduced to checks that the verifier performs by querying the

oracles. Both the gate openings and Spartan's final packing-witness opening can be analyzed this way. We first bound the statistical error with oracle access, then account for replacing the oracles with hash commitments.

Lemma G.4 (GKR with deferred gate authentication)**.** *Fix an honestly preprocessed payload index. Let L bound the number of coordinates in every output or slot point sampled as one block. Suppose public gate values are either computed by the verifier or recorded for later authentication, with their values and evaluation points fixed. In the oracle model, the Base, Folded, and Laned reductions have GKR transition error at most*

$$\epsilon_{\mathsf{GKR},\mathsf{rbr}} \leq \frac{\max\{L,3\}}{|\mathbb{K}|-1}. \tag{70}$$

Suppose the gate-opening argument starts in a bad state whenever any recorded claim is false, and its predicate satisfies the prover-message and final-verification conditions of Definition G.1. At every bad prefix, assume its next challenge satisfies Eq. (68) with error $\epsilon_{\mathsf{gate},\mathsf{rbr}}$. GKR followed by gate authentication then has RBR error at most $\max\{\epsilon_{\mathsf{GKR},\mathsf{rbr}}, \epsilon_{\mathsf{gate},\mathsf{rbr}}\}$. A slot point and the following selector challenge are separate challenge blocks.

Proof. Set $\mathsf{State}(x,\tau) = 0$ before the first verifier challenge. During GKR thereafter, set it to 1 exactly when the partial transcript is satisfying (Definition G.3). For a false instance, the initial output claim is incorrect, so this also agrees with the pending-claim interpretation before the first challenge. A failed consistency check forces rejection at the end. Once a gate claim is recorded, later GKR messages cannot change it, so a false recorded claim keeps the state bad throughout GKR. It remains to consider the current arithmetic claims when all recorded gate claims are correct.

For an incorrect output, the claimed and actual output MLEs differ by a nonzero multilinear polynomial. A fresh L-coordinate output point makes their evaluations agree with probability at most $L/|\mathbb{K}|$. Every affine sumcheck has individual degree at most two. For multiplication layer reductions, the prover likewise sends quadratic polynomials using Gruen's optimization (Remark D.1). Since a nonzero quadratic polynomial has at most two zeros, each sumcheck round has error at most $2/|\mathbb{K}|$.

At a terminal check, correct gate values and correct source evaluations imply the correct current claim by the layer identity. Thus a passing check on a false claim leaves either a false recorded gate value or a false source evaluation. For a multiplication reduction, write $\vec{\rho} = (\vec{r}_c, \vec{r}_s)$ for the common suffix of the two source points. After the source values are fixed, a fresh $\alpha \in \mathbb{K}$ gives the single claim $(1-\alpha)v_0 + \alpha v_1 \stackrel{?}{=} \widetilde{L}_{2i}(\alpha, \vec{\rho})$. This uses linearity of the source MLE in its leading selector coordinate. For $b \in \{0,1\}$, let $\delta_b = v_b - \widetilde{L}_{2i}(b, \vec{\rho})$. If at least one source evaluation is false, the discrepancy $(1-\alpha)\delta_0 + \alpha\delta_1$ is a nonzero polynomial of degree at most one, since its values at 0 and 1 are δ_0 and δ_1. The resulting source claim is therefore false except with probability at most $1/|\mathbb{K}|$. Starting from this single claim, the verifier obtains one source-evaluation claim in the next affine reduction.

Base's terminal slot vectors are fixed before its fresh slot point. If a vector differs from the corresponding true slot table, its multilinear discrepancy vanishes with probability at most $\log N/|\mathbb{K}|$. If a coefficient table is incorrect, this test leaves the recorded scalar coefficient claim false except with the same bound. The common slot point can test several tables. If any table is incorrect, fix one such table before sampling the point. Simultaneous correctness implies that this fixed discrepancy vanished. This accounts for the block's total degree without adding a factor for the number of tables.

For the Laned change of point in Eq. (30), the verifier excludes at most one value when sampling a lane-coordinate challenge. Degree-two discrepancies consequently have error at most $2/(|\mathbb{K}|-1)$.

The change of point is an exact identity on the remaining support, so it introduces no additional probabilistic check.

Finally, the client evaluates the input MLE from its input, including the public padding and packing map. A false terminal input claim rejects deterministically. The verifier checks every saved gate claim with its original layer, points, and values. During the gate argument, a false claim either rejects or reduces to a false opening claim, except at a bad-to-good challenge transition. An execution that accepts an incorrect output must therefore leave a bad state at some challenge. Each challenge belongs to either GKR or gate authentication, so the larger of their two per-round bounds is the RBR error of the composition. □

Appendix H.2 instantiates this bound for the concrete challenge field and admitted circuit dimensions.

Packing in the oracle model. Packing must preserve the plaintext messages fixed by the original oracles. We introduce an intermediate protocol used only in the security proof of Folded and Laned. In this protocol, the prover supplies a one-entry ciphertext oracle at each GKR round. The parties complete the gate-opening IOP, after which the verifier samples the well-formedness coefficients and the prover sends the digit and test ciphertexts. The verifier reads the original oracles and checks the packing and test equations in Eqs. (33) and (34) directly. These checks take the place of the interactive packing argument. The verifier then applies the same test thresholds, digit decryption, plaintext recomposition, and GKR checks as in Appendix F.1.1. For BFV, it uses the scaled-residual test in Eq. (64).

Lemma G.5 (Statistical soundness of the oracle protocols)**.** *Fix the client request and secret key, and condition on a good gate-opening setup. For each variant, let* $\epsilon_{\pi,\mathsf{rbr}} := \max\{\epsilon_{\mathsf{GKR},\mathsf{rbr}}, \epsilon_{\mathsf{gate},\mathsf{rbr}}\}$ *be the bound from Lemma G.4, with gate openings in the oracle model. Base's oracle protocol has statistical RBR error at most* $\epsilon_{\pi,\mathsf{rbr}}$*. For the packed variants, assume the well-formedness guarantee in Eq. (54), or its BFV analogue in Theorem F.6. The intermediate oracle protocol then has statistical RBR error at most*

$$\epsilon_{\mathsf{mid},\mathsf{rbr}} := \max\{\epsilon_{\pi,\mathsf{rbr}}, 2^{-T}\}. \tag{71}$$

All well-formedness coefficients are sampled in one challenge block.

Proof. Use the gate-opening bounds of Theorem C.1 and Lemma E.2, with their fixed representations supplied as oracle words. Base transmits every entry of its ciphertext oracles, so direct transmission implements the oracle access exactly. Lemma G.4 therefore gives its bound.

For a packed variant, use the same bad states through GKR and the gate-opening IOP, interpreting each message from its original ciphertext oracle. At the end of this interaction, a bad state means that the plaintext transcript decoded from the original ciphertexts makes the verifier reject. Once the well-formedness coefficients are sampled, the packing and test equations determine all digit and test ciphertexts. Declare the state after this challenge bad if the verifier would reject this response. A different response fails an exact equation check, so a prover message cannot turn this bad state good.

A bad state can become good at this challenge only if all tests pass and the packed transcript accepts although the original plaintext transcript rejects. This requires packing to change at least one decoded message. The well-formedness guarantee bounds this event by 2^{-T} for every fixed ciphertext list. All earlier challenges have error at most $\epsilon_{\pi,\mathsf{rbr}}$, so their composition has the stated maximum RBR error. The argument holds against unbounded provers. □

G.3 From Oracles to Hash Commitments

Ciphertext commitments. The intermediate IOP of Lemma G.5 composes the GKR, gate-opening, and well-formedness reductions. We instantiate its ciphertext oracles by committing to each aligned ciphertext ct_i as $C_i = H(\mathsf{ct}_i)$, for $i \in [r]$. The committed intermediate protocol ends with the prover revealing the original ciphertext list together with the digit and test ciphertexts. The verifier checks each hash opening and the packing and test equations, then performs the well-formedness and GKR checks from Appendix F.1.1. This committed intermediate protocol is also used only in the security proof. Lemma G.8 shows how Spartan replaces the final ciphertext openings in Folded and Laned.

The reduction reconstructs ciphertexts by sampling continuations from their commitment prefixes. Its sampling parameter J trades additional simulation work for a reconstruction error of $1/J$ per ciphertext position. If two accepted openings of the same commitment differ, they give a collision in H.

Gate-opening oracles. Use the consistent extractor of the VCoED-compatible gate PCS (Definition G.2) for every gate-opening oracle, including intermediate folding words. Let $\delta_{\mathsf{gate,auth}}$ bound failures of this guarantee for the selected accepting transcript, including its public-index openings, and let $\delta_{\mathsf{gate,setup}}$ bound setup failure. The algebraic and proximity checks remain part of the gate-opening IOP.

Lemma G.6 (Soundness after ciphertext commitments)**.** *Assume the oracle soundness and well-formedness hypotheses of Lemma G.5 and the gate extraction property above. Let $\mathcal{A}$ be a classical state-restoration adversary for the committed intermediate protocol, with at most q_{mv} fresh challenges, including final verification. For every integer $J \geq 1$, there are collision finders $\mathcal{B}_{i,J}$ for H, indexed by $i \in [r]$, such that*

$$\begin{aligned}\mathbf{Pr}[\mathcal{A}\ \textit{wins}] \leq\ & q_{\mathsf{mv}}\epsilon_{\mathsf{mid,rbr}} + \delta_{\mathsf{gate,setup}} + \delta_{\mathsf{gate,auth}} \\ & + \frac{r}{J} + \sum_{i\in[r]} \mathrm{Adv}^{\mathsf{coll}}_H(\mathcal{B}_{i,J}).\end{aligned} \tag{72}$$

Each collision finder runs at most J sampled continuations of a simulation of $\mathcal{A}$. Its time is $O((J+1)U)$, where U includes one complete simulation, gate-oracle extraction, transcript processing, and ciphertext opening and comparison costs. The parameter J is used only in the reduction.

Proof. First simulate $\mathcal{A}$ while supplying the extracted gate words as actual oracles. Keep the original public transcript, and check that every authenticated answer used in its selected gate openings agrees with these words. On good setup, outside the authentication failure event, an accepting real transcript also accepts in this simulation with the same challenges. The simulation remains defined when setup or authentication fails: it still supplies reconstructed gate words, checks agreement with the selected authenticated answers, and performs all algebraic and proximity checks. This gives a well-defined adversary for a protocol whose gate words are already available and whose ciphertexts remain hash committed.

To apply the state-restoration compiler of [26, Theorem 7.1], specialize its simulation to reconstruct only the ciphertext oracles. A ciphertext oracle has one entry, and opening it reveals that entry. Two different accepted answers for the same commitment immediately give a collision in H. The singleton query bound in [26, Lemma 5.6] contributes $1/J$ per ciphertext position. This bound applies to the transcript selected after the full state-restoration execution. For each ciphertext position, reconstruction starts at the first move consistent with that transcript through that position, so the reconstruction errors sum over its r positions [26, Construction 6.3 and Section 6.2.2]. In the

compiler's simulation, the gate words and explicit messages are copied directly, so they incur no reconstruction error. The simulator stores words by their original public prefixes and includes the original commitments and explicit messages in its ideal challenge queries. This keeps distinct prefixes separate even if their extracted words coincide. It forwards each fresh challenge request once, including those needed for final verification, and therefore uses at most q_{mv} fresh oracle challenges. Lemma G.5 and Eq. (69) bound the oracle adversary's success by $q_{\mathsf{mv}}\epsilon_{\mathsf{mid,rbr}}$, outside setup failure.

When sampling continuations, the compiler restores the full simulation state, including the independent PCS random-oracle simulator and extraction record. These samples use the same oracle simulation, so the authentication error enters only in the initial comparison. The reduction samples and keeps the client key internally to evaluate the final predicate. All challenges and oracle query locations remain public. Together these bounds give Eq. (72). □

G.4 Extracting the Packing Witness

To replace the final ciphertext openings, we must recover a list that opens every round commitment and satisfies the packing and test equations. The state-restoration knowledge error bounds the probability that the verifier accepts but extraction fails to recover a valid assignment to $\mathcal{R}_{\mathsf{pack}}$ (Appendix F.1.2).

We analyze the witness as an oracle word W, whose decoded polynomial over the packing base field $\mathbb{F}$ fixes a candidate assignment before Spartan's challenges. The decoder required for a VCoED-compatible packing PCS (Definition G.2) recovers every constrained coordinate of this assignment. Write $\epsilon_{\mathsf{SpPCS,rbr}}$ for the packing PCS's opening error.

Using the opening IORs, the verifier checks the witness and public-index evaluation claims from Appendix F.1.2 against their oracle words. Composing them with Spartan's reductions gives the packing IOP.

For the candidate assignment z, Spartan first checks that the multilinear extension of the row residuals $(Az)_i(Bz)_i - (Cz)_i$ vanishes at a random point. For a relation with 2^{ℓ_C} padded constraint rows, this contributes error at most $\ell_C/|\mathbb{K}_{\mathsf{pack}}|$. The constraint sumcheck includes a linear equality factor, which is omitted from the prover's quadratic message. A false residual can vanish when either the quadratic discrepancy or the linear equality factor is zero, giving error at most $3/|\mathbb{K}_{\mathsf{pack}}|$. The A, B, C batch uses coefficients $1, \gamma, \gamma^2$ for a fresh challenge γ, so it has degree at most two. The wiring and public-index product sumchecks also have individual degree at most two. Let d_{batch} bound the degree of any further batching check, taking $d_{\mathsf{batch}} = 0$ when none is used. In the public-index argument of Appendix F.1.2, the verifier checks each table contribution separately, without further random batching. These bounds give

$$\begin{aligned}\epsilon_{\mathsf{Sp,rbr}} &:= \frac{\max\{\ell_C, 3, d_{\mathsf{batch}}\}}{|\mathbb{K}_{\mathsf{pack}}|},\\ \epsilon_{\mathsf{pack,rbr}} &:= \max\{\epsilon_{\mathsf{Sp,rbr}}, \epsilon_{\mathsf{SpPCS,rbr}}\}.\end{aligned} \tag{73}$$

Lemma G.7 (Interactive packing extraction)**.** *With a VCoED-compatible PCS for packing (Definition G.2), the packing IOP has statistical RBR knowledge error at most $\epsilon_{\mathsf{pack,rbr}}$ conditioned on good setup. For the committed protocol, a straight-line extractor has state-restoration knowledge error*

$$\kappa_{\mathsf{pack}}(q_{\mathsf{mv}}) \le q_{\mathsf{mv}}\epsilon_{\mathsf{pack,rbr}} + \delta_{\mathsf{auth}} + \delta_{\mathsf{setup}}, \tag{74}$$

where q_{mv} includes explored challenges and final verification. Unless setup or authentication fails, decoding at the end gives the same candidate as decoding at the witness commitment.

Proof. The extractor decodes the initial witness representation using D and always checks every constraint of $\mathcal{R}_{\mathsf{pack}}$ before returning an assignment, even if the PCS extraction guarantee fails. The knowledge guarantee must hold for every public packing instance, including a true statement paired with an invalid candidate assignment. If this assignment is valid, extraction succeeds regardless of the later transcript.

Set $\mathsf{State} = 0$ before the first verifier challenge. Afterward, the state is bad when a consistency check has failed or a pending claim is false. If the candidate assignment is invalid, its constraint residual is nonzero. Except on one of the algebraic errors bounded above, a false witness or matrix evaluation claim then remains to be checked. Using the public-index argument and PCS openings, the verifier eventually checks every such claim. Until it is checked, a false claim remains part of the verifier's state, so no prover message can remove the last false claim. If decoding instead returns $\perp$, keep the state bad until the required witness opening begins, then use its PCS predicate, which starts bad regardless of the claimed evaluation. In either case, acceptance requires a bad-to-good challenge transition, giving the maximum RBR bound.

The first witness word fixes the extracted candidate before any challenge, so later messages cannot change whether it is valid. The bound therefore covers the joint event of a bad-to-good transition and extraction failure, as required for RBR knowledge soundness [28, Definition 31.1.6]. The stated knowledge error for the ideal interaction follows from the state-restoration knowledge theorem [28, Theorem 31.3.1]. For the committed interaction, the PCS extractor recovers the oracle words, after which the same decoder supplies the candidate witness. Accounting for setup and authentication failures gives the stated bound. For Merkle commitments in the random oracle model, this is the interactive BCS knowledge reduction of [28, Lemma 26.1.4], with the multi-extraction guarantee covering all opening oracles and the selected transcript. □

Let L be the chronological record length, S_{pack} the relation-checking cost, t_{PCS} the PCS extraction cost, and t_{D} the decoding and assignment-recovery cost. Then $t_{\mathsf{ext}} \le t_{\mathsf{PCS}} + t_{\mathsf{D}} + O(S_{\mathsf{pack}} + L)$.

Lemma G.8 (Replacing the final openings)**.** *Assume the packing PCS is VCoED-compatible (Definition G.2). Let $\mathcal{A}$ be a classical state-restoration adversary for Folded or Laned, with at most q_{core} fresh challenges through the well-formedness coefficients and q_{pack} during the packing argument. Both budgets cover all explored transcripts and final verification. There is an adversary $\mathcal{B}$ for the committed intermediate protocol of Appendix G.3, using at most q_{core} fresh challenges, such that*

$$\mathbf{Pr}[\mathcal{A}\ \mathit{wins}] \le \mathbf{Pr}[\mathcal{B}\ \mathit{wins}] + \kappa_{\mathsf{pack}}(q_{\mathsf{pack}}). \tag{75}$$

Its running time is at most $t_{\mathcal{A}} + t_{\mathsf{ext}} + t_{\mathsf{sim}}$, where t_{sim} covers challenge simulation, public packing verification, and maintaining the execution record.

Proof. Run $\mathcal{A}$, forwarding its GKR, gate-opening, and well-formedness challenge requests to the intermediate protocol's state-restoration game. Sample packing challenges locally, using the prescribed distributions and returning the same answer whenever the same full transcript prefix is queried again. Keep the chronological PCS query record throughout this exploration. Once $\mathcal{A}$ selects its final transcript, complete any missing challenge queries and the public packing verification, including its PCS hash queries. Then apply Lemma G.7 once to the selected packing transcript and the execution record. If extraction fails, output a rejecting transcript. Otherwise, replace the packing argument by the extracted ciphertext list.

If $\mathcal{A}$ wins and extraction succeeds, this list opens every original commitment and produces the revealed digit and test ciphertexts. The gate-opening transcript and all remaining checks are unchanged, including the test thresholds, packed decryption, recomposition, and GKR verification.

The intermediate verifier therefore accepts the same incorrect output. The probability that the selected packing interaction accepts but extraction fails is at most $\kappa_{\mathsf{pack}}(q_{\mathsf{pack}})$, which proves Eq. (75). □

G.5 Soundness of the Compiled Protocol

Within the intermediate and packing IOPs, each challenge belongs to one component, so their RBR bounds take the maximum of the component errors. The final reduction incurs errors from both packing extraction and the intermediate protocol. We conservatively give each the full challenge budget, yielding the sum in the theorem below.

Theorem G.9 (Transcript packing soundness)**.** *Use the compiler of Theorem 6.1 with VCoED-compatible gate and packing PCSs (Definition G.2). Conditioned on good setup, let $\epsilon_{\mathsf{mid,rbr}}$ bound the intermediate IOP's statistical RBR error and $\epsilon_{\mathsf{pack,rbr}}$ the packing IOP's statistical RBR knowledge error (Lemma G.7). Lemma G.5 gives the intermediate bound for* Laminate*. Define*

$$\epsilon_{\mathsf{tp}} := \epsilon_{\mathsf{mid,rbr}} + \epsilon_{\mathsf{pack,rbr}}. \tag{76}$$

Let δ_{setup} and δ_{auth} jointly bound setup and PCS authentication/extraction failures, including public-index openings and intermediate folding words. For every classical state-restoration adversary $\mathcal{A}$ using at most q_{mv} fresh challenges, including final verification, and every integer $J \geq 1$, there are collision finders $\mathcal{B}_{i,J}$ for H, indexed by $i \in [r]$, such that

$$\begin{aligned}\mathbf{Pr}[\mathcal{A}\ \textit{wins}] \leq\ & q_{\mathsf{mv}}\epsilon_{\mathsf{tp}} + \delta_{\mathsf{setup}} + \delta_{\mathsf{auth}} \\ & + \frac{r}{J} + \sum_{i\in[r]} \mathrm{Adv}^{\mathsf{coll}}_{H}(\mathcal{B}_{i,J}).\end{aligned} \tag{77}$$

Each collision finder uses one initial simulation and at most J sampled continuations. If the initial simulation and each continuation take at most U work, including extraction and transcript processing, its total work is at most $(J+1)U$. For polynomial resource bounds, collision resistance and negligible component errors give negligible state-restoration soundness error.

Proof. Given an attack on the compiled protocol, extract a packing witness to obtain an attack on the committed intermediate protocol, as in Lemma G.8. Packing extraction contributes at most $q_{\mathsf{mv}}\epsilon_{\mathsf{pack,rbr}}$ to the error, in addition to PCS setup and authentication failures. By Lemma G.6, the intermediate attack succeeds with probability at most

$$q_{\mathsf{mv}}\epsilon_{\mathsf{mid,rbr}} + \frac{r}{J} + \sum_{i\in[r]} \mathrm{Adv}^{\mathsf{coll}}_{H}(\mathcal{B}_{i,J}),$$

with its PCS failures accounted for in the same way.

Both reductions use the same consistent reconstruction of the PCS words. The joint setup and authentication errors therefore enter once, when comparing the original execution with these reconstructed oracles. The subsequent simulations are defined even on failed authentication or extraction, and the RBR bounds use the unconditioned challenge distributions. Each sampled continuation reruns the packing extractor and rejects unless it obtains a ciphertext list satisfying every constraint. Repeating extraction therefore increases the reduction's running time but adds no knowledge-error term beyond the initial replacement of the packing argument. This proves Eq. (77).

Reduction cost and negligibility. Each collision finder runs one initial simulation and at most J continuations, giving the stated $(J+1)U$ work bound, including extraction, transcript

processing, state restoration, and ciphertext opening and comparison costs [26, Construction 6.5 and Theorem 7.1].

For any inverse-polynomial target error, choose a polynomial J large enough that r/J is less than half that target. Every collision finder then has polynomial running time, so collision resistance makes their total success probability negligible. The statistical oracle errors and PCS failures are also negligible by hypothesis. This gives negligible state-restoration soundness, as in [26, Corollary 7.2]. □

Fiat–Shamir and VCoED integrity. In the proof of Theorem 6.2, we apply Fiat–Shamir to the entire interaction and obtain negligible soundness error.

H Evaluation Methodology Details

H.1 Builds and Timing Boundaries

The local evaluator and verifier use native field arithmetic with the same compiler optimization settings, including automatic vectorization. Proof-message decryption and the packed variants' threshold checks use SEAL. Section 7.1 defines the reported timing boundaries. The artifact's benchmark guide gives the build environment and detailed timing breakdowns.

H.2 Integrity Parameters

We instantiate Theorem G.9 with the parameters below to obtain the 2^{-50} integrity bound in Corollary H.1. The calculation accounts for PCS opening errors, well-formedness tests, shared code setup, authentication, ciphertext commitments, and Fiat–Shamir.

We use BaseFold [78] at inverse code rate 8, with the parameters in Table 15. The field for GKR and gate openings is $\mathbb{K} = \mathbb{F}_{65537^8}$. For packing, we encode over $\mathbb{F}_q$ and sample challenges in $\mathbb{K}_{\mathsf{pack}} = \mathbb{F}_{q^2}$, where $q = 1053858393162252289$.

Openings	Code	Max. entries	Query paths	RBR error
Laned gates	Reed–Solomon	2^{13}	150	$\leq 2^{-110}$
Packing	Reed–Solomon	2^{34}	150	$\leq 2^{-110}$
Wide gates	Foldable	2^{40}	300	$\leq 2^{-112}$

Table 15: PCS parameters and arithmetic RBR bounds for a good setup. Entry counts are before encoding.

For the arithmetic error calculation, we limit GKR challenge blocks to 64 coordinates. Outside the PCS, each gate or packing challenge tests a polynomial identity of total degree at most 512.

Every evaluated packing relation has at most 2^{32} padded constraint rows, so $\ell_C \leq 32$. The artifact's packing guide gives the counts for each configuration. The A, B, C batch has degree two, and the verifier checks each public-index table contribution separately without a further batching challenge. Thus $d_{\mathsf{batch}} = 0$ and $\epsilon_{\mathsf{Sp,rbr}} \leq 32/q^2$ in Eq. (73). Together with the PCS bounds below, this gives the packing IOP bound $\epsilon_{\mathsf{pack,rbr}} \leq 512/q^2$ used in the final calculation.

Choice of codes. We use Reed–Solomon codes for Laned gate tables and packing witnesses, whose encoded lengths fit the evaluation domains of their respective fields. Within a decoding radius below half the code's minimum distance, there is at most one nearby codeword. We analyze BaseFold's folding rounds within this unique decoding radius using the proximity-gap bounds below.

Wide tables can exceed the gate field's radix-two Reed–Solomon domain, so we use BaseFold's generic foldable codes. Starting from a Reed–Solomon code on 2^{13} message entries, we fix one public hierarchy for tables of up to 2^{40} entries. Its random diagonals are independent uniform nonzero elements of $\mathbb{K}$, sampled at fixed, domain-separated random-oracle addresses shared by all openings. We show below that, with probability at least $1 - 2^{-123}$ over setup, every code in the hierarchy has relative distance greater than 0.6965. This distance is lower than the Reed–Solomon guarantee and requires more query paths. The setup error is charged once for the shared hierarchy, while Reed–Solomon setup is deterministic.

Wide gate openings use the honestly encoded public index and local-operator checks of Appendix C.

Wide-code distance and setup error. For a code on L message variables, let D_L be the distance bound obtained by applying the zero-count bound of [78, Theorem 2] at each extension. Conditioned on successful setup, this code and all smaller codes in the hierarchy have distance strictly greater than D_L, with $D_L > 0.6965$ at every supported size. We allocate failure probability at most 2^{-128} to each of the 27 extensions, giving shared setup error at most $27 \cdot 2^{-128} < 2^{-123}$. The artifact gives the recurrence and exact calculations for these bounds.

Recovering the packing witness. For Spartan's knowledge argument, we need a candidate witness fixed before its first challenge. The security reduction uses the interactive BCS extractor [28, Section 25.2 and Lemma 26.1.4] to obtain this witness. For each PCS commitment, the extractor uses the hash queries available when the prover first supplies its Merkle root, the hash digest at the top of the commitment tree. It reconstructs the committed word from those queries, filling any unrecovered entries with zero. The reduction saves this word for the given Merkle root and public tree configuration, so later queries and revisited prefixes cannot change it. Except with the authentication error bounded below, every authenticated entry used by an accepted opening agrees with this saved word.

The reduction then decodes the initial witness word within the radius specified below, returning $\perp$ if decoding fails. The entries of this word and the Reed–Solomon evaluation points lie in $\mathbb{F}_q$, although the opening challenges lie in the larger field $\mathbb{K}_{\mathsf{pack}}$. Raising the decoded polynomial's coefficients to the qth power preserves its agreement with the word. Uniqueness therefore forces these coefficients to lie in $\mathbb{F}_q$ as well. This gives the fixed candidate witness required for a VCoED-compatible packing PCS (Definition G.2).

PCS opening errors. We choose shrinking decoding radii so that K distinct good folding challenges would force the preceding state to be good. A bad state can therefore become good for at most $K - 1$ challenge values. We specialize the weighted folding argument of [50, Sections 5–6] using proximity radii

$$\delta_j = \delta_0(1 - 1/K)^j \quad (0 \le j \le L),$$
$$(K, \delta_0) = \begin{cases} (512, 223/512) & \text{for Reed–Solomon,} \\ (2^{16}, D_L/3) & \text{for Wide foldable codes.} \end{cases}$$

At round j, a state is good if the sumcheck checks so far pass, a decoded message satisfies the current sumcheck claim, and at least a $1 - \delta_j$ fraction of positions of the initial word follow paths that pass the folding checks so far and end at the corresponding codeword value. A folding challenge is good if some prover response produces a good next state.

For Reed–Solomon codes, we show that the polynomials decoded at K good challenges are all folds of a single candidate polynomial for the preceding round. With distance bound $d = 7/8$, the proximity-gap result of [8, Theorem 1.3 and Section 2.3] applies at radius γ when $d/3 \le \gamma \le$

$d/2 - 1/n$ and $K \geq (d/\gamma - 1)/(d - 2\gamma)$, where n is the code length. The parameter calculations in the artifact show that our radii satisfy these conditions for $n \geq K$. For shorter codes, the same conclusion follows from [9, Theorem 4.1] since $K > n$ and $\delta_j < d/2$.

For Wide codes, the code used after folding has distance strictly greater than $3\delta_j$. Two decoded words at distinct challenges determine a candidate codeword before folding. If a third decoded word differed from the corresponding fold of this candidate, their difference would be a nonzero word in the code used after folding. It could be nonzero only where at least one of the three decoded words disagrees with the corresponding fold of the preceding word, on at most a $3\delta_j$ fraction of coordinates. This contradicts the distance bound. In both cases, counting over these original positions bounds the fraction of paths that fail an earlier check or disagree with the reconstructed codeword by $K\delta_j/(K-1) = \delta_{j-1}$. The quadratic sumcheck identity holds identically once it holds at these K challenges. Thus at most $K - 1$ challenges can turn a bad state good. The folding errors are bounded by $512/|\mathbb{K}|$ for Laned gates, $512/|\mathbb{K}_{\mathsf{pack}}|$ for packing, and $2^{16}/|\mathbb{K}| < 2^{-112}$ for Wide codes.

The verifier samples each opening's query paths independently, with replacement, in one challenge after the folding commitments. With s paths, the terminal query error is at most $(1-\delta_L)^s$. At every supported size, this error is smaller than 2^{-112} with 150 query paths for Reed–Solomon codes and 300 for Wide codes. The artifact verifies these bounds with exact arithmetic for Reed–Solomon codes and Wide codes. Taking the maximum of the folding and query errors gives Table 15.

Well-formedness parameters. The packed benchmarks use $T = 120$ tests and $c = 37/4$, with thresholds chosen to allow noise coefficients of magnitude at most 17781 after alignment and the final modulus switch (Eq. (41)). These parameters satisfy Eq. (56) for the evaluated workloads, with plaintext modulus $t = 65537$ and the single 60-bit final modulus q. The artifact gives the threshold parameters and mask bounds and verifies these inequalities for each workload. Completeness also depends on the probability δ_{noise} that any honest ciphertext exceeds this limit (Corollary F.4), which our benchmarks do not estimate.

Hash assumptions and adversarial budgets. We allow a classical server at most $q_{\mathsf{mv}} = 2^{60}$ fresh protocol challenges, $q_{\mathsf{FS}} = 2^{60}$ Fiat–Shamir hash queries, and $q_{\mathsf{PCS}} = 2^{60}$ PCS-hash queries in the original execution. These budgets cover all explored transcripts, setup queries, and final verification, as applicable. We also impose a work budget $U \leq 2^{60}$ per simulated execution, including the server's computation and the reduction's extraction, verification, and transcript processing. We measure this work in Poseidon permutation evaluations. Each collision finder runs at most $J + 1$ such simulations (Theorem G.9). A ciphertext-hash evaluation may require many permutations.

For ciphertext commitments, we use Poseidon with six field elements for both its capacity and its digest. The artifact's ciphertext-commitment guide specifies the full permutation parameters, constant generation, and parameter checks. We use the birthday-security target behind Poseidon's capacity and round-selection rules [48, Sections 2.1 and 5.2]. Concretely, for each collision finder $\mathcal{B}$ with work bound $W \leq 2^{125}$, we assume

$$\mathrm{Adv}_H^{\mathsf{coll}}(\mathcal{B}) \leq W^2/q^6. \tag{78}$$

The ciphertext hash is assumed collision resistant, while the independent PCS and Fiat–Shamir hashes are modeled as random oracles. We use 320-bit PCS hashes and 256-bit Fiat–Shamir hashes. No new oracle queries are needed for straight-line extraction.

Random-oracle errors. Let c_{PCS} count the distinct Merkle roots fixed at challenge prefixes or in PCS setup. Each online Merkle commitment precedes a fresh challenge, and each setup commitment's root is a setup hash output, so $c_{\mathsf{PCS}} \leq q_{\mathsf{mv}} + q_{\mathsf{PCS}} + 1 < 2^{62}$. The query-record extraction argument of [28, Section 18.5] applies to all our Merkle trees together. Reconstruction can fail only if a fresh hash answer equals an earlier answer, a child digest already supplied in

a parent-hash query, or an already supplied Merkle root. There are at most $3q_{\mathsf{PCS}} + c_{\mathsf{PCS}}$ such targets per query, including queries made during verification. Combining this union bound with the hash-chain Fiat–Shamir bound of [28, Theorem 15.2.1] gives

$$\delta_{\mathsf{auth}} \le \frac{3q_{\mathsf{PCS}}^2 + c_{\mathsf{PCS}} q_{\mathsf{PCS}}}{2^{320}} < 2^{-197},$$
$$\delta_{\mathsf{FS}} \le \frac{q_{\mathsf{FS}}^2}{2^{256}} \le 2^{-136}.$$

Their sum is below 2^{-135}.

Corollary H.1 (Concrete integrity after Fiat–Shamir)**.** *Under the parameters, hash assumptions, and adversarial budgets above, the probability that a classical malicious server makes the client accept an incorrect output is*

$$\mathbf{Pr}[\textit{client accepts an incorrect result}] < 2^{-50}. \tag{79}$$

For packed variants, require Eq. (56).

Proof. The GKR and gate-opening bounds give Base statistical RBR error below 2^{-112}. For packed variants, the well-formedness error is at most 2^{-120} by Theorem F.3 and Lemma F.2. Thus the two oracle errors satisfy

$$\epsilon_{\mathsf{mid,rbr}} < 2^{-112}, \qquad \epsilon_{\mathsf{pack,rbr}} \le 512/q^2 < 5 \cdot 2^{-113}.$$

Their contribution in Theorem G.9 is less than $7 \cdot 2^{-53}$ at $q_{\mathsf{mv}} = 2^{60}$.

Choose the compiler's sampling count $J = 2^{64}$, which is used only in the security reduction. Since $r \le 517 < 2^{10}$, its sampling error is at most $r/J \le 2^{-54}$. Each collision finder uses at most $(J+1)U < 2^{125}$ work. As $q^6 > 2^{359}$, the summed collision probabilities are less than $2^{10}(2^{125})^2/q^6 < 2^{-99}$. Together with the authentication and Fiat–Shamir errors and the shared setup error below 2^{-123}, these remaining terms contribute less than 2^{-98}. The total is therefore less than

$$7 \cdot 2^{-53} + 2^{-54} + 2^{-98} = 15 \cdot 2^{-54} + 2^{-98} < 2^{-50}.$$

Base omits the packing and ciphertext-commitment terms, so the same bound applies. □

H.3 Asymptotic Comparison with Prior Work

Phalanx's asymptotic analysis uses $\Theta(\sqrt{n})$ rows of length $m = \Theta(\sqrt{n})$ [80, Section 5.1]. At each of its t NTT stages, each output depends on $\Theta(m^{1/t})$ values, requiring $\Theta(tnm^{1/t})$ scalar operations across the rows. With SIMD batching, this cost falls by at most a factor of N, giving the $\Omega((t/N)n^{1+1/(2t)})$ term in Table 2. Its $3n$ matrix-product multiplications account for the linear term.

In Phalanx's Brakedown-style opening, the prover sends columns and a linear combination of the rows, whose combined length is at least $2\sqrt{n}$ field elements for any matrix with n entries. During the packed sumcheck, the prover sends at least one ciphertext in each of $\log_2(n/N)$ rounds, followed by the remaining values [80, Section 4.3 and Figure 9]. Each ciphertext contributes $\Theta(N)$ coefficients, so the total communication is $\Omega(\sqrt{n} + N\log(n/N))$, as in Table 2.

Phalanx computes its commitment and sumchecks from the same witness, so it needs only the larger of their depths. Packing and the NTT take $1 + t/2$ levels, while matrix products, packing, and sumchecks take 4. The latter is lower than Phalanx's reported five levels because we count scalar multiplications as half a level [80, Sections 4.3 and 5.2].

Blind Fractal's NTT routine uses $t-1$ radix-b butterfly stages followed by a matrix multiplication [40, Section 6.1.1]. With $P \le N$ packed values per ciphertext, write $k = n/P = b^{t-1}w$, where w is the row width in ciphertexts. These stages require at least $k[(t-1)(b-1)+w]$ multiplications. By the arithmetic–geometric mean inequality, this is at least $k[tk^{1/t}-(t-1)]$, giving $\Omega(t(n/N)^{1+1/t})$ FHE operations for fixed t.

The Blind Fractal prover halves its FRI tables until they fit in one ciphertext and opens ciphertexts from each queried layer [40, Appendices C.3 and F]. Since a ciphertext holds at most N field elements, this also gives $\Omega(\log(n/N))$ ciphertexts. Counting $\Theta(N)$ coefficients per ciphertext gives the $\Omega(N\log(n/N))$ communication bound in Table 2.

Blind Fractal needs an inverse NTT and a forward NTT for the initial encoding, then another pair for its polynomial-sumcheck response. Unpacking, matrix products, and the sumcheck and FRI combinations add four plaintext-multiplication levels, giving $4t+4$ before output conversion [40, Appendices C.1–C.3] [41].

H.4 Concrete Estimates for Prior Work

The prior-work entries in Table 9 assume size-n witnesses in each scheme's native field and retain the papers' matrix-density assumptions and primitive prices. The artifact's comparison guide gives the counting and storage calculations and reproduces the estimates under the assumptions below.

Phalanx. Phalanx's 1034-second matrix computation at $n = 2^{20}$ scales by $n/2^{20}$ at fixed $N = 2^{14}$ [80, Section 5.2]. This also applies to Phalanx*.

At $n = 2^{20}$, we use the reported full proof of 504 ciphertexts and 61.4 MB [80, Table 3]. At larger sizes, we count the opened columns, row combinations, and two sumcheck transcripts [80, Figure 6 and Section 5.1], giving at least 510 and 904 ciphertexts at $n = 2^{27}$ and $n = 2^{34}$. We price these at the reported average bytes per ciphertext and omit other messages and authentication paths.

Blind Fractal. The authors' estimator [41] uses 256 packed values per ciphertext and an average matrix row weight of three. Following the paper, we estimate work at ring dimension 12,288 using operation timings measured at ring dimension 2^{14} [40, Section 8.2.1]. We keep both dimensions fixed. We choose the estimator's least expensive NTT parameters with depth at most five (four at $n = 2^{27}$).

The larger instances fail the source's noise checks, and at $n = 2^{34}$ the original field is also too small for the code domain and stated integrity target. We use these parameters in the estimates and omit the cost of enlarging them.

For proof size, we use output dimension 768, a 24-bit modulus, and 16 slots, as in Remark 3 [40]. Following their Appendix F, we use the expected number of distinct ciphertexts requested by 101 query paths sampled independently with replacement, including paired FRI openings. The estimates exclude hashes, other proof data, and the conversion and repacking needed to produce the smaller ciphertexts.

Additional modulus bits. Phalanx's reported 420-bit modulus leaves 300 bits for proving after excluding its 60-bit evaluation-key prime and 60-bit output limb [80, Section 5]. Blind Fractal's published noise increments total 250 bits [40, Table 1]. We keep these values for larger circuits and leave out any extra bits needed for witness reduction.

Decryption estimates. The prior-work client entries estimate lower bounds from proof-ciphertext decryption alone, omitting input encryption, output decryption, and all remaining verification work. For Phalanx we use 0.40 ms per ciphertext, rounded down from our measured time to decrypt and unpack a ciphertext with one modulus limb. We use 504 ciphertexts at the reported baseline and the

counts above at larger sizes. For Blind Fractal, we use our BGV timings as a proxy for GBFV, scale this price by $N \log N$ to output dimension 768, and use the expected number of opened ciphertexts.

H.5 Round Counts and Communication for Laminate

We derive the round counts used in Theorem 6.2 and the communication bounds in Table 2.

Round counts. Let r count the BhIP rounds that send encrypted GKR messages, including one terminal round per layer reduction. We account separately for the public gate-opening arguments. Recall that $d = 2d_\times + 1$ and $s_i = \log S_i$ in the dense alternating arithmetization. Every non-input odd-layer width appears once in a multiplication reduction and once in an affine reduction, while the input width appears once. Thus, the Base and Folded BhIP round counts are

$$r_{\mathsf{base}} = 2\sum_{i=1}^{d_\times} s_{2i-1} + s_d + d,$$
$$r_{\mathsf{fold}} = r_{\mathsf{base}} + d \log N.$$

The laned BhIP round count has the same form as the Folded BhIP count, using the packed widths S'_i from Appendix E.3. Write $m = \sum_{i=1}^{d_\times} S_{2i-1}$ and $m' = \sum_{i=1}^{d_\times} S'_{2i-1}$. By Jensen's inequality,

$$\begin{aligned} r_{\mathsf{base}} &\le d\left(\log\left(\frac{2m + S_d}{d}\right) + 1\right), \\ r_{\mathsf{fold}} &\le d\left(\log\left(\frac{2m + S_d}{d}\right) + \log N + 1\right), \\ r_{\mathsf{laned}} &\le d\left(\log\left(\frac{2m' + S'_d}{d}\right) + \log N + 1\right). \end{aligned} \tag{80}$$

Proof sizes. In all three BhIPs, the prover sends at most two extended ciphertexts per round, or $2R$ raw ciphertexts. Their affine sumchecks have individual degree at most two, and the multiplication reductions use the quadratic round polynomials from Gruen's equality optimization (Remark D.1). In both cases, the verifier recovers one evaluation from the current claim, leaving two field elements to transmit (Remark 3.2). The Base BhIP prover sends vectors of per-slot evaluations, while the Folded and laned BhIP provers aggregate these evaluations across slots homomorphically. Terminal rounds also send at most two extended ciphertexts. Padding shorter round messages with public zero ciphertexts gives $M = 2R$ for the transcript packing compiler. Its hypotheses require $2R \mid N$ and $2R < N$.

The $\mathsf{Laminate}_{\mathsf{base}}$ proof therefore contains at most $2Rr$ ciphertexts. For each packed variant, Theorem 6.1 gives $D\lceil 2Rr/N \rceil + T$ ciphertexts, r hash outputs, and the public messages and openings of the packing argument. Fiat-Shamir challenges are reconstructed by the verifier and need not be transmitted. The public coefficient-opening arguments remain in all three proofs, including their claimed values and auxiliary sumchecks. For the wide BhIPs these are described in Appendix C, and for the laned BhIP in Appendix E.3.1. The Base affine reduction additionally sends an N-entry table over $\mathbb{K}$ as described in Appendix C. There are $d_\times + 1$ such tables.

For the bounds in Table 2, assume each polynomial opening for an m-entry table communicates $O_{\lambda_{\mathsf{int}}}(\log^2 m)$ field elements and hash words. The gate arguments use a constant number of such openings and logarithmic-round sumchecks per layer, giving $O_{\lambda_{\mathsf{int}}}(\log^2 n)$ communication per layer. The coefficient tables for $\mathsf{Laminate}_{\mathsf{base}}$ add $O(dN)$ elements. Together with the encrypted transcript, this gives $O_{\lambda_{\mathsf{int}}}(dN \log n_{\mathsf{FHE}} + d \log^2 n)$ for $\mathsf{Laminate}_{\mathsf{base}}$.

For the packed variants, the ciphertexts and round commitments contribute $O_{\lambda_{\mathsf{int}}}(r+N)$ when $T = \Theta(\lambda_{\mathsf{int}})$. The packing relation has size polynomial in N, r, so interactive Spartan with these openings communicates $O_{\lambda_{\mathsf{int}}}(\log^2(Nr))$. Using $r = O(d \log n)$ and the table's convention $N \leq n$, the total is $O_{\lambda_{\mathsf{int}}}(d \log^2 n + N)$, including the coefficient openings.