BOCCHI: Compact Blind SNARK via Malformation-Tolerant Ciphertext Aggregation

Ye Yu
School of Computing
National University of Singapore
ye.yu@u.nus.edu

Kexi Huang
School of Computing
National University of Singapore
kexi.huang@u.nus.edu

***Abstract*—Blind SNARKs combine fully homomorphic encryption (FHE) with cryptographic proofs to enable verifiable computation over encrypted data, and can substantially outperform SNARK-over-FHE approaches in efficiency. However, ciphertext openings in existing constructions can inflate proofs to tens or hundreds of megabytes. We present BOCCHI, a compact and efficient blind SNARK that addresses this bottleneck by committing to ciphertexts natively and opening them through a few aggregates. To establish soundness even for malformed ciphertexts, we introduce canonical binding and a lightweight noise filter that achieves it with optimal error, tools that may be of independent interest. We further formulate this transformation as a Commit–Aggregate–Check (CAC) compiler and show its effectiveness on two systems with different PIOPs (GKR and Fractal). At 2^{22} constraints, BOCCHI produces a 6.57 MB proof in 115.3 seconds, $9.3\times$ smaller and $2.54\times$ faster than Phalanx. Its proof is the smallest among five baselines. Our implementation is available at https://anonymous.4open.science/r/BOCCHI-Artifacts.**

1. Introduction

Fully homomorphic encryption (FHE) enables a client to outsource computation on private data to an untrusted server: the server evaluates a function on ciphertexts and never sees the data. FHE alone, however, provides no integrity. A server may skip or alter the computation, which the client cannot tell. Verifiable computation over encrypted data (VCoED) [1], [2] closes this gap by requiring the server to also prove that the encrypted result is correct.

Two routes lead to such proofs. The first proves the homomorphic evaluation itself with a SNARK over the ciphertexts (SNARK-over-FHE) [3], [4], [5]. It must arithmetize modular arithmetic over large rings, rounding, and key switching, and therefore incurs a large overhead. The second, blind SNARKs [6], [7], [8], runs the prover of an ordinary proof system homomorphically on the encrypted witness and lets the client decrypt the prover messages and verify the plaintext relation. Since the proven relation is the plaintext computation rather than its FHE implementation, blind SNARKs are substantially more efficient, and a line of recent systems has made them practical for circuits with millions of gates [9], [10], [11], [12].

The remaining obstacle is the proof size. Every value that the client checks arrives as a ciphertext, which at practical parameters occupies on the order of a hundred kilobytes even after modulus switching, while the value it carries may be a single field element. Existing blind SNARKs open many committed ciphertexts: the blind polynomial commitment of Phalanx [11] opens 307 columns of ciphertexts, blind Fractal [8] opens the queried ciphertexts of every committed oracle, and Laminate [9] sends packed ciphertexts for every GKR round [13] together with additional well-formedness tests. Their proofs thus reach tens to hundreds of megabytes; at 2^{22} constraints, we measure 61 MB for Phalanx, 149 MB for its RAA-code variant [10], and 247 MB for the base variant of Laminate. The client, typically the weakest party in VCoED, must download all of these ciphertexts.

The proof is large because existing blind PCSs treat ciphertexts as opaque symbols: the server encodes the encrypted witness homomorphically, so every opened column consists of whole ciphertexts, which the client must decrypt to check. A ciphertext, however, is just a vector of coefficients over each RNS limb, and these coefficients can be committed natively, as in SNARK-over-FHE [4], [14]. Opened columns then become far smaller and are checked without any decryption. A native commitment, though, binds the server to ciphertexts, whereas the hIOP is sound only with respect to their plaintexts; connecting the two requires the client to decrypt, and decrypting each claimed ciphertext separately brings the per-ciphertext cost back.

Bridging cheaply means checking the claims against a few random linear combinations of the committed ciphertexts rather than against each of them. The usual argument that such a check is sound [15] needs decryption to be linear, but decryption is linear only on well-formed ciphertexts, and nothing forces a malicious server to send them; once the noise of a combination crosses a rounding boundary, its decryption deviates from the combination of the decryptions by a carry that the server can steer. This obstacle is not specific to our setting: it arises whenever the client relies on the decryption of a ciphertext that the server has combined, whatever the combination is for, and recent systems

that combine ciphertexts for quite different purposes have run into it. Gama et al. [8] combine ciphertexts to batch their proof of decryption, which assumes that decryption remains correct on the combination; this turns out to be a proof gap, and with their parameters the proof accepts a value other than the decryption with probability about 2^{-6}. Laminate [9] combines sparse ciphertexts to compress them, and its cross-slot packing allows an adversary to overwrite earlier round messages, as its authors disclose in a revised version; their fix adds noise-checked subset sums that inflate the proof by $2.39\times$. Enforcing well-formedness in general, by certifying the noise of every committed ciphertext, brings back exactly the per-ciphertext cost that combining is meant to remove. Combining ciphertexts thus appears to force a choice between soundness and compactness.

This work. In this work, we show that this choice is a false one. The soundness of a blind SNARK requires only that each committed ciphertext bind the server to a single plaintext fixed before the challenges, which need not be its decryption; every ciphertext, well-formed or not, determines such a canonical value. We introduce BOCCHI (Binding Opening with Compressed Ciphertexts for HE IOP), a compact blind SNARK that commits to ciphertexts natively, coefficient by coefficient, and bridges this commitment to the plaintext claims through a few aggregated ciphertexts authenticated by a native linear argument. A filtered batch check makes the client accept canonical values with an error that we prove optimal, and several prover-side techniques further optimize the performance. Moreover, we distill this transformation into a Commit–Aggregate–Check (CAC) compiler, which applies to blind SNARKs with different PIOPs and HE schemes.

At 2^{22} constraints, BOCCHI produces a 6.57 MB proof in 115.3 seconds on a single core, $9.3\times$ smaller and $2.54\times$ faster than Phalanx. Its proof is the smallest among all baselines at every scale we evaluate, including the GKR-based systems, whose proofs grow with circuit depth. In prover time, it outperforms every baseline except the base variant of Laminate, which runs on BGV, where ciphertext multiplication is about $55\times$ faster in our measurements. Applying the CAC compiler, we estimate that the proof of Laminate shrinks by $2.2\times$ while its prover time halves, and that the publicly verifiable proof of blind Fractal shrinks by $45\times$ at a 7% increase in server time.

1.1. Technical Overview

Hybrid blind PCS. BOCCHI keeps the blind Spartan PIOP and changes how the witness is committed and opened. The server encrypts each row of the witness matrix into a ciphertext c_j and commits to the ciphertext residues with Ligero [16] on each RNS limb, with no homomorphic encoding. To answer the final evaluation query, it first sends claim carriers whose decryptions are the row claims y_j. Fiat–Shamir then fixes multipliers $\beta_{k,j} \in S$, and the server sends K aggregates $D_k = \sum_j \beta_{k,j} c_j$. Multiplying a ciphertext by a public integer is linear on every limb, so the native Ligero opening proves that each D_k is this combination without any FHE arithmetic. The client then checks $\langle \mathsf{Dec}(D_k), u \rangle = \sum_j \beta_{k,j} y_j$ for the public query weights u. The opening thus costs about $K + e$ ciphertexts instead of hundreds of opened columns, and the homomorphic encoding disappears from the prover.

Optimizations. With a compact opening, the cost shifts to repacking the claims and round messages and to the ciphertext multiplications of the first sumcheck. Dual-domain packing repacks them in the coefficient domain, merging ciphertexts during the reduction in the spirit of LWE-to-RLWE conversion [17], with far fewer automorphisms than slot-wise rotate-and-sum and no mask multiplications. A noise-free Gruen method [18] folds the remaining equality factors into public weights that the prover applies anyway, halving the ciphertext products per round without lengthening the critical noise path. Since the packed outputs already lie in a subring, ring switching applies without a trace, and each round then costs little, allowing a later termination to minimize the communication.

Malformation tolerance. Rather than detecting malformed ciphertexts, we make them harmless. Our composition theorem shows that blind soundness needs only one plaintext per ciphertext fixed before the challenges, and Dec is not the only such map. Canonical binding thus asks each claim to match a canonical value determined by the ciphertext itself, which equals its decryption when the ciphertext is well-formed. Our filter theorem shows that the client can achieve this by rejecting any aggregate whose noise exceeds a threshold. For every multiplier the filter admits, the decryption is exactly linear in that multiplier with the canonical value as its slope, and a wrong claim, being a different line, intersects with it at most once. Each repetition therefore errs with probability at most $1/|S|$, which is optimal.

The CAC compiler. Any blind SNARK pays a ciphertext for each opening of a committed encrypted oracle, and the CAC compiler replaces these openings with claims and a few filtered aggregates: it commits to the encrypted oracles with a linear argument over the limbs, fixes the claims before the aggregation challenges, and runs BatchCheck followed by the unmodified plaintext verifier. We prove the soundness of the compiler and instantiate it for Laminate, a GKR-based system over BGV whose outer SNARK already serves as the linear argument, so that only its well-formedness tests change, and for blind Fractal over GBFV, where WHIR [19] on each limb replaces the Merkle commitments. The proof size of both substantially shrinks.

1.2. Contributions

- We present BOCCHI, a compact blind SNARK for R1CS over BFV. BOCCHI combines a hybrid blind PCS with three further optimizations, namely dual-domain packing, a noise-free Gruen method, and ring switching with balanced termination.
- We introduce canonical binding against malformed ciphertexts, which suffices for the soundness of blind SNARKs. We further design an efficient noise filter

TABLE 1. FREQUENTLY USED NOTATION.

Symbol	Meaning
λ	security parameter
n	number of R1CS constraints
N, R	ring dimension; $R = \mathbb{Z}[X]/(X^N + 1)$
t, R_t	prime plaintext modulus; $R_t = R/tR$
Q, q_i, ℓ	ciphertext modulus $Q = \prod_{i\in[\ell]} q_i$
s	secret key
$x(c)$	scaled phase of a ciphertext c
ε, ϵ	noise bound; soundness error
ϕ_g, ρ_x	automorphism $X \mapsto X^g$; rotation by x
$\mathbb{E}$, e	challenge field $\mathbb{E} = \mathbb{F}_{t^e}$ of the PIOP
W, m	witness matrix $W \in \mathbb{F}_t^{m\times N}$
K, S	number of aggregates; multiplier set $S \subseteq \mathbb{F}_t$
$\Lambda, \mathcal{Y}$	comparison map $\Lambda : R_t \to \mathcal{Y}$ on aggregates
κ	noise threshold of the filter

that achieves it with optimal error. These tools may be of independent interest.

- We formulate the CAC compiler, prove its soundness, and apply it to two systems with different PIOPs and HE schemes, reducing their estimated proof sizes by $2.2\times$ and $45\times$.
- We implement BOCCHI end-to-end and conduct extensive evaluation. Compared with the Spartan-based baselines, it proves 2.15–$2.58\times$ faster with 3.35–$22.50\times$ smaller proofs. Our implementation is open source at https://anonymous.4open.science/r/BOCCHI-Artifacts.

2. Preliminaries

2.1. Notation

We show the frequently used letters in Table 1. For a positive integer n, let $[n] := \{1, \ldots, n\}$; logarithms are to base 2. We represent a centered lift of $a \in \mathbb{F}_t$ with $\bar{a} \in (-t/2, t/2)\cap\mathbb{Z}$. The lift requires $\bar{a} \bmod t = a$. Lifts and reductions act coefficient-wise on polynomials and entry-wise on vectors, and we identify a polynomial of degree less than N with its coefficient vector. For $y \in \mathbb{R}^N$, $\lfloor y \rceil$ rounds every entry to the nearest integer, with ties rounded up, $\|y\|_\infty$ is the largest absolute value of an entry, and $\|y\|_\mathbb{Z} := \|y - \lfloor y \rceil\|_\infty$ is the distance from y to the nearest integer vector. For vectors u, v, $\langle u, v\rangle$ denotes their inner product and $u \odot v$ their entry-wise product. For a matrix W, W_j is its j-th row and $W_{j,i}$ its (j, i)-th entry. We write $a \leftarrow S$ for sampling a uniformly from a finite set S.

2.2. Proof Systems

Sumcheck. The sumcheck protocol [20] reduces a claim $H = \sum_{b\in\{0,1\}^\nu} g(b)$ about a ν-variate polynomial g of degree at most δ in each variable to one evaluation of g. In round $i \in [\nu]$, the prover sends the univariate polynomial

$$g_i(X) = \sum_{b\in\{0,1\}^{\nu-i}} g(r_1, \ldots, r_{i-1}, X, b),$$

the verifier checks $g_i(0) + g_i(1) = g_{i-1}(r_{i-1})$, with $g_0(r_0) := H$, and replies with a uniform $r_i \in E$. At the end, the verifier checks $g_\nu(r_\nu) = g(r_1, \ldots, r_\nu)$ with one query to g. The soundness error is at most $\nu\delta/|E|$.

R1CS and Spartan. A rank-1 constraint system (R1CS) over $\mathbb{F}_t$ consists of matrices $A, B, C \in \mathbb{F}_t^{n\times n}$ and is satisfied by a vector $z = (w, 1, io)$, which contains the witness w and the public input and output io, if $(Az) \odot (Bz) = Cz$. The Spartan PIOP [21] proves satisfiability with two sumchecks over a challenge field $\mathbb{E}$. The verifier samples $\tau \in \mathbb{E}^{\log n}$, and the first sumcheck proves

$$0 = \sum_{b\in\{0,1\}^{\log n}} \widetilde{\mathrm{eq}}(\tau, b)\big(\widetilde{Az}(b)\,\widetilde{Bz}(b) - \widetilde{Cz}(b)\big). \quad (1)$$

A naive implementation requires cubic round polynomials. Gruen [18] factors the public $\widetilde{\mathrm{eq}}$ out of the right-hand side,

$$g_i(X) = \widetilde{\mathrm{eq}}(\tau_{<i}, r_{<i})\,\widetilde{\mathrm{eq}}(\tau_i, X)\,h_i(X),$$

where $h_i(X) = \sum_b \widetilde{\mathrm{eq}}(\tau_{>i}, b)\,F(r_{<i}, X, b)$ and $F := \widetilde{Az}\,\widetilde{Bz} - \widetilde{Cz}$. The prover sends only the quadratic h_i, and two evaluations per round suffice.

The sumcheck reduces (1) to claimed values of $\widetilde{Az}$, $\widetilde{Bz}$, and $\widetilde{Cz}$ at a random point r_x. The verifier combines the three claims with random weights, and the second sumcheck proves that the combination equals $\sum_b \widetilde{M}(r_x, b)\,\tilde{z}(b)$, where $\widetilde{M}$ is the same combination of $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$. It ends at a random point r_y, where the verifier needs $\widetilde{M}(r_x, r_y)$, which involves only public data, and $\tilde{z}(r_y)$, which reduces to a single evaluation of $\tilde{w}$ and one of the public io.

IOPs and their compilation. In a public-coin interactive oracle proof (IOP) [22], [23], the prover sends oracles, the verifier replies with random challenges, and finally queries a few oracle entries and decides. In a polynomial IOP (PIOP), such as Spartan's, the oracles are polynomials queried at evaluation points; an IOP-based polynomial commitment turns a PIOP into an IOP. Following Gama et al. [8], we work with holographic IOPs (hIOPs) [24], where the verifier also accesses a preprocessed encoding of the relation. We use round-by-round soundness [25]: in every round, a prover of a false statement can turn the interaction in its favor with probability at most ϵ over the next challenge. The BCS transformation [22] makes an hIOP non-interactive in the random-oracle model by committing oracles with Merkle trees and deriving challenges via Fiat–Shamir [26]. A prover trying up to Q_H candidate messages succeeds with probability at most $Q_H\epsilon + O(Q_H^2 2^{-\lambda})$ [27].

Polynomial commitments and linear arguments. A polynomial commitment scheme (PCS) lets a prover commit to a polynomial and later prove its evaluation at a point chosen by the verifier. A closely related primitive is a linear argument, which proves that vectors sent in the clear are prescribed linear combinations of committed ones. Ligero [16] provides one natively: after committing to a matrix $A \in \mathbb{F}_q^{m\times k}$, the prover shows that a vector D equals $\beta^\top A$ for a verifier-chosen β, at the cost of sending D, one extra row for a proximity test, and L opened columns with their Merkle authentication paths. Setting $\beta = v$ answers the

evaluation query $v^\top Au$, so Ligero is in particular a PCS. Conversely, any multilinear PCS yields a linear argument: a claim $M\mathbf{f} = \mathbf{y}$ is batched by a random combination into $\langle \mathbf{a}, \mathbf{f} \rangle = y$, and a sumcheck [20] on $\sum_{x \in \{0,1\}^\nu} \tilde{a}(x)\tilde{f}(x)$ reduces it to one evaluation of $\tilde{f}$, opened via the PCS, and one evaluation of $\tilde{a}$, computed by the verifier in $O(n)$ time.

2.3. Homomorphic Encryption

BFV. We use the BFV scheme [28], [29] for our main system. BFV operates over $R = \mathbb{Z}[X]/(X^N + 1)$ with N a power of two. Messages lie in $R_t = R/tR$ for a prime t, and ciphertexts are pairs $c = (c_0, c_1) \in R_Q^2$, where $Q = \prod_{i \in [\ell]} q_i$ is a modulus chain composed of distinct primes. In implementation, each ciphertext is stored and computed as ℓ residue limbs [30], since the Chinese Remainder Theorem (CRT) gives $R_Q \cong \prod_{i \in [\ell]} R_{q_i}$.

An encryption $c = \mathsf{Enc}(a)$ of $a \in R_t$ satisfies $c_0 + c_1 s \equiv \lfloor Q/t \rfloor \cdot \bar{a} + v \pmod{Q}$ for a short $v \in R$. We call

$$x(c) := \tfrac{t}{Q}\,(c_0 + c_1 s) \in \mathbb{Q}[X]/(X^N + 1) \cong \mathbb{Q}^N$$

its scaled phase. $\mathsf{Dec}(c) := \lfloor x(c) \rceil \bmod t$ gives the decryption result, and $\|x(c)\|_{\mathbb{Z}} \in [0, \frac{1}{2}]$ represents the noise. Let $\varepsilon < \frac{1}{2}$ bound the noise of every ciphertext committed by an honest server in the protocol. We call a ciphertext well-formed if its noise is at most ε, and malformed otherwise. Nothing forces a server to send well-formed ciphertexts: every pair in R_Q^2 decrypts to some message, and its noise cannot be observed without s.

Homomorphic operations. Computational operations include addition and multiplication, the operands of which include scalars, plaintexts and ciphertexts (we leave automorphism to the next paragraph). For all operations above except ciphertext multiplication, the scaled phase is linear. For example, $x(c + c') \equiv x(c) + x(c')$, and $x(\bar{p}\, c) \equiv \bar{p}\, x(c) \pmod{tR}$ for every $\bar{p} \in R$. Each such operation leads to noise growth: addition adds the noises of the two ciphertexts, and multiplication inflates the noise by at most $\|\bar{p}\|_1$. Other maintenance operations include relinearization, which is invoked after ciphertext–ciphertext multiplication to collapse the ciphertext degree to 2, and modulus switching, which maps a ciphertext to a shorter modulus chain. The scaled phase is not linear for these operations.

Encoding and automorphism. There are two ways to embed a vector in a polynomial. A straightforward method is using the coefficient, which is named coefficient encoding. Another method, SIMD encoding, maps the vector to the polynomial evaluations on roots of unity, enabling element-wise multiplication. The two encoding styles are interchangeable with an NTT. Automorphism permutes the encrypted message based on the encoding method. Let $\phi_g \colon R_t \to R_t,\ a(X) \mapsto a(X^g)$; we write $1 := \phi_1$ for the identity. This directly applies to the message in coefficient encoding; a special automorphism ϕ_{-1} switches the two rows. For SIMD encoding, it is transformed into a rotation $\rho_x(a)_{r,i} = a_{r,i+x}$ with indices modulo $N/2$. Since $(\mathbb{Z}/2N\mathbb{Z})^\times = \langle 5 \rangle \times \langle -1 \rangle$, rotations and automorphisms are identified by $\phi_{5^x} = \rho_x$.

Other schemes. Section 5 also applies to BGV [31] and GBFV [32]. BGV places the message in the low-order bits, $c_0 + c_1 s \equiv \bar{a} + tv \pmod{Q}$, and GBFV generalizes BFV to a polynomial plaintext modulus $t(X)$, such as $X^k - b$, whose slots are small-degree extensions of a large prime field. Appendix A extends the results of Section 4 to both.

2.4. Blind SNARKs

Verifiable computation over encrypted data. In verifiable computation over encrypted data (VCoED), a client encrypts its private input and sends it to a server, which evaluates a public function homomorphically and returns the encrypted result with a proof of its correctness. One approach proves the arithmetic over R_Q, including rounding and key switching, with a SNARK over the ciphertexts (SNARK-over-FHE). Blind SNARKs [6], [7], [8] instead prove the plaintext relation: the server runs the prover homomorphically on the encrypted witness, so the proven relation is over $\mathbb{F}_t$ rather than over R_Q, and rounding and key switching need not be proven.

Blind hIOPs. As in the plaintext setting, a blind SNARK is built in two steps [8]: an hIOP is first made blind, and the blind hIOP is then compiled into an argument. Let Π be a public-coin hIOP for a relation $\mathcal{R}$ over $\mathbb{F}_t$. Its blind version $\mathcal{E}[\Pi]$ takes a statement given as ciphertexts ctx, whose decryption is a statement of Π. Each message that the prover of Π sends is replaced by ciphertexts that encrypt it, either sent directly or committed as an oracle whose entries the verifier queries. Since the challenges are public, the server computes these ciphertexts homomorphically from the encrypted witness, which is efficient when the prover of Π is a low-depth arithmetic circuit in the witness and the challenges. The verifier decrypts the ciphertexts and runs the verifier of Π on the plaintexts. Since Dec fixes the plaintext of a ciphertext as soon as it is sent, $\mathcal{E}[\Pi]$ inherits the round-by-round soundness of Π.

Compiling $\mathcal{E}[\Pi]$ with BCS gives a designated-verifier (DV) blind SNARK [8]. In private proof delegation (zkDel), the client knows the witness and delegates the proving of a publicly verifiable zkSNARK. It decrypts the blind proof, publishes the plaintexts that the verifier of Π reads, and attaches a proof of decryption (PoD) showing that they are the decryptions, under a committed key and with bounded noise, of the ciphertexts in the proof. We refer to this setting as publicly verifiable (PV).

Instantiation of blind SNARKs. Phalanx [11] and Huang et al. [10], whose system we call RAA-Phalanx, instantiate $\mathcal{E}[\Pi]$ with Spartan over BFV. Due to the constraints on the multiplication depth, the table is recomputed from the beginning, which incurs an $O(n \log n)$ complexity. A round value first appears as an encrypted vector whose slots must be summed and packed into few ciphertexts, which takes rotations; we call this response packing. Instead of running all rounds under encryption, the server may terminate early and send the remaining folded tables, which

the client decrypts to finish the sumcheck in the clear [11], [12]; this trades round messages for dense tables.

The final evaluation of $\tilde{w}$ is answered with a blind PCS, for which both systems use an encrypted version of Ligero. The server encodes the rows of the encrypted witness homomorphically with an HE-friendly linear code, Reed–Solomon in Phalanx and generalized RAA codes in RAA-Phalanx, and commits to the columns of the encoded ciphertext matrix with a Merkle tree. To answer the query $v^\top W u$, it must open L columns, each consisting of ciphertexts. The homomorphic encoding restricts the choice of the code, and every opened column adds ciphertexts to the proof, so the openings dominate the proof size.

2.5. Threat Model

We consider a client, a server, and, in the PV setting, an auditor. The client holds the private input and generates the keys honestly. The server receives the public and evaluation keys and the encrypted input, evaluates a public function, and proves the result.

Soundness. The server is malicious. It may deviate arbitrarily from the protocol and choose each message adaptively. In the case of PV, the server and client may collude, or equivalently, the server may access the private key. We require that the client (or the auditor) accepts only if the decrypted statement is in the relation, except with negligible probability, and knowledge soundness wherever the underlying hIOP provides it.

Privacy. The server (and the auditor) must learn nothing about the client's input beyond the public function and the sizes involved. In DV, its view consists only of ciphertexts, public challenges, and the client's decision, so privacy follows from the IND-CPA security of BFV up to the decision bit. The bit is known to enable key-recovery attacks [33], [34]. We model this decision bit as permitted leakage and reuse a key only until the first rejection.

One concern for privacy in PV is that a malicious server can homomorphically cancel or replace the zero-knowledge masks. Therefore, as in any PV blind SNARK that publishes decryptions, the privacy assumes that the server refrains from that, which a client knowing the witness can enforce by recomputing the values before publishing them.

3. BOCCHI: a Compact Blind SNARK

3.1. Overview

Figure 1 shows BOCCHI. The client encrypts its input and the server evaluates the circuit homomorphically and proves it with the blind Spartan PIOP. We change only how the witness is committed and opened, and how the transcripts are computed and transmitted. The server commits to the encrypted witness directly on the ciphertext residues and without any homomorphic encoding. It then runs both sumchecks homomorphically, and the client decrypts their round messages. The PIOP ends with a single evaluation of the committed witness, which the hybrid opening answers. The server first sends a claim, after which the multipliers are sampled by the client for the server to aggregate the committed ciphertexts and prove the aggregation with a native linear argument (①). The client compares the claims with the decrypted aggregates after a noise filter (②). The hybrid blind PCS is detailed in Section 3.2. Three optimizations further accelerate the computation and shrink the PIOP messages: dual-domain packing (A), noise-free Gruen method (B), and ring switching with balanced termination (C). These optimizations are described in Section 3.3. We also describe how to extend BOCCHI to more general scenarios in Section 3.4.

3.2. Hybrid Blind PCS

To construct a blind PCS, previous works directly adapt Ligero to its encrypted version. This necessitates an HE-friendly code and consequently a large number of opened columns. For example, Phalanx [11] uses the Reed–Solomon (RS) code at the price of k sequential multiplications, $O(kn^{1+1/k})$ computation complexity, and 307 opened columns. In comparison, our blind PCS is based on two observations: first, the dense columns, each representing an equality check, can be aggregated soundly with a linear argument; second, the linear relation required by the aggregation can be efficiently proven with one opening of PCS over ciphertexts. Therefore, our design combines the blind claim with native linear arguments over ciphertexts to simultaneously minimize the number of opened ciphertexts and remove the computation load of homomorphic PCS, as shown in Figure 2.

Commitment over the ciphertexts. Let the plaintext witness vector be $w \in \mathbb{F}_t^{mN}$, arranged as $W \in \mathbb{F}_t^{m\times N}$. Encrypting each row j into a corresponding ciphertext c_j renders the blind witness $A \in \mathbb{Z}_Q^{m\times 2N}$, which can be further decomposed into $A^{(i)} \in \mathbb{F}_{q_i}^{m\times 2N}$ along different RNS limbs. For simplicity, we only present the cases where one column of ciphertexts gives optimal performance. When $m \gg 2N$, one can follow the standard technique to rearrange the ciphertexts into an approximate square. Then, the decomposed ciphertexts are committed with Ligero [16].

Opening within the ciphertexts. A multilinear evaluation query factors into public weight vectors $v \in \mathbb{E}^m$ and $u \in \mathbb{E}^N$, where $\mathbb{E}$ is the PIOP challenge field. An honest opening supplies the row claims $y_j = \sum_i u_i W_{j,i}$, from which the client obtains the requested evaluation as $\sum_j v_j y_j$. Therefore, after the query has been fixed, the server homomorphically computes the claim carriers Y. The client then decrypts Y to obtain $y \in \mathbb{E}^m$.

To validate that Y is honestly computed, the two parties execute the following protocol: for $k \in [K]$, the client derives independent uniform scalars $\beta_{k,\cdot} \in S \subseteq \mathbb{F}_t$ and sends their fixed centered lifts $\overline{\beta}_{k,j} \in \mathbb{Z}$. The server forms

$$\hat{c}_k = \sum_{j=1}^{m} \overline{\beta}_{k,j}\, c_j \quad \text{in } R_Q^2,$$

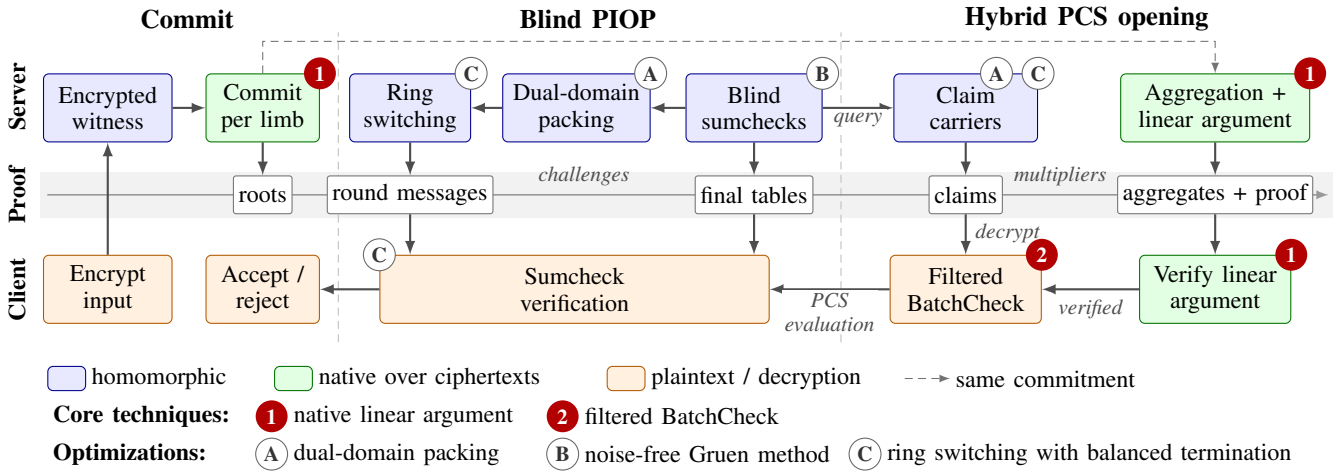

Figure 1. Overview of BOCCHI. The server runs the blind PIOP homomorphically and opens the committed witness with the hybrid PCS; the client holds the key. The middle band is the proof in transcript order; all challenges are derived by Fiat–Shamir, so the aggregation multipliers are fixed only after the claims. Colors mark where each step computes. Markers 1 and 2 locate the core techniques (Sections 3.2 and 4), A–C the optimizations (Section 3.3).

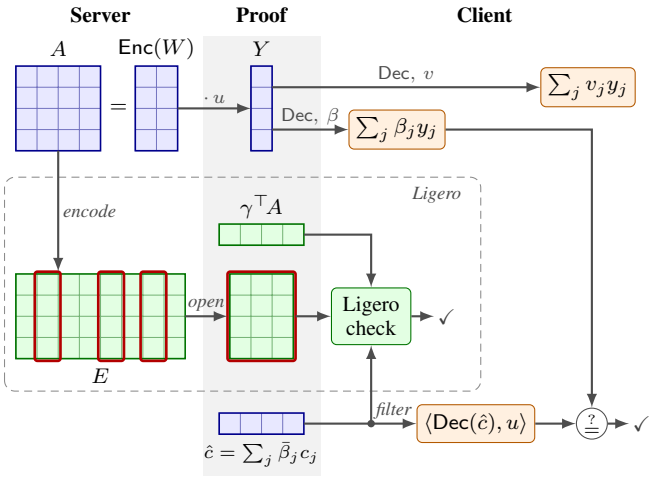

Figure 2. Toy hybrid PCS opening with $m = 4$ ciphertexts of $2N = 4$ residues and a rate-1/2 code. The rows of A are the ciphertexts $c_j = \mathsf{Enc}(W_j)$. The K repetitions and the limbs are not shown.

which serves as a bridge between two worlds. On one hand, $\hat{c}$ is a row combination of the committed ciphertexts and can be efficiently verified by Ligero; on the other hand, $\hat{c}$ can be decrypted to messages that help verify the claim by checking

$$\sum_{j=1}^{m} \beta_{k,j} y_j = \langle \mathsf{Dec}(\hat{c}_k), u \rangle, \forall k \in [K].$$

Once the claim is verified, the client computes the opening $\sum_j v_j y_j$ themselves.

Security analysis. The opening can fail in two ways: (i) the native argument accepts some $\hat{c}_k \neq \sum_j \overline{\beta}_{k,j} c_j$, or (ii) every $\hat{c}_k$ is the prescribed combination, yet the client accepts claims that the commitment did not fix, so that the server could equivocate on the evaluation. Ligero's proximity and consistency errors bound the first term by

$$\epsilon_{\mathsf{LA}} \leq \left(\frac{1+\omega}{2}\right)^{L} + \sum_{i=1}^{\ell} \frac{8N+1}{q_i^{e_i}},$$

where $(\omega, L, \ell, q_i, e_i)$ are the code rate, number of queries, number of ciphertext modulus, the i-th modulus, and its corresponding extension degree, respectively. For the second term, Dec is linear on the aggregates for well-formed ciphertexts, and a wrong claim survives K independent random combinations with probability $|S|^{-K}$. We defer the remaining analysis in cases of malformed ciphertexts to Section 4, which shows that with a lightweight filter, a wrong claim still passes with probability at most $|S|^{-K}$.

Cost analysis. Encoding and committing are cheap in computation in both designs. The communication of our method comprises three parts: the K aggregated ciphertexts, the Ligero opening that contains the opened columns with their Merkle paths and a proximity response, and the claim. For practical scales at about $n = 2^{20}$, the second term is negligible, and the opening therefore costs only $K + e$ ciphertexts, compared to hundreds of ciphertexts of homomorphic PCSs. For large $n > 2^{30}$, Ligero's columns take over to be a bottleneck. However, notice that any PCS that proves linear constraints over $\mathbb{F}_{q_i}$ can replace Ligero. A polylogarithmic one shrinks the second term again, leaving the Kg aggregates and me/Ng claims, where g is the number of ciphertexts grouped in one row. Applying the square root optimization derives a constant advantage factor of $\sqrt{L/K}$ over prior homomorphic PCSs. Under our setting, this is $6.19\times$ over Phalanx, and $10.27\times$ over RAA-Phalanx.

3.3. Optimizations

Dual-domain packing. Throughout the computation, all messages are encoded in a SIMD-style. However, we

identify two processes that can be accelerated through coefficient packing due to its compact representation for scalar responses. To repack the claim, the server needs to sum across all slots for all m ciphertexts, and compress the results to compact ciphertexts; the PIOP round messages go through a similar process. Prior works first compute

$$c_j^{sum} = (1+\phi_{-1}) \prod_{i=0}^{\log N-2} \left(1+\rho_{2^i}\right) c_j, \qquad \forall j \in [m],$$

and then extract values with binary masks and sum the results by $Y = \sum_j p_j c_j^{sum}$. However, this incurs $em \log N$ rotations for the repacking process and makes it one of the bottlenecks for the server. Our observation is that repeated rotations are unnecessary because the first element in the coefficient domain is already the sum. Inspired by the LWE-to-RLWE conversion by [17], we propose to merge the ciphertexts during reduction. For simplicity, we assume $m = 2^h \leq N$ with h being integral. Starting from $c_j^{(0)} = c_j$, for $l \in [h]$ and $0 \leq j < m/2^l$, first compute

$$c_j^{(l)} = \left(c_j^{(l-1)} + X^{\frac{N}{2^l}} c_{j+\frac{m}{2^l}}^{(l-1)}\right) + \phi_{2^l+1}\left(c_j^{(l-1)} - X^{\frac{N}{2^l}} c_{j+\frac{m}{2^l}}^{(l-1)}\right),$$

where X^a represents simply multiplying the ciphertext polynomials with a monomial and only requires reordering the coefficients. Then, since only one ciphertext remains, the server directly computes

$$Y = \prod_{l=h+1}^{\log N} \left(1+\phi_{2^l+1}\right) c_0^h = \sum_{j=0}^{m-1} X^{jN/m}\, c_j^{sum}.$$

The client decrypts the claim in the coefficient domain. This method only requires about $e(m + \log(N/m))$ automorphisms, which is nearly $14\times$ fewer in our case compared to the straightforward method. Moreover, it circumvents the noise growth caused by mask multiplications.

Noise-free Gruen method. Gruen's optimization [18] removes the public factor $eq(\tau_{<i}, r_{<i})eq(\tau_i, X)$ from the round polynomial, so each ciphertext round sends two values of a quadratic. However, under FHE, the remaining factors $eq(\tau_{>i}, u)$ and $eq(\tau_{slot}, s)$ must still be applied homomorphically, and applying them as a separate plaintext multiplication on each product, as a direct implementation does, adds one plaintext factor to the noise. To tackle this problem, we instead merge $eq(\tau_{>i}, u)$ into the public weights that rebuild A and C from the original ciphertexts in each round, and apply $eq(\tau_{slot}, s)$ once per round value after summing over pairs. Since the noise growth of a plaintext multiplication depends only on the size of its representative, not on how many public factors it combines, every message carries the same plaintext factors as without the split, while the ciphertext products per round are halved. Since the messages lie in the critical path of the noise, this optimization enables a more efficient modulus chain.

Ring switching and balanced termination. Ring switching [35] maps a ciphertext whose plaintext lies in the subring $R' = \mathbb{Z}_t[X^d]$ to dimension $N' = N/d$ and effectively shrinks the communication. A general SIMD-encoded ciphertext, however, must first be moved into R' by the trace

$$\mathrm{Tr}_{R/R'} = \prod_{l=\log N'+1}^{\log N} \left(1+\phi_{2^l+1}\right),$$

which costs $\log d$ automorphisms and multiplies the noise by up to d. However, recall that our dual-domain packing lands in the coefficient domain and has all the automorphisms applied if $m \leq N'$. Therefore, the output already lies in R', and ring switching applies directly. Since our method avoids the trace and the noise growth, it allows for a smaller modulus, and consequently a smaller subring.

This also shifts the balance of early termination [11], [12], which trades round messages for the dense tables left to the client. Once a round message costs one small ciphertext, each additional round halves the tables almost for free in proof size. Therefore, we stop with 2^{11} and 2^{12} entries left for the first and second sumcheck, respectively. The tables themselves stay dense and are sent in the large ring.

3.4. Extension to General Scenarios

BOCCHI serves VCoED in which the server has no private input, but it can be easily extended to other scenarios, including VCoED with private input, such as model weights, or zkDel. In both, Spartan and the hybrid PCS are made zero-knowledge by standard masking: a masking polynomial for each sumcheck [36], [37] and random padding of the witness and the constraint system. For servers with private input, since the client holds s, noise flooding is required, enlarging the modulus but leaving the protocol unchanged. For the same reason, the native PCS must be hiding. The private input is then committed once with another PCS over $\mathbb{F}_t$. For private proof delegation, as in Gama et al. [8], the client publishes the plaintexts it decrypts with a PoD under its committed key.

4. Malformation Tolerance via Lightweight Noise Filtering

The hybrid PCS of Section 3 never decrypts a committed ciphertext on its own; the client only decrypts K aggregates and compares them with earlier claims. This section shows when this is sound. Aggregated decryption fails in three different ways in existing designs (Section 4.1). We replace the unattainable goal of consistency with Dec by canonical binding (Section 4.2), prove that a single noise filter achieves it with optimal error (Section 4.3), and show that canonical binding suffices for blind SNARKs (Section 4.4). These tools may be of independent interest.

4.1. Pitfalls of Aggregated Decryption

A server can produce malformed ciphertexts without the key, for example by adding $\lfloor Q/2t \rfloor$ to c_0. For an aggregate $\hat{c} = \sum_j \bar{\beta}_j c_j$, the phase is linear but its rounding is not:

$$\mathsf{Dec}(\hat{c}) = \sum_j \beta_j \mathsf{Dec}(c_j) + (\lfloor \sigma \rceil \bmod t),$$

where $\sigma := \sum_j \bar{\beta}_j \big(x(c_j) - \lfloor x(c_j) \rceil\big)$. We call $\lfloor \sigma \rceil$ the carry. It vanishes whenever $\|\sigma\|_\infty < \frac{1}{2}$, in particular for honest inputs with $\sum_j |\bar{\beta}_j| \varepsilon < \frac{1}{2}$, but a malformed ciphertext lets it depend on the multipliers. This loophole underlies several soundness bugs and attacks in existing schemes.

Cross-slot packing in Laminate. To compress thousands of ciphertexts produced during the GKR protocol, Laminate [9] multiplies each ciphertext with a plaintext mask and sends their sum $\sum_u^R P_u \cdot c_u$. The verifier reads the message of each GKR round from the slots agreed in advance, assuming $\mathsf{Dec}(P_u \cdot c) = P_u \odot \mathsf{Dec}(c)$. However, as its authors disclose in a revised version, this step invalidates the whole system. An adversary can forge an arbitrary ciphertext after they have received all challenges by setting the last ciphertext as $P_R^{-1}(c^* - \sum_u^{R-1} P_u \cdot c_u)$ with most masks invertible in R_Q, where c^* encrypts all intended messages. Their fix adds λ random $\{-1, 0, 1\}$ subset sums whose noise is checked, each catching a noisy ciphertext with probability at least $\frac{1}{2}$, while inflating the proof by $2.39\times$.

Batched decryption proofs in Gama et al. For public verifiability, Gama et al. [8] publish the queried plaintexts with one PoD for a random combination $\sum_i \alpha_i \mathsf{ct}_i$ with dense $\alpha_i \in R_t$. Their soundness proof relies on decryption remaining correct on the combination, which the stated noise bound does not guarantee. An adversary can construct a malformed ciphertext c whose phase is that of an honest ciphertext shifted by $\frac{1}{2}(1 + X + \cdots + X^{N-1})$, such that for every α whose lift has an even coefficient sum, $\alpha \cdot c$ has honest noise and decrypts to $\alpha \cdot m'$ for a fixed $m' \neq \mathsf{Dec}(c)$. The batched PoD thus accepts m' with probability about 2^{-6} for their GBFV parameters. Although this does not directly invalidate the system, it reveals a substantial gap during the paper's soundness proof.

Naive aggregation in our system. Suppose the verifier of our system naively accepts the aggregated ciphertexts as well-formed. Then, an adversary can raise the probability of equivocation well above $2^{-\lambda}$ with a malformed ciphertext whose phase, in a single coefficient, lies between two adjacent fractions with denominators of order $\sqrt{t}$, making the carry follow two different linear functions on two sets of multipliers. Therefore, two different claims for that ciphertext each pass a repetition with probability over $1/\sqrt{t}$. In our parameter setting where $\log t \approx 20$, $K = 8$ repetitions only give about 2^{-75} security.

Our insight. Naive aggregated decryption fails to meet the supposed security. Yet, enforcing a zero carry means certifying the noise of every ciphertext, with a detector as in Laminate or one PoD per ciphertext, incurring considerable per-ciphertext cost that our aggregation is meant to remove. Our insight is that a malformed ciphertext is harmless as long as its value is fixed before the challenges, to enforce which a lightweight noise filter applied on the aggregated ciphertext by the verifier suffices.

4.2. Canonical Binding

To formalize a malformed yet acceptable ciphertext, we propose the definition of canonical binding for any protocol in which a verifier accepts claims about the values of committed ciphertexts. Let $\{c_i\}_{i \in [m]}$ be ciphertexts fixed by a commitment. A query is a pair $v = (j_v, \Lambda_v)$ of an index $j_v \in [m]$ and an $\mathbb{F}_t$-linear map $\Lambda_v : R_t \to \mathcal{Y}$, which asks for Λ_v applied to the plaintext of c_{j_v}; full comparison is $\Lambda_v = 1$. A blind opening for a set V of queries is a Σ-shaped protocol in which the prover justifies the claim $y_v \in \mathcal{Y}$ for every $v \in V$. We call a blind PCS canonically binding if its opening is, i.e., if every accepted evaluation equals the evaluation of the plaintext polynomial formed by the canonical values of its committed ciphertexts.

Definition 1 (Canonical binding)**.** A blind opening is ϵ-canonically binding if, given the parameters and keys, there exists a deterministic map $\mathsf{Can} : R_Q^2 \to R_t$ such that for all $\{c_j\}_{j \in [m]}$, $\{y_v\}_{v \in V}$, and V with $y_v \neq \Lambda_v(\mathsf{Can}(c_{j_v}))$ for some $v \in V$, the verifier accepts with probability at most ϵ.

Decrypting every queried ciphertext is a 0-canonically binding opening with $\mathsf{Can} = \mathsf{Dec}$, while paying one ciphertext per query. Laminate's subset sum test also keeps $\mathsf{Can} = \mathsf{Dec}$, at the price of λ combinations. The naive aggregated check of Section 4.1 is cheap but at best $O(t^{-1/2})$-canonically binding per repetition under full comparison, with no known guarantee once Λ compresses (Appendix B). Dropping consistency with Dec on malformed ciphertexts is what allows an aggregated opening to be both cheap and sound, and Section 4.4 shows that nothing is lost, since blind soundness only needs a plaintext per ciphertext that is fixed before the challenges. This gives the following construction:

Definition 2 (BatchCheck)**.** Let $S \subseteq \mathbb{F}_t$, $\kappa > 0$, and K be fixed in advance. On committed ciphertexts $\{c_j\}_{j \in [m]}$ and claims $\{y_j\}_{j \in [m]} \in \mathcal{Y}^m$:

1) the verifier samples $\beta_{k,j} \leftarrow S$ for $k \in [K]$, $j \in [m]$;
2) the prover computes $\hat{c}_k = \sum_j \bar{\beta}_{k,j} c_j$ and proves it with a sound linear argument;
3) the verifier accepts if and only if $\|x(\hat{c}_k)\|_{\mathbb{Z}} \leq \kappa$ and $\Lambda(\mathsf{Dec}(\hat{c}_k)) = \sum_j \beta_{k,j} y_j$ for all k.

BatchCheck extends the naive aggregated check with a noise filter, the first condition of step 3, which restricts the scaled phase to a narrow interval at no extra cost for the client. We now prove that BatchCheck is canonically binding with optimal error for some well-chosen κ.

4.3. The Filter Theorem

Let $\Delta(S) := \{\bar{\beta} - \bar{\beta}' : \beta, \beta' \in S\} \setminus \{0\}$. Let $M_S := \max_{\beta \in S} |\bar{\beta}|$ and $M_D := \max_{\delta \in \Delta} |\delta|$. We require

$$\text{(P1)} \quad 4 M_D \kappa < 1, \qquad \text{(P2)} \quad m M_S \varepsilon \leq \kappa.$$

For honest inputs, the carry vanishes because $\|\sigma\|_\infty \le mM_S\varepsilon \le \kappa < \frac{1}{2}$, and the aggregate passes the filter and decrypts correctly. BatchCheck is thus perfectly complete.

For soundness, we construct the following Can, which depends only on c, S, κ, and the secret key:

Definition 3 (Canonical value)**.** Denote $x(c)$ with x. Let $\mathcal{D}_c := \{\delta \in \Delta(S) : \|\delta(x - \lfloor x \rceil)\|_{\mathbb{Z}} \le 2\kappa\}$. If $\mathcal{D}_c \neq \emptyset$, let $\mathsf{Can}(c) := \lfloor x \rceil + \psi \bmod t$, where $\psi := \lfloor \delta(x - \lfloor x \rceil) \rceil / \delta$ for any $\delta \in \mathcal{D}_c$; otherwise let $\mathsf{Can}(c) := \mathsf{Dec}(c)$.

Note that this definition is invariant for different δ: for $\delta, \delta' \in \mathcal{D}_c$, write $\delta(x - \lfloor x \rceil) = z + e$ with $\|e\|_\infty \le 2\kappa$ and $z \in \mathbb{Z}^N$. Then, the vector $\delta' z - \delta z' = \delta e' - \delta' e$ has $\delta' z - \delta z'$ integral, while $\|\delta e' - \delta' e\|_\infty \le 4M_D\kappa < 1$. Hence, ψ is independent of δ. Besides, since $|\delta| < t$, its denominator is coprime to t, $\psi \mod t$ is well-defined.

Next, we prove that BatchCheck is canonically binding under this construction. For simplicity, we only present the proof for scalar multipliers over BFV, the case that BOCCHI uses. In fact, the results generalize to every cyclotomic ring in which t generates a square-free ideal, and dense multipliers. We provide the full proof in Appendix A, and a toy example illustrating the intuition in Figure 3.

Theorem 4 (Filter theorem)**.** BatchCheck *is an* $|S|^{-K}$*-canonically binding opening.*

Proof. The K repetitions use independent multipliers on fixed ciphertexts and claims, so it suffices to show that one repetition accepts with probability at most $1/|S|$. Let j^* be an index with $y_{j^*} \neq \Lambda(\mathsf{Can}(c_{j^*}))$, and fix all multipliers except β_{j^*}. For two accepted values $\beta \neq \beta'$, first consider $\hat{c} - \hat{c}' = \delta \cdot c_{j^*}$. Since $\hat{c}$ and $\hat{c}'$ pass the filter, the equation remains linear for decryption. Therefore,

$$\mathsf{Dec}(\hat{c}) - \mathsf{Dec}(\hat{c}') = \mathsf{Dec}(\delta \cdot c_{j^*}) = \delta \cdot \mathsf{Can}(c_{j^*}),$$

where the last equation can be inferred from Definition 3. Moreover, since both pass the aggregate check, subtracting the two comparisons and using the $\mathbb{F}_t$-linearity of Λ gives

$$\Lambda(\mathsf{Dec}(\hat{c}) - \mathsf{Dec}(\hat{c}')) = \sum_j (\beta_j - \beta'_j) y_j = \delta \cdot y_{j^*}$$

Combining the two equations gives $\delta(y_{j^*} - \Lambda(\mathsf{Can}(c_{j^*}))) = 0$. Since $\delta = \beta - \beta' \neq 0$, $\Lambda(\mathsf{Can}(c_{j^*})) = y_{j^*}$, which demonstrates a contradiction. □

This error is optimal for multipliers from $S \ni 0$: even if all ciphertexts are well-formed, a claim with one wrong entry passes whenever $\beta_{j^*} = 0$, i.e., with probability $1/|S|$. The filter thus tolerates malformed ciphertexts at no cost in soundness error.

The tradeoff. For $S = [-b, b] \cap \mathbb{Z}$, (P1) and (P2) give $\log_2(1/\varepsilon) > 3 + \log_2 m + 2\log_2 b$ bits of noise headroom, compared to $1 + \log_2 m + \log_2 b$ bits for the aggregated ciphertexts to be well-formed. A larger multiplier set tightens the noise budget, but reduces the number of repetitions required since the error per repetition drops to $1/(2b+1)$. Therefore, our strategy is to select the largest possible multiplier set as long as the modulus chain is not modified. We also show that this filter outperforms its unfiltered counterparts even when assuming its unproven security of $\sqrt{1/b}$ in Appendix B.

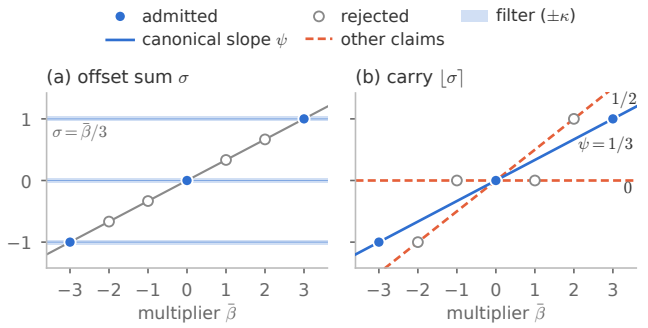

Figure 3. A toy example for one coefficient of a malformed c_{j^*} with offset $\frac{1}{3}$ and multipliers $\bar{\beta} \in \{-3, \dots, 3\}$, all other ciphertexts being honest. (a) The filter admits $\bar{\beta}$ only when the offset sum σ lies within κ of an integer. (b) A claim y passes exactly where the carry lies on the line of slope y, read modulo t. Without the filter, the claims 0, $\frac{1}{3}$, and $\frac{1}{2}$ each pass for 3 of the 7 multipliers; with it, only the admitted points remain, which lie on the line of slope $\psi = \frac{1}{3}$.

4.4. Composition Theorem

We now formalize the intuition that canonical binding is all a blind proof needs. Let $\mathcal{E}[\Pi]$ be a blind hIOP as in Section 2.4, and let T be a set of committed oracle entries that the verifier of Π reads only through a map Λ fixed before any claim. In $\mathcal{E}[\Pi]_{\mathrm{BC}}$, instead of decrypting T directly, the verifier receives claims for them after all IOP rounds and accepts if and only if BatchCheck on T and the verifier of Π, which is based on the claims, both accept.

Theorem 5 (Composition)**.** *Let* Π *be round-by-round (knowledge) sound with error* ϵ_Π*, and let* BatchCheck *be* ϵ_{bind}*-canonically binding with map* Can*. Then* $\mathcal{E}[\Pi]_{\mathrm{BC}}$ *is round-by-round (knowledge) sound with error* $\max(\epsilon_\Pi, \epsilon_{\mathrm{bind}})$.

Proof sketch. Round-by-round soundness of Π is witnessed by a set of doomed partial transcripts. We call a blind transcript doomed if it becomes a doomed transcript of Π when the entries of T are read with Can and all other ciphertexts with Dec. Both maps act on individual ciphertexts and are fixed in advance, so the plaintext of every prover message is determined before the challenge of its round, and the first two properties carry over. This is the argument of Gama et al. [8] for Dec, which uses only that Dec is such a map and never its homomorphic correctness; any other fixed map works equally well. The claims and the multipliers form one extra round. If every claim equals $\Lambda(\mathsf{Can}(\cdot))$ of its entry, the verifier of Π sees a complete doomed transcript and rejects; otherwise, since the claims precede the multipliers, BatchCheck accepts with probability at most ϵ_{bind}. Appendix D gives the full proof. □

5. The Commit–Aggregate–Check Compiler

Given the general and sound BatchCheck protocol as described in Section 4.3, the cost that BOCCHI removes is not specific to Spartan or BFV because every blind SNARK pays a ciphertext for each opening of a committed encrypted oracle. The hybrid PCS of Section 3.2 removes this cost in three steps, which give the compiler its name: it commits to the encrypted oracles natively with a linear argument over the RNS limbs, fixes a claim for every queried entry and aggregates the queried ciphertexts into a few random combinations, and checks the claims against these aggregates with BatchCheck. None of these steps depends on the PIOP, and the analysis of Section 4 extends to BGV and GBFV, as shown in Appendix A. We formulate this transformation as the Commit–Aggregate–Check (CAC) compiler and prove its soundness in Section 5.1, and apply it to Laminate and blind Fractal, which differ from BOCCHI in both the PIOP and the HE scheme in Section 5.2.

5.1. Compiler and Its Soundness

Given an instantiation of BatchCheck and a sound linear argument LA that proves that given vectors are prescribed linear combinations of committed ones, the compiler applies to any public-coin, non-adaptive blind hIOP $\mathcal{E}[\Pi]$ whose verifier reads every queried entry through a map Λ fixed before the claims.

The compiled protocol runs $\mathcal{E}[\Pi]$ with three changes. First, each oracle is committed with LA. Second, after the queries, the prover only delivers the claims, which are bound to their canonical values, instead of opening them. Third, the prover sends K aggregates of the queried ciphertexts with an LA proof, and the verifier checks the claims against them with BatchCheck.

Theorem 6 (Compiler)**.** *Let* Π *be round-by-round (knowledge) sound with error* ϵ_Π*, let* LA *have extractable binding commitments and knowledge error* ϵ_{LA}*, and let* BatchCheck *be* ϵ_{bind}*-canonically binding. The compiled designated-verifier protocol has (knowledge) soundness error at most* $\max(\epsilon_\Pi, \epsilon_{\mathrm{bind}}) + \epsilon_{\mathsf{LA}}$.

Proof. A first hybrid aborts whenever LA accepts an aggregate that is not the prescribed combination of the committed ciphertexts, which the extractable commitments fix before the queries; this costs ϵ_{LA} and leaves the aggregation ideal, as in Section 4.4. The remaining protocol is the BCS transformation of $\mathcal{E}[\Pi]_{\mathsf{BC}}$ with oracles committed by LA, which the BCS analysis permits since it uses only binding and extractability. By Theorem 5, $\mathcal{E}[\Pi]_{\mathsf{BC}}$ is round-by-round sound with error $\max(\epsilon_\Pi, \epsilon_{\mathsf{bind}})$. Appendix E gives the full proof. □

5.2. Application to Other Systems

Table 2 lists the three instances. Specifically, applying the compiler to blind Spartan over BFV with Ligero as LA yields BOCCHI. Since the verifier of the underlying PIOP runs unchanged on plaintext claims, the compiler applies to linear and nonlinear verifiers alike. We describe the compiled protocols briefly here and estimate their costs in Section 6.3.

TABLE 2. THREE INSTANCES OF THE CAC COMPILER. FOR BLIND FRACTAL, e_s AND $\mathbb{F}$ ARE THE INDICATOR OF SLOT s AND THE GBFV SLOT FIELD, RESPECTIVELY.

	BOCCHI	Laminate [9]	Blind Fractal [8]
PIOP	Spartan	GKR	Fractal
HE scheme	BFV	BGV	GBFV
Committed	witness	round messages	FRI oracles
Linear arg.	Ligero	outer SNARK	WHIR
Multipliers	$\mathbb{F}_t$	$[-b, b] \cap \mathbb{Z}$	$\mathbb{F}\, e_s$, $[-b, b] \cap \mathbb{Z}$
Comparison	via Λ_u	full	full
Verification	DV	DV	PV

Laminate. Laminate [9] runs a GKR prover under BGV and commits to each round message, flattened into a ciphertext c_u, with a hash. Besides c_{final} and the subset sums of Section 4.1, it sends an outer SNARK proving in-circuit that both are the claimed combinations of the committed ciphertexts. For each masked ciphertext $c'_u := P_u \cdot c_u$, the claim y'_u equals $\mathsf{Dec}(c_{\mathsf{final}})$ on the slots of P_u and zero elsewhere. Thus, Laminate is already a hybrid in our sense, with its outer SNARK and c_{final} serving as the linear argument and the claims, respectively.

Our instance keeps the outer SNARK and replaces only the subset sums. After all commitments and c_{final}, independent multipliers $\beta_{k,u} \in S$ are sampled. The prover sends $c^*_k := \sum_u \bar{\beta}_{k,u} c'_u$ for $k \in [K]$, which the outer SNARK proves exactly as it proved the subset sums, and the verifier filters each c^*_k and checks $\mathsf{Dec}(c^*_k) = \sum_u \beta_{k,u} y'_u$. This is a BatchCheck with full comparison, so by Theorem 4 and Lemma 12, $y'_u = \mathsf{Can}(c'_u)$ holds except with probability $|S|^{-K}$. Hence $\mathsf{Dec}(c_{\mathsf{final}})$ on the slots of P_u is a deterministic function of c_u alone, fixed when round u is committed, which is exactly the round isolation that Laminate needs, so the attack of Section 4.1 is caught.

Blind Fractal. Gama et al. [8] adapt Fractal to GBFV ciphertexts and compile it with BCS. Their PV proof therefore carries every queried ciphertext with its authentication path, in addition to the published values and the batched PoD of Section 4.1.

Our instance changes the commitments, the openings, and the PoD. (1) One WHIR [19] instance per limb replaces each Merkle tree; since WHIR proves linear constraints over prime fields, it serves as the linear argument. (2) After the queries, the prover packs every queried value into a few dense ciphertexts Y, which is sent as the claim. (3) For a queried position $v = (o, j, s)$, namely the s-th slot of the j-th ciphertext $c_{o,j}$ of oracle o, multipliers $\beta_{k,v} \in \mathbb{F}$ are sampled, and the prover sends $\hat{c}_k = \sum_v L(\beta_{k,v} e_s)\, c_{o,j}$ for $k \in [K]$, where e_s is the indicator of slot s and L a short lift. The verifier then executes BatchCheck. Lifting each position separately makes the multiplier set depend only on

its slot, so each canonical value depends only on its oracle entry, as Definition 3 requires. Binding then follows from Theorem 11, a single-slot version of the filter theorem, with error $1/|\mathbb{F}|$ per combination.

For PV, the client publishes the claims y. Since Y fixes its canonical values before β, it suffices to bind y to them: the transcript including y yields multipliers $\beta'_{k,\ell} \in [-b', b'] \cap \mathbb{Z}$, which act as the same scalar on every slot of Y, and anyone can compute $\hat{c}'_k = \sum_\ell \beta'_{k,\ell} Y_\ell$ without a further round trip. One PoD under the committed key then certifies, for every $\hat{c}_k$ and $\hat{c}'_k$, its decryption to the public expected value and a noise bound satisfying (P1). We modify the PoD to its precise version [38], since the slack of an approximate norm proof is too large for (P1). The bound of Theorem 6 thus grows by ϵ_{PoD} and the $(2b'+1)^{-K'}$ error of the $\hat{c}'_k$ check, as proven in Corollary 10 (II).

6. Evaluation

6.1. Experimental Setup

We implement BOCCHI end-to-end based on the code of [10], and evaluate its efficiency and compactness with extensive experiments. All experiments run on a single core of an AMD Ryzen 9 8945HX with 16GB of RAM. We use Microsoft SEAL [39] with Intel HEXL acceleration [40] for all FHE operations and all baselines with codes available. We set the modulus chain to $[36, 60, 60]$ with a 60-bit special prime and $N = 2^{14}$ to achieve 128-bit security, as indicated by the Homomorphic Encryption Standard [41]. The switched ring has $N' = 2^{11}$ and only a 36-bit limb and a 17-bit prime. We use a 20-bit plaintext modulus with $S = \mathbb{F}_t$ and set $\kappa = 2^{-22}$, under which the worst-case noise $\varepsilon = t(1+N)/(2Q) < 2^{-63}$. Therefore, both (P1) and (P2) hold. We also let $e = K = 8$ to satisfy 128-bit soundness. All baselines are modified to achieve the same level of security and soundness. For the generalized systems with our CAC compiler, we present an estimate of the prover time and proof size.

Baselines. We include five baselines: Phalanx [11], RAA-Phalanx [10], blind Fractal [8], Laminate [9], and Lasagne [12]. Phalanx and RAA-Phalanx share our PIOP, so we evaluate them in the same codebase as ours. For Laminate[1] and blind Fractal, we run their released code on our scale and parameters, within which the former is an end-to-end implementation, and the latter is an estimation script. We also adapt the implementation of Laminate-base to BFV to serve as a calibration point for GKR-based methods, whose FHE backend is BGV. Since Lasagne reports numbers based on theoretical projection and has no public implementation, we evaluate its own cost model and calibrate the result with their reported $1.7\times$ gap between estimation and evaluation. We mark all the estimated results with †.

The GKR-based baselines, Laminate and Lasagne, inherit the dependency on the structure of the payload. The proof size of both grows with circuit depth. Moreover, Laminate assumes perfectly layered circuits with its prover time sensitive to residual connections. Lasagne handles such wires at the price of a fixed cost per layer in prover time. We thus evaluate both on a degree-8 polynomial, a representative 3-layer payload between their best and worst cases. The scales of the circuits are such that its R1CS has the same number of constraints as other systems. For the Spartan-based systems, including ours, we use random R1CS instances that have dependency across different slots to show the generality. On the degree-8 polynomial, BOCCHI's results only differ by less than 1%.

Experimental scope. There are three stages for the VCoED server: payload computation, witness packing, and proof generation. We reasonably assume that during payload computation, all slots are utilized, which is prioritized and has already been achieved in most FHE-based applications, such as PPML [42], [43], [44], [45] and PIR [46], [47]. Without this assumption, SIMD encrypted computation itself would be highly impractical, and the server should switch to other schemes such as TFHE [48].

Under this assumption, the payload computation accounts for only 1–6% of the proof generation time, making any runtime differences caused by the slight difference in moduli even less significant. Furthermore, since each witness has already been compactly generated during the computation process, there is no need to pack them. Therefore, in our evaluation, we only measure the proof generation time. We also note that the assumption is highly favorable for GKR-based methods since their cost is closely related to the number of FHE gates.

6.2. Main System

Comparison with baselines. We show our main results in Table 3. Compared to blind Fractal, BOCCHI is over $800\times$ faster with a 16.44–$20.45\times$ smaller proof. Among Spartan-based methods, BOCCHI achieves a consistent 2.15–$2.58\times$ acceleration and 3.35–$22.50\times$ proof compression. We analyze the performance breakdown in both prover time and proof size over these baselines in detail in the paragraphs below. Compared to Laminate-base, BOCCHI is 3.73–$4.26\times$ slower in exchange for a proof shrunk by 19.59–$37.43\times$. However, this efficiency gap is mainly due to the $55\times$ latency discrepancy of ciphertext multiplication between BFV and BGV, as indicated by its BFV-based variant. After aligning the scheme to BFV, Laminate-base would become 1.45–$2.42\times$ slower than us. Laminate-fold aggregates the ciphertexts with an outer SNARK, while inflating the prover time to over 5000 seconds, which is at least $47\times$ slower than us. Lasagne is also 1.10–$5.72\times$ slower than us with a 1.59–$2.24\times$ larger proof. Similar to Laminate-base, our efficiency advantage becomes more significant when the backend FHE schemes are aligned. To conclude, our method represents a promising Spartan-based technical route that is suitable for general FHE circuits, as a complement to the GKR-based methods.

1. Laminate-fold is extrapolated by small-scale experiments because it requires over 400GB of memory.

TABLE 3. PROOF SIZE (MB) AND PROVER TIME (S) AT 2^{18}, 2^{20}, AND 2^{22} R1CS CONSTRAINTS.

Scale	Phalanx		RAA-Phalanx		Blind Fractal†		Laminate-base			Laminate-fold†		Lasagne†		Ours	
	Time	Size	Time	Size	Time	Size	Time_{BGV}	Time_{BFV}	Size	Time	Size	Time	Size	Time	Size
2^{18}	21.4	19.7	21.1	19.7	$\geq 10^4$	97.0	2.3	14.2	115.1	≥ 5000	44.0	56.1	9.4	9.8	5.9
2^{20}	75.9	51.8	75.5	51.8	$\geq 3 \cdot 10^4$	115.8	8.1	66.4	181.2	≥ 5000	45.2	75.5	12.0	30.5	6.1
2^{22}	292.9	61.0	297.0	148.6	$\geq 10^5$	135.0	30.9	278.6	247.3	≥ 5500	46.7	127.1	14.8	115.3	6.6

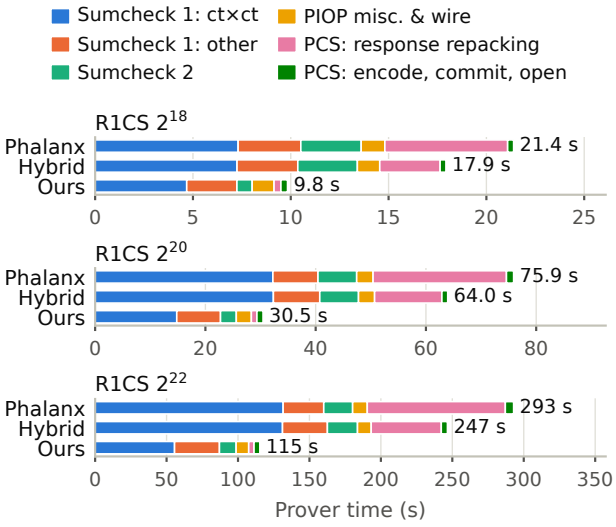

Figure 4. Prover time breakdown at R1CS sizes 2^{18}, 2^{20} and 2^{22}. Hybrid represents our PCS without the optimizations of Section 3.3.

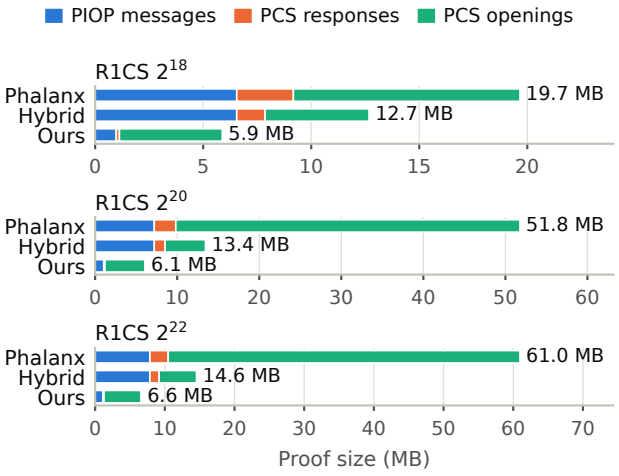

Figure 5. Proof size breakdown. PCS openings are the opened ciphertext columns and Merkle paths for Phalanx, and the Ligero proof (combinations, queried columns and paths) in Hybrid and ours.

Prover time decomposition. We present a fine-grained prover time decomposition in Figure 4. Two main bottlenecks exist in the decomposition of Phalanx: the ciphertext multiplications in the first sumcheck and the rotations of response packing. In contrast, the PCS encoding stage, either done in ciphertext in Phalanx or in plaintext in Hybrid and ours, contributes little to the overall latency. Applying our hybrid PCS reduces half of the rotations by eliminating the need for proximity test, therefore accelerating the proof generation by about $1.20\times$. Furthermore, merging the equality weight allows us to apply Gruen's method without inflating the modulus chain, leading to a 1.55–$2.36\times$ speedup in the ciphertext multiplication; on the other hand, our dual-domain packing technique reduces the latency of response packing to a minor proportion of 3–4%. In total, our tailored optimizations achieve an additional 1.83–$2.15\times$ speedup over the hybrid method.

Proof size decomposition. As shown in the proof size breakdown in Figure 5, the opened ciphertexts in PCS dominate the communication in Phalanx. Our hybrid PCS substantially compresses this opening by 3.33–$16.02\times$. The large range of compression rate originates from the fact that in Phalanx, the proximity test degenerates to opening all columns at the scale of 2^{18} and 2^{20}. Therefore, their opening grows linearly at this point, while our opening remains constant due to the large N used. With our compact opening, the bottleneck shifts to the PIOP messages. We further apply a two-fold solution: first, by adjusting the early-stop configuration, we compress the table with minor computation overhead; second, ring switching and dual-domain packing simultaneously compact sparse ciphertexts into fewer and smaller ciphertexts, while minimizing the noise and the number of automorphisms incurred. These optimizations additionally reduce the proof size by 2.16–$2.21\times$ over the hybrid method.

Scalability. We also discuss precisely the scalability of our scheme. Despite the $O(n \log n)$ asymptotic complexity, the prover time scales linearly in all realistic settings. The superlinear terms include rebuilding each round's tables from the committed ciphertexts and the PCS encoding. These together are estimated to account for less than 10% of the prover time up to $n = 2^{44}$. As for the proof size, as stated in Section 3.2, one can replace Ligero with polylogarithmic outer PCS, such as WHIR. With this change alone, we estimate that BOCCHI achieves 21–34 MB for $n = 2^{34}$. A recursive linear argument over the committed claims may lower the estimate to 10 MB and achieve a succinct proof; we leave its analysis to future work.

Verifier time. Our verifier is more efficient than Phalanx and RAA-Phalanx because of the fewer ciphertexts decrypted. At 2^{20}, verification takes only 0.25s, compared

TABLE 4. APPLYING THE CAC COMPILER TO OTHER SYSTEMS. PROVER TIME SHOWS THE RELATIVE CHANGE. SIZES ARE PRESENTED IN MB.

System	Original size	CAC size	Prover time
Laminate-fold†	45.2	20.6	-50%
PV Blind Fractal†	185.3	4.1	$+7\%$

to the 0.53s, 0.55s and 0.65s latency of Phalanx, RAA-Phalanx and Laminate-base. The bottleneck for the verifier lies in the evaluation of $\tilde{M}$, which requires $O(\mathrm{nnz})$ time and leads to a verifier slower than GKR-based methods at larger scales. Two techniques can be exploited: for SIMD-parallel workloads, the matrix can be decomposed, reducing the work by $N\times$; for other workloads, one may use holography. With either technique, the verifier is estimated to take less than 1s at 2^{34}.

6.3. Generalized Systems

Section 5 described the methods of applying our CAC compiler to these baseline methods. In this subsection, we briefly report the parameters and results to validate the effectiveness of the compiler, with Table 4 summarizing the results. For both systems, we estimate at $n = 2^{20}$.

Laminate. Current parameters leave 29.8 bits of noise tolerance. Combining (P1), (P2) in Section 4.3 and Laminate's bound $\log_2 W_\Sigma \le 16.8$ gives $S = [-16, 16] \cap \mathbb{Z}$ and $K = 26$. By compressing 120 test ciphertexts to the 26 filtered ciphertexts, we compress the overall proof size by $2.19\times$. Moreover, the latency of the outer SNARK for transcript packing predominates the prover time, within which the well-formed test accounts for 64% of the constraints. Therefore, our compiler not only compresses the proof, but also accelerates its prover time by $2.00\times$. Additionally, in this setting, the completeness error is reduced from the original 2^{-100} to 0.

PV blind Fractal. For $\hat{c}$, we sample multipliers βe_s with β uniform in the slot field $\mathbb{F}$, which give $p^{-2} \approx 2^{-128}$ per repetition. Its dense lift has $M_D < 2^{17.7}$, which still fits in the 48-bit PoD modulus. The claim carriers Y lack this headroom, so we use $S = [-2^8, 2^8] \cap \mathbb{Z}$ instead. We set $(K, K') = (2, 15)$ to achieve 128-bit soundness. Under these parameters, the public proof shrinks by $45\times$ with WHIR, while the commitment and opening only add 7% to the total server time. This modification through our compiler also amends the gap in the soundness proof stated in Section 4.1. A minor drawback is that the filter is incompatible with the original approximate-norm PoD of blind Fractal, so we replace it with an exact version. However, this incurs little computation overhead because as in [8], the bottleneck of PoD can be transferred to the server without additional encryption or proof.

7. Related Work

Verifiable computation over encrypted data. Early VCoED schemes compress ciphertexts with homomorphic hashing before proving, which restricts the supported functions or FHE parameters [1], [2], [49]. Later work proves the FHE evaluation itself, for example with SNARKs for ring arithmetic [3], [50], with lattice-based SNARKs [4], by arithmetizing TFHE bootstrapping in a general-purpose SNARK [5], or with VOLE-based proofs [51]. Viand et al. [52] systematize this design space. Such SNARK-over-FHE approaches must prove modular arithmetic over large rings together with rounding and key switching, so that even a single bootstrapping takes minutes to prove [5]. Blind SNARKs avoid this overhead by proving the plaintext relation instead. BOCCHI keeps this advantage but adopts the native view of SNARK-over-FHE where it is cheap: it commits to ciphertext residues directly and proves only linear relations over them, which involve neither rounding nor key switching.

HELIOPOLIS [6] showed that IOP-based proofs over homomorphically encrypted data are practical by evaluating FRI [53] under encryption, and Garg et al. [7] gave a general framework for proving statements about encrypted witnesses. Gama et al. [8] formalized blind hIOPs, showed that they remain sound under BCS compilation, instantiated blind Fractal over GBFV, and extended it to PV with a PoD. Phalanx [11] designed an FHE-friendly Spartan-based SNARK that exploits SIMD packing, and Huang et al. [10] replaced its Reed–Solomon code with generalized RAA codes for linear-time, constant-depth encoding. Laminate [9] and Lasagne [12] instead build on GKR, which avoids committing to intermediate values. These systems have made blind proving practical for millions of gates, but their proofs remain large: the Spartan-based systems open hundreds of ciphertext columns, blind Fractal opens the queried ciphertexts of every committed oracle, and the GKR-based systems send ciphertexts for every round, so that their proofs grow with circuit depth. Where ciphertexts are combined to save space, soundness has proven delicate. BOCCHI targets exactly this cost. It complements the GKR-based systems with a Spartan-based route for general FHE circuits, and the CAC compiler shrinks the proofs of Laminate and blind Fractal themselves.

A separate line of work avoids the heavy proof. VERITAS [54] embeds secret redundancy into plaintexts and checks it after decryption, but it does not support server-side private inputs. Moreover, Cheon and Jang [55] construct forgery attacks against both encodings that succeed whenever the underlying scheme is fully homomorphic. In comparison, blind SNARKs are general and provably sound.

Zero-knowledge proof delegation. Proof generation can also be delegated by secret-sharing the witness among several servers that run the prover jointly with MPC [56], [57], [58]. These approaches distribute trust across multiple servers, whereas blind SNARKs delegate to a single server and rely only on the security of FHE. Gama et al. provide an instantiation with their blind Fractal, but it requires the client to forward every queried ciphertext and its decryption to the auditor, along with a PoD. Compared to this method, our CAC compiler provides a superior choice.

8. Conclusion

We presented BOCCHI, a compact blind SNARK that commits to ciphertexts natively and opens them through a few filtered aggregates. It rests on two tools of independent interest: canonical binding, which tolerates malformed ciphertexts yet suffices for blind-SNARK soundness, and a lightweight noise filter that realizes it with optimal error. At 2^{22} constraints, BOCCHI outputs a 6.57 MB proof in 115.3 s, with up to $9.3\times$ smaller proofs and $2.54\times$ faster proving than Phalanx. The CAC compiler extends both tools to other PIOPs and HE schemes, offering a general way to open committed ciphertexts. Together, these results bring blind SNARKs closer to practice.

References

[1] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *ACM CCS*, 2014.

[2] A. Bois, I. Cascudo, D. Fiore, and D. Kim, "Flexible and efficient verifiable computation on encrypted data," in *PKC*, 2021.

[3] C. Ganesh, A. Nitulescu, and E. Soria-Vazquez, "Rinocchio: SNARKs for ring arithmetic," *J. Cryptol.*, vol. 36, no. 4, 2023.

[4] S. Atapoor, K. Baghery, H. V. L. Pereira, and J. Spiessens, "Verifiable FHE via lattice-based SNARKs," *IACR Commun. Cryptol.*, vol. 1, no. 1, 2024.

[5] L. T. Thibault and M. Walter, "Towards verifiable FHE in practice: Proving correct execution of TFHE's bootstrapping using plonky2," in *ACM CCS*, 2025.

[6] 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.

[7] S. Garg, A. Goel, and M. Wang, "How to prove statements obliviously?" in *CRYPTO*, 2024.

[8] 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.*, vol. 2, no. 3, 2025.

[9] K. Peshawaria, Z. Liu, B. Fisch, and E. Tromer, "Laminate: Succinct SIMD-friendly verifiable FHE," IACR ePrint 2025/2285.

[10] 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," IACR ePrint 2026/487.

[11] X. Zhang *et al.*, "Phalanx: An FHE-friendly SNARK for verifiable computation on encrypted data," in *ACM CCS*, 2025.

[12] X. Zhang, R. Wang, Q. Niu, P. Liu, X. Lu, L. Zhao, R. Hou, and Y. Deng, "Lasagne: Practical verifiable computation over encrypted data," IACR ePrint 2026/857.

[13] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," *J. ACM*, vol. 62, no. 4, 2015.

[14] Z. Zhou, Y. Li, Z. Yang, C. Hong, and T. Wei, "PRISM: Efficient zkSNARKs for RNS-based homomorphic encryption," in *ACM CCS*, 2026.

[15] M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *EUROCRYPT*, 1998.

[16] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight sublinear arguments without a trusted setup," in *ACM CCS*, 2017.

[17] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient homomorphic conversion between (ring) LWE ciphertexts," in *ACNS*, 2021.

[18] A. Gruen, "Some improvements for the PIOP for ZeroCheck," IACR ePrint 2024/108.

[19] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev, "WHIR: Reed–Solomon proximity testing with super-fast verification," in *EUROCRYPT*, 2025.

[20] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *J. ACM*, vol. 39, no. 4, 1992.

[21] S. Setty, "Spartan: Efficient and general-purpose zkSNARKs without trusted setup," in *CRYPTO*, 2020.

[22] E. Ben-Sasson, A. Chiesa, and N. Spooner, "Interactive oracle proofs," in *TCC*, 2016.

[23] O. Reingold, G. N. Rothblum, and R. D. Rothblum, "Constant-round interactive proofs for delegating computation," in *STOC*, 2016.

[24] A. Chiesa, D. Ojha, and N. Spooner, "Fractal: Post-quantum and transparent recursive proofs from holography," in *EUROCRYPT*, 2020.

[25] R. Canetti *et al.*, "Fiat-Shamir: From practice to theory," in *STOC*, 2019.

[26] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO*, 1986.

[27] A. Chiesa and E. Yogev, "Building cryptographic proofs from hash functions," https://snargsbook.org, 2024.

[28] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *CRYPTO*, 2012.

[29] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," IACR ePrint 2012/144.

[30] J.-C. Bajard, J. Eynard, M. A. Hasan, and V. Zucca, "A full RNS variant of FV like somewhat homomorphic encryption schemes," in *SAC*, 2016.

[31] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *ITCS*, 2012.

[32] R. Geelen and F. Vercauteren, "Fully homomorphic encryption for cyclotomic prime moduli," in *EUROCRYPT*, 2025.

[33] M. Chenal and Q. Tang, "On key recovery attacks against existing somewhat homomorphic encryption schemes," in *LATINCRYPT*, 2014.

[34] J. Kang, E. Pohle, and J. Spiessens, "Verifiable computation on encrypted data from homomorphic encryption over random encodings," IACR ePrint 2026/2422.

[35] C. Gentry, S. Halevi, C. Peikert, and N. P. Smart, "Field switching in BGV-style homomorphic encryption," IACR ePrint 2012/240.

[36] A. Chiesa, M. A. Forbes, and N. Spooner, "A zero knowledge sumcheck and its applications," IACR ePrint 2017/305.

[37] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song, "Libra: Succinct zero-knowledge proofs with optimal prover computation," in *CRYPTO*, 2019.

[38] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, "Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general," in *CRYPTO*, 2022.

[39] "Microsoft SEAL (release 4.1)," https://github.com/Microsoft/SEAL, 2023.

[40] F. Boemer, S. Kim, G. Seifu, F. D. M. de Souza, and V. Gopal, "Intel HEXL: Accelerating homomorphic encryption with Intel AVX512-IFMA52," in *WAHC*, 2021.

[41] M. Albrecht *et al.*, "Homomorphic encryption security standard," HomomorphicEncryption.org, Tech. Rep., 2018.

[42] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *USENIX Security*, 2018.

[43] Q. Pang, J. Zhu, H. Möllering, W. Zheng, and T. Schneider, "BOLT: Privacy-preserving, accurate and efficient inference for transformers," in *IEEE S&P*, 2024.

[44] Y. Yu, Y. Zhou, Y. Chen, P. Soto, W. Xiong, and M. Li, "Cachemir: Fully homomorphic encrypted inference of generative large language model with KV cache," in *IEEE S&P*, 2027.

[45] J. Yu *et al.*, "OptiPrime: Optimizing private inference through protocol-hardware co-design," in *IEEE/ACM MICRO*, 2026.

[46] S. Angel, H. Chen, K. Laine, and S. Setty, "PIR with compressed queries and amortized query processing," in *IEEE S&P*, 2018.

[47] S. J. Menon and D. J. Wu, "Spiral: Fast, high-rate single-server PIR via FHE composition," in *IEEE S&P*, 2022.

[48] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, 2020.

[49] D. Fiore, A. Nitulescu, and D. Pointcheval, "Boosting verifiable computation on encrypted data," in *PKC*, 2020.

[50] I. Cascudo, A. Costache, D. Cozzo, D. Fiore, A. Guimarães, and E. Soria-Vazquez, "Verifiable computation for approximate homomorphic encryption schemes," in *CRYPTO*, 2025.

[51] Z. Zhou *et al.*, "ZHE: Efficient zero-knowledge proofs for HE evaluations," in *IEEE S&P*, 2025.

[52] A. Viand, C. Knabenhans, and A. Hithnawi, "Verifiable fully homomorphic encryption," arXiv:2301.07041.

[53] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Fast Reed–Solomon interactive oracle proofs of proximity," in *ICALP*, 2018.

[54] S. Chatel, C. Knabenhans, A. Pyrgelis, C. Troncoso, and J.-P. Hubaux, "VERITAS: Plaintext encoders for practical verifiable homomorphic encryption," in *ACM CCS*, 2024.

[55] J. H. Cheon and D. Jang, "Cryptanalysis on lightweight verifiable homomorphic encryption," in *ASIACRYPT*, 2025.

[56] A. Ozdemir and D. Boneh, "Experimenting with collaborative zk-SNARKs: Zero-knowledge proofs for distributed secrets," in *USENIX Security*, 2022.

[57] S. Garg, A. Goel, A. Jain, G.-V. Policharla, and S. Sekar, "zkSaaS: Zero-knowledge SNARKs as a service," in *USENIX Security*, 2023.

[58] A. Chiesa, R. Lehmkuhl, P. Mishra, and Y. Zhang, "Eos: Efficient private delegation of zkSNARK provers," in *USENIX Security*, 2023.

Appendix

Appendix A. The Filter Theorem over Cyclotomic Rings

Setting. Let $\mathcal{K}$ be a cyclotomic field of degree N with ring of integers $R = \mathbb{Z}[\zeta]$, and identify R with $\mathbb{Z}^N$ and $\mathcal{K}_\mathbb{R} := \mathcal{K} \otimes_\mathbb{Q} \mathbb{R}$ with $\mathbb{R}^N$ through the power basis. Then $\lfloor \cdot \rceil$, $\|\cdot\|_\infty$, and $\|\cdot\|_\mathbb{Z}$ act on coefficients, $\lfloor y + r \rceil = \lfloor y \rceil + r$ for $r \in R$, and $r = 0$ whenever $\|r\|_\infty < 1$. For $a \in \mathcal{K}_\mathbb{R}$, the operator norm $\|a\|_{\mathrm{op}} := \sup_{y \neq 0} \|ay\|_\infty / \|y\|_\infty$ is subadditive, at least 1 on $R \setminus \{0\}$, and equal to $|a|$ for $a \in \mathbb{Z}$. The plaintext modulus $t \in R \setminus \{0\}$ generates a squarefree ideal $tR = \prod_{\mathfrak{p} \mid tR} \mathfrak{p}$, so $R_t := R/tR \cong \prod_\mathfrak{p} \mathbb{F}_\mathfrak{p}$ with $\mathbb{F}_\mathfrak{p} := R/\mathfrak{p}$; we write $a_\mathfrak{p}$ for slot $\mathfrak{p}$ of $a \in R_t$ and $\odot$ for the product of R_t, which acts slotwise. Reduction modulo tR extends to a ring homomorphism $\psi \mapsto \psi \bmod tR$ on $\mathcal{O}_t := \{\psi \in \mathcal{K} : \mathrm{ord}_\mathfrak{p}(\psi) \geq 0 \ \forall \mathfrak{p} \mid tR\}$, and ψ has a pole at $\mathfrak{p}$ if $\mathrm{ord}_\mathfrak{p}(\psi) < 0$.

For $c = (c_0, c_1) \in R_Q^2$ under key s, the scaled phase $x(c) = \frac{t}{Q}(c_0 + c_1 s) \in \mathcal{K}_\mathbb{R}$ is defined modulo tR; let $\mu(c) := \lfloor x(c) \rceil$ and call $\theta(c) := x(c) - \mu(c)$ its offset, so $\mathsf{Dec}(c) = \mu(c) \bmod tR$. Multipliers come from a set $S \subseteq R_t$ with a fixed injective lift $L : S \to R$, written $\bar{\beta} := L(\beta)$, which is the centered lift if $S \subseteq \mathbb{F}_t$. As in Section 4.3, let $\Delta(S) := \{\bar{\beta} - \bar{\beta}' : \beta \neq \beta' \in S\}$, $M_D := \max_{\delta \in \Delta(S)} \|\delta\|_{\mathrm{op}}$, and $M_S := \max_{\beta \in S} \|\bar{\beta}\|_{\mathrm{op}}$ in (P1) and (P2). Since $x(\bar{\beta} c) \equiv \bar{\beta}\, x(c) \pmod{tR}$, an aggregate $\hat{c} = \sum_{j \in [m]} \bar{\beta}_j c_j$ with offset sum $\sigma := \sum_j \bar{\beta}_j \theta(c_j)$ satisfies, as in Section 4.1, the *carry identity* $\theta(\hat{c}) = \sigma - \lfloor \sigma \rceil$ and $\mathsf{Dec}(\hat{c}) = \sum_j \beta_j \odot \mathsf{Dec}(c_j) + (\lfloor \sigma \rceil \bmod tR)$.

Lemma 7 (Rational decoding)**.** *Assume (P1) and let $\theta, \sigma' \in \mathcal{K}_\mathbb{R}$. (i) If $\mathcal{D}_\theta := \{\delta \in \Delta(S) : \|\delta\theta\|_\mathbb{Z} \leq 2\kappa\}$ is nonempty, there is a unique $\psi(\theta) \in \mathcal{K}$ with $\lfloor \delta\theta \rceil = \delta\, \psi(\theta)$ for all $\delta \in \mathcal{D}_\theta$. (ii) Let $\mathcal{A} := \{\beta \in S : \|\bar{\beta}\theta + \sigma'\|_\mathbb{Z} \leq \kappa\}$. For $\beta \neq \beta' \in \mathcal{A}$, $\delta := \bar{\beta} - \bar{\beta}'$ lies in $\mathcal{D}_\theta$ and $\lfloor \bar{\beta}\theta + \sigma' \rceil - \lfloor \bar{\beta}'\theta + \sigma' \rceil = \delta\, \psi(\theta)$. Consequently, for each $\mathfrak{p} \mid tR$, all $\beta \in \mathcal{A}$ agree in slot $\mathfrak{p}$ if $\psi(\theta)$ has a pole at $\mathfrak{p}$, and otherwise $\delta\, \psi(\theta) \equiv (\beta - \beta')_\mathfrak{p} \cdot (\psi(\theta) \bmod \mathfrak{p}) \pmod{\mathfrak{p}}$.*

Proof. (i) Let $n_\delta := \lfloor \delta\theta \rceil$ and $e_\delta := \delta\theta - n_\delta$. For $\delta, \delta' \in \mathcal{D}_\theta$, $\delta' n_\delta - \delta n_{\delta'} = \delta e_{\delta'} - \delta' e_\delta$ lies in R and has $\|\cdot\|_\infty \leq 4 M_D \kappa < 1$, so it vanishes and $\psi := n_\delta / \delta$ does not depend on δ. (ii) Write $\bar{\beta}\theta + \sigma' = n + e$ and $\bar{\beta}'\theta + \sigma' = n' + e'$ with $n, n' \in R$ and $\|e\|_\infty, \|e'\|_\infty \leq \kappa$. Then $\delta\theta = (n - n') + (e - e')$ with $\|e - e'\|_\infty \leq 2\kappa < \frac{1}{2}$ by (P1) and $M_D \geq 1$, so $\lfloor \delta\theta \rceil = n - n'$, $\delta \in \mathcal{D}_\theta$ ($\delta \neq 0$ as L is injective), and $n - n' = \delta\psi$ by (i). If $\mathrm{ord}_\mathfrak{p}(\psi) < 0$, then $\delta\psi \in R$ forces $\delta \in \mathfrak{p}$, and $\delta \equiv \beta - \beta' \pmod{tR}$ with $tR \subseteq \mathfrak{p}$ gives $\beta_\mathfrak{p} = \beta'_\mathfrak{p}$; otherwise reduce $\delta\psi$ modulo $\mathfrak{p}$. □

Definition 8 (Canonical value)**.** For a ciphertext c with $\theta := \theta(c)$, let $\mathsf{Can}(c) := \mu(c) + \psi(\theta) \bmod tR$ if $\mathcal{D}_\theta \neq \emptyset$ and $\psi(\theta) \in \mathcal{O}_t$, and $\mathsf{Can}(c) := \mathsf{Dec}(c)$ otherwise.

For $S \subseteq \mathbb{F}_t$, this is Definition 3, as $\mathcal{D}_{\theta(c)} = \mathcal{D}_c$ and $|\delta| < t$ rules out poles. $\mathsf{Can}(c)$ depends only on c, the key, and (S, L, κ), and equals $\mathsf{Dec}(c)$ for well-formed c, since (P2) gives $\|\delta\theta(c)\|_\infty \leq 2 M_S \varepsilon \leq 2\kappa < \frac{1}{2}$ and hence $\psi = 0$.

Theorem 9 (Filter theorem over rings)**.** *Assume (P1). Fix ciphertexts $c_1, \ldots, c_m$, claims $y_1, \ldots, y_m \in R_t$, and an additive map $\Lambda : R_t \to \mathcal{Y}$. One check draws $\beta_1, \ldots, \beta_m \leftarrow S$ and accepts iff $\|x(\hat{c})\|_\mathbb{Z} \leq \kappa$ and $\Lambda\big(\mathsf{Dec}(\hat{c}) - \sum_j \beta_j \odot y_j\big) = 0$. For any index j^*, with $\Xi := \mathsf{Can}(c_{j^*}) - y_{j^*}$, the check accepts with probability at most $\max\{\epsilon_{\mathrm{slot}}(S), \epsilon_\Lambda(S, \Xi)\}$, where $\epsilon_{\mathrm{slot}}(S) := \max_{\mathfrak{p}, a} \Pr_{\beta \leftarrow S}[\beta_\mathfrak{p} = a]$ and $\epsilon_\Lambda(S, \Xi) := \max_{\beta_0 \in S} \Pr_{\beta \leftarrow S}[\Lambda((\beta - \beta_0) \odot \Xi) = 0]$.*

Proof. Fix all multipliers except $\beta := \beta_{j^*}$, and let $\theta := \theta(c_{j^*})$ and $\sigma' := \sum_{j \neq j^*} \bar{\beta}_j \theta(c_j)$; by the carry identity, the filter passes iff $\beta \in \mathcal{A}$ as in Lemma 7. If $|\mathcal{A}| \leq 1$, the check accepts with probability at most $\max_{\beta_0} \Pr[\beta = \beta_0] \leq \epsilon_{\mathrm{slot}}$. Otherwise $\psi := \psi(\theta)$ exists, and if it has a pole at $\mathfrak{p}$, then $\mathcal{A}$ lies in one fiber $\{\beta : \beta_\mathfrak{p} = a\}$, so again the bound is ϵ_{slot}. In the remaining case $\psi \in \mathcal{O}_t$ and $\mathsf{Can}(c_{j^*}) = \mu(c_{j^*}) + \psi \bmod tR$, so the carry identity and Lemma 7 give $\mathsf{Dec}(\hat{c}(\beta)) - \mathsf{Dec}(\hat{c}(\beta')) = (\beta - \beta') \odot \mathsf{Can}(c_{j^*})$ for $\beta, \beta' \in \mathcal{A}$. The comparisons at β and β' thus differ by $\Lambda((\beta - \beta') \odot \Xi)$, so if some $\beta_0 \in \mathcal{A}$ is accepted, every accepted

β satisfies $\Lambda((\beta-\beta_0)\odot\Xi)=0$, which has probability at most ϵ_Λ. Averaging over the other multipliers completes the proof. □

Corollary 10 (Instances)**.** *Assume (P1). One check accepts with probability at most (I)* $1/\min_{\mathfrak{p}}|\mathbb{F}_{\mathfrak{p}}|$ *for dense multipliers* $S=R_t$*,* $\Lambda=1$*, and* $y_{j^*}\neq\mathsf{Can}(c_{j^*})$*, i.e.,* $1/t$ *for fully split BFV and* $p^{-2}\approx 2^{-128}$ *for the* $\mathbb{F}_{p^2}$ *slots of GBFV with a 64-bit* p*; and (II)* $1/|S|$ *for scalar multipliers* $S\subseteq\mathbb{F}_t$ *with* t *prime, an* $\mathbb{F}_t$*-linear* Λ*, and claims* $y_j\in\mathcal{Y}$ *compared as in* BatchCheck *with* $y_{j^*}\neq\Lambda(\mathsf{Can}(c_{j^*}))$*, which gives Theorem 4 over* K *independent repetitions.*

Proof. (I) Uniform $\beta\in\prod_{\mathfrak{p}}\mathbb{F}_{\mathfrak{p}}$ gives $\epsilon_{\text{slot}}=1/\min_{\mathfrak{p}}|\mathbb{F}_{\mathfrak{p}}|$, and if $\Xi_{\mathfrak{p}}\neq 0$, then $(\beta-\beta_0)\odot\Xi=0$ forces $\beta_{\mathfrak{p}}=(\beta_0)_{\mathfrak{p}}$. (II) Every slot of $\beta\in\mathbb{F}_t$ equals β, so $\epsilon_{\text{slot}}=1/|S|$; by $\mathbb{F}_t$-linearity, the proof of Theorem 9 applies with $\Lambda((\beta-\beta_0)\odot\Xi)$ replaced by $(\beta-\beta_0)(\Lambda(\mathsf{Can}(c_{j^*}))-y_{j^*})$, which vanishes only at $\beta=\beta_0$. □

Single-slot variant. The blind Fractal instance holds claims only for queried slots. For a slot $\mathfrak{p}$ with indicator $e_{\mathfrak{p}}$, let $S_{\mathfrak{p}}:=\{\beta e_{\mathfrak{p}}:\beta\in\mathbb{F}_{\mathfrak{p}}\}$ with a fixed injective lift L and $M_{\mathfrak{p}}:=\max_{\delta\in\Delta(S_{\mathfrak{p}})}\|\delta\|_{\text{op}}$. Fix ciphertexts c_j and distinct positions $v=(j_v,\mathfrak{p}_v)$ with claims $y_v\in\mathbb{F}_{\mathfrak{p}_v}$. The check draws independent uniform $\beta_v\in\mathbb{F}_{\mathfrak{p}_v}$, forms $\hat{c}:=\sum_v L(\beta_v e_{\mathfrak{p}_v})c_{j_v}$, and accepts iff $\|x(\hat{c})\|_{\mathbb{Z}}\leq\kappa$ and $\mathsf{Dec}(\hat{c})_{\mathfrak{p}}=\sum_{v:\mathfrak{p}_v=\mathfrak{p}}\beta_v y_v$ in every slot $\mathfrak{p}$. With $\theta:=\theta(c_{j_v})$ and $\psi:=\psi(\theta)$ from Lemma 7 for $S_{\mathfrak{p}_v}$, let $\mathsf{Can}_v:=\mu(c_{j_v})_{\mathfrak{p}_v}+(\psi \bmod \mathfrak{p}_v)$ if $\mathcal{D}_\theta\neq\emptyset$ and $\text{ord}_{\mathfrak{p}_v}(\psi)\geq 0$, and $\mathsf{Can}_v:=\mathsf{Dec}(c_{j_v})_{\mathfrak{p}_v}$ otherwise. Since each position is lifted separately, Can_v depends only on c_{j_v}, $\mathfrak{p}_v$, the key, and public parameters, and it equals $\mathsf{Dec}(c_{j_v})_{\mathfrak{p}_v}$ for well-formed ciphertexts.

Theorem 11 (Single-slot filter)**.** *If* $4M_{\mathfrak{p}}\kappa<1$ *for all* $\mathfrak{p}$ *and* $y_{v^*}\neq\mathsf{Can}_{v^*}$ *for some* v^**, the check accepts with probability at most* $1/|\mathbb{F}_{\mathfrak{p}_{v^*}}|$*; independent repetitions multiply.*

Proof. Let $(j,\mathfrak{p}):=v^*$, fix all β_v with $v\neq v^*$, and let $\beta:=\beta_{v^*}$ and $\theta:=\theta(c_j)$. The other positions on c_j have fixed multipliers, so the offset sum is $L(\beta e_{\mathfrak{p}})\theta+\sigma'$ for a fixed σ', and the filter passes iff $\beta\in\mathcal{A}$ as in Lemma 7 for $S_{\mathfrak{p}}$. If $|\mathcal{A}|\geq 2$, ψ exists; a pole at $\mathfrak{p}$ would make all $\beta\in\mathcal{A}$ agree in slot $\mathfrak{p}$, i.e., be equal, so $\text{ord}_{\mathfrak{p}}(\psi)\geq 0$, and the carry identity with Lemma 7 gives $\mathsf{Dec}(\hat{c}(\beta))_{\mathfrak{p}}-\mathsf{Dec}(\hat{c}(\beta'))_{\mathfrak{p}}=(\beta-\beta')\mathsf{Can}_{v^*}$. The slot-$\mathfrak{p}$ comparisons at β and β' thus differ by $(\beta-\beta')(\mathsf{Can}_{v^*}-y_{v^*})\neq 0$, so at most one $\beta\in\mathcal{A}$ is accepted. Averaging over the other multipliers proves the claim. □

Completeness holds if $\sum_v\|L(\beta_v e_{\mathfrak{p}_v})\|_{\text{op}}\,\varepsilon\leq\kappa$ for all multipliers.

Reduction from BGV. In BGV, $\mathsf{Dec}(c)=\chi(c) \bmod t$ for the phase $\chi(c):=[c_0+c_1 s]_Q$ with centered representatives; let $\theta(c):=\chi(c)/Q\in[-\frac{1}{2},\frac{1}{2}]^N$ be the relative phase.

Lemma 12 (BGV reduction)**.** *Let* $\gcd(Q,t)=1$*,* $\hat{c}=\sum_j\bar{\beta}_j c_j$*, and claims* $y_j\in R_t$*. With* $\tilde{y}_j:=-Q^{-1}(y_j-\mathsf{Dec}(c_j)) \bmod t$ *and* $\sigma:=\sum_j\bar{\beta}_j\theta(c_j)$*, we have* $\mathsf{Dec}(\hat{c})=\sum_j\beta_j\odot y_j$ *iff* $\lfloor\sigma\rceil\equiv\sum_j\beta_j\odot\tilde{y}_j \pmod t$*, and* $\|\chi(\hat{c})\|_\infty\leq\kappa Q$ *iff* $\|\sigma\|_{\mathbb{Z}}\leq\kappa$*.*

Proof. As $\sigma=\sum_j\bar{\beta}_j\chi(c_j)/Q$, we have $\chi(\hat{c})=[\sum_j\bar{\beta}_j\chi(c_j)]_Q=Q(\sigma-\lfloor\sigma\rceil)$. Hence $\mathsf{Dec}(\hat{c})\equiv\sum_j\beta_j\odot\mathsf{Dec}(c_j)-Q\lfloor\sigma\rceil \pmod t$; substitute into the comparison and multiply by $-Q^{-1}$. The second claim follows from $\|\chi(\hat{c})\|_\infty=Q\|\sigma\|_{\mathbb{Z}}$. □

A BGV check is thus a check as above with $\mu(c_j)=0$, offsets $\theta(c_j)$, claims $\tilde{y}_j$ obtained from y_j by a bijection fixed with c_j, and the filter $\|\chi(\hat{c})\|_\infty\leq\kappa Q$, so all results above apply with canonical value $\mathsf{Dec}(c_j)-Q(\psi(\theta(c_j)) \bmod tR)$; for an $\mathbb{F}_t$-linear Λ, the equivalence holds after applying Λ, as Q is invertible modulo t.

Appendix B. Unfiltered Checks

This appendix quantifies Section 4.1 with explicit attacks on the aggregated check without the filter. Let t be prime and $S=[-b,b]\cap\mathbb{Z}$ with $2b+1\leq t$. As in the proof of Theorem 4, fix all multipliers except $\beta:=\beta_{j^*}$. By the carry identity, an unfiltered check with full comparison then accepts on one coefficient iff $\lfloor\bar{\beta}\theta+\sigma'\rceil\equiv\beta y+r \pmod t$, where θ is the offset of c_{j^*} in that coefficient, y the claim minus the decryption there, and σ', r collect the other ciphertexts.

Example 13 (Two claims at about $b^{-1/2}$)**.** Let $\sigma'=r=0$ and let θ lie between adjacent fractions $p/q<p'/q'$ with $q,q'\approx\sqrt{b}$. On each multiple $\bar{\beta}\in S$ of q with $|\bar{\beta}(\theta-p/q)|<\frac{1}{2}$, $\lfloor\bar{\beta}\theta\rceil=\bar{\beta}p/q\equiv\beta\cdot pq^{-1}$, so $y=pq^{-1}$ passes; as $1/(qq')\approx 1/b$, this covers nearly all multiples of q, and symmetrically for $y=p'q'^{-1}$. As a numeric example, for $t=786433$, $S=\mathbb{F}_t$, and $\theta=0.0715408323$ between $46/643$ and $45/629$, the claims 320444 and 51262 pass on 1223 and 1225 of the t multipliers, about $1.38/\sqrt{t}\approx 2^{-9.3}$ each, so $K=8$ repetitions give only about 2^{-75}.

As one of the two claims is wrong under any canonical map, the unfiltered check binds with error no better than about $b^{-1/2}$. By the tradeoff of Section 4.3, the headroom for range b' with the filter allows $b=4b'^2$ without it, where the attack's $b^{-1/2}=1/(2b')$ already matches the filtered error $1/(2b'+1)$. Therefore, even if no stronger attack existed, the unfiltered check would thus be no better than the filtered one, which alone comes with a proof.

Example 14 (Cancellation under compression)**.** Let a ciphertext have offsets θ_1 and $\theta_1-\frac{1}{2}$ in two coefficients and 0 elsewhere, and let Λ subtract the two. For even $\bar{\beta}$, the carries differ by $\lfloor\bar{\beta}\theta_1\rceil-\lfloor\bar{\beta}\theta_1-\bar{\beta}/2\rceil=\bar{\beta}/2\equiv 2^{-1}\beta$, so the claim $y=2^{-1}$ passes with probability about $\frac{1}{2}$.

The passing value is a property of the pair and of Λ. In BOCCHI, Λ_u is fixed by the evaluation query only after the commitment, so an unfiltered compressed check binds no value at commitment time, whereas Theorem 4 holds for every $\mathbb{F}_t$-linear Λ.

Appendix C. Deviation from Dec in Batched PoDs

This appendix proves the claim of Section 4.1 on the batched PoD of Gama et al. [8] in the setting of Appendix A, with multipliers $\alpha \in S \subseteq R_t$ acting as $\alpha \cdot c := \bar{\alpha} c$.

Proposition 15. *Let p be a rational prime and $\mathfrak{P} \mid pR$ a prime with $pR + tR = R$ and $e_{\mathfrak{P}} := \mathrm{ord}_{\mathfrak{P}}(pR)$. Choose $w \in R$ with $\mathrm{ord}_{\mathfrak{P}}(w) = e_{\mathfrak{P}} - 1$ and $\mathrm{ord}_{\mathfrak{Q}}(w) \geq \mathrm{ord}_{\mathfrak{Q}}(pR)$ for the other primes $\mathfrak{Q} \mid pR$, and let $z := w/p$. (i) For $r \in R$, $rz \in R$ iff $r \in \mathfrak{P}$. (ii) Let $x(c) = a + z + \eta$ with $a \in R$ and $\|\bar{\alpha}\eta\|_\infty < \frac{1}{2}$ for all $\alpha \in S$, and let $m' := a + wp^{-1} \bmod tR$. Whenever $\bar{\alpha} \in \mathfrak{P}$, $\theta(\alpha \cdot c) = \bar{\alpha}\eta$ and $\mathsf{Dec}(\alpha \cdot c) = \alpha \odot m'$. (iii) Hence any check that only bounds the noise of the combination and compares it with the claims accepts the claim m' with probability at least $\Pr_{\alpha \leftarrow S}[\bar{\alpha} \in \mathfrak{P}]$, although in general $m' \neq \mathsf{Dec}(c) = a + \lfloor z + \eta \rceil \bmod tR$.*

Proof. (i) At primes not above p, $\mathrm{ord}(z) = \mathrm{ord}(w) \geq 0$; at $\mathfrak{Q} \neq \mathfrak{P}$ above p, $\mathrm{ord}_{\mathfrak{Q}}(z) \geq 0$; and $\mathrm{ord}_{\mathfrak{P}}(rz) = \mathrm{ord}_{\mathfrak{P}}(r) - 1$. An element of $\mathcal{K}$ with all valuations nonnegative lies in R. (ii) If $\bar{\alpha} \in \mathfrak{P}$, then $\bar{\alpha}a + \bar{\alpha}z \in R$ and $\|\bar{\alpha}\eta\|_\infty < \frac{1}{2}$, so this is the rounding of $x(\alpha \cdot c)$. Since $p \cdot \bar{\alpha}z = \bar{\alpha}w$ and p is invertible modulo tR, $\bar{\alpha}z \equiv \bar{\alpha}wp^{-1} \pmod{tR}$. (iii) follows; in a batch, the other ciphertexts are honest and contribute no carry. □

Example 16 (GBFV parameters of Gama et al.)**.** For the $(21 \cdot 2^j)$-th cyclotomic ring and $p = 2$, we have $e_{\mathfrak{P}} = 2^{j-1}$, residue degree 6, and two primes above 2, so $|R/\mathfrak{P}| = 2^6$. For $t = X^k - b$ with b a power of two, $t \equiv X^k$ is a unit modulo $2R$, so $2R + tR = R$. If lifts are equidistributed modulo $\mathfrak{P}$, the acceptance probability would be about 2^{-6} per batch.

In fact, the accepted value is canonical. Let $\|\bar{\alpha}\eta\|_\infty \leq \kappa$ for all $\alpha \in S$, as for honest noise, with κ satisfying (P1), and let $\delta \in \Delta(S) \cap \mathfrak{P}$. Then $\delta z \in R$ and $\|\delta\eta\|_\infty \leq 2\kappa < \frac{1}{2}$, so $\lfloor \delta\theta(c) \rceil = \delta(z - \lfloor z + \eta \rceil)$ and $\delta \in \mathcal{D}_{\theta(c)}$. Lemma 7 thus gives $\psi(\theta(c)) = z - \lfloor z + \eta \rceil$, which lies in $\mathcal{O}_t$ because $pR + tR = R$, and hence $\mathsf{Can}(c) = a + z \bmod tR = m'$.

Appendix D. Proof of the Composition Theorem

Round-by-round soundness. Let Π be a public-coin hIOP for $\mathcal{R}$ in which each round consists of a prover oracle f_i and a uniform verifier challenge r_i, followed by non-adaptive queries and a decision. Π is round-by-round sound with error ϵ_Π [25] if there is a set $\mathcal{D}_\Pi$ of doomed partial transcripts such that (1) $(x, \emptyset) \in \mathcal{D}_\Pi$ for every false statement x; (2) for every $(x, \mathsf{tr}) \in \mathcal{D}_\Pi$ ending before a prover message and every next message f, $\Pr_r[(x, \mathsf{tr}, f, r) \notin \mathcal{D}_\Pi] \leq \epsilon_\Pi$; and (3) the verifier rejects every complete transcript in $\mathcal{D}_\Pi$. The knowledge version adds an extractor that outputs a witness from any (x, tr, f) whose escape probability exceeds ϵ_Π.

Decoding maps. A decoding map assigns to each ciphertext of $\mathcal{E}[\Pi]$ a plaintext as a deterministic function of the ciphertext, its position in the protocol, the key, and the public parameters, and is applied to transcripts entrywise. Both Dec and Can_T, which applies Can to the entries of T and Dec to all other ciphertexts, are decoding maps.

Proof of Theorem 5. Call a blind partial transcript (c_x, tr) doomed iff $(\mathsf{Can}_T(c_x), \mathsf{Can}_T(\mathsf{tr})) \in \mathcal{D}_\Pi$. If the decoded statement $\mathsf{Can}_T(c_x)$ is false, the empty transcript is doomed. From a doomed state, the image $f := \mathsf{Can}_T(c_i)$ of any next message c_i is fixed before r_i is drawn, because Can_T is a decoding map, so the escape probability equals that of Π on f and is at most ϵ_Π; for knowledge, the extractor of Π runs on Can_T-images with oracle access to Can_T. The claims and all K multiplier sets form one extra round after the queries, and a complete transcript is doomed iff the verifier rejects. Let the state after the queries be doomed and y the claims. If every claim equals $\Lambda(\mathsf{Can}(\cdot))$ of its entry, the verifier of Π sees a complete transcript in $\mathcal{D}_\Pi$ and rejects whatever the multipliers. Otherwise some claim is wrong, and since the claims and Λ precede the multipliers, $\mathsf{BatchCheck}$ accepts with probability at most ϵ_{bind}. Hence $\mathcal{E}[\Pi]_{\mathsf{BC}}$ is round-by-round (knowledge) sound with error $\max(\epsilon_\Pi, \epsilon_{\mathrm{bind}})$. Since ϵ_{bind} bounds probabilities over the multipliers for fixed ciphertexts and claims, this holds even against a prover that knows the key. □

Appendix E. Proof of the Compiler Theorem

Consider a prover P^* making at most Q_H random-oracle queries; ϵ_Π, ϵ_{bind}, and ϵ_{LA} bound a single attempt.

Hybrid 1 aborts whenever the verifier accepts an LA proof for an aggregate $\hat{c}_k$ that is not the prescribed combination of the committed ciphertexts. Extractable binding fixes these ciphertexts at commitment time, and the knowledge soundness of LA bounds the abort probability by $Q_H \epsilon_{\mathsf{LA}}$.

Hybrid 2. Every accepted aggregate is now the prescribed combination of ciphertexts fixed before the queries, and the claims are hashed before the multipliers: the carriers in DV, and in PV the published claims before the multipliers of the check that binds them. The protocol is thus the BCS compilation [22] of $\mathcal{E}[\Pi]_{\mathsf{BC}}$ with LA in place of Merkle trees, which the BCS analysis permits as it uses only binding, extractable commitments and the random oracle. Several $\mathsf{BatchChecks}$, e.g., one binding carriers to committed entries and one binding published claims to carriers, add one round each, escaping a doomed state with at most their own binding errors, whose sum is ϵ_{bind}. By Theorem 5, $\mathcal{E}[\Pi]_{\mathsf{BC}}$ is round-by-round (knowledge) sound with error $\max(\epsilon_\Pi, \epsilon_{\mathrm{bind}})$, so P^* succeeds with probability at most $Q_H \max(\epsilon_\Pi, \epsilon_{\mathrm{bind}}) + O(Q_H^2 2^{-\lambda})$ [27], and adding $Q_H \epsilon_{\mathsf{LA}}$ proves the DV bound.

Public verifiability. In PV, the client publishes the claims and a PoD, under a key fixed by a commitment in the setup, that every filtered aggregate passes its comparison and has noise at most κ'. Except with probability ϵ_{PoD}, this yields a key s^* under which this holds. As $4M_D\kappa' < 1$, the filter theorems hold under s^* with threshold κ', and Can under s^*

is fixed before the challenges because the key commitment is, so the DV argument applies and adds ϵ_{PoD}.