

Encore: Aggregatable FHE-over-SNARK with Constant Proof Size

Katerina Sotiraki
Yale University

Yunhao Wang
Yale University, IC3

Fan Zhang
Yale University, IC3

Abstract—Verifiable fully homomorphic encryption (FHE) enables a client to verify the correctness of homomorphic computation by an untrusted server. A promising paradigm, *FHE-over-SNARK*, proves correctness at the FHE *plaintext* level and has proofs scaling at least logarithmically with circuit size. The alternative, proving at the *ciphertext* level (i.e., *SNARK-over-FHE*), forces non-arithmetic FHE maintenance operations into the proven relation and inflates circuit size by orders of magnitude, yet offers constant proof size. Proving FHE computation with the classical LPCP approach introduced by Bitansky et al., later enhanced in LUNA, follows this paradigm.

We revisit the LPCP framework and lift it into the *FHE-over-SNARK* paradigm, obtaining proofs of *constant* size, independent of the circuit size. Our key technique, an *encoding-ciphertext multiplication*, allows us to prove arithmetic relations on the plaintext level by preserving the homomorphism of the underlying messages of both the encodings and the ciphertexts. By leveraging this technique, our protocol leaves the costly maintenance operations entirely outside the proof relation. The resulting protocol, Encore, also natively supports SIMD operations and is agnostic to the circuit structure. The security of Encore is based on a multi-key variant of strict linear targeted malleability (LTM), which we show to be equivalent to the standard LTM assumption of prior LPCP-based SNARKs. We further present a distributed prover framework, d-Encore, so that the proof generation is distributed for efficiency to multiple sub-provers, while the verifier checks one succinct proof.

On a 2^{20} -gate circuit mixing multiplication and addition, with SIMD ring dimension 2^{14} , Encore produces a 4.64 MB proof in 9.38 single-core hours and verifies it in under 10 ms. This improves on Phalanx and Blind ZK by three to five orders of magnitude in prover time, proof size and verification time; on RAA by five orders of magnitude in proof size and $2.09\times$ in prover time; on Laminate by $2.87\times$ in proof size and $1.19\times$ in prover time; and on Lasagne by $30.7\times$ and $2.55\times$, respectively.

1. Introduction

Outsourcing computation to untrusted servers raises two orthogonal concerns: *privacy* of the data and *integrity* of the result. Fully homomorphic encryption (FHE) addresses the first, letting a server compute directly on encrypted data, with applications spanning private information retrieval [22], [31], [32], oblivious message retrieval [23], [38], [44], [45], [46], [47], privacy-preserving machine learning [36], [37], and single secret leader election [8], [41], [62]. After a decade of progress [10], [15], [16], [18], [19], FHE is now practical

for many such tasks, and schemes like BFV/BGV [10], [11], [12], [19] enable SIMD operations to improve the throughput of homomorphic evaluation. Integrity, however, remains largely unaddressed: A malicious server can cheat in the homomorphic evaluation and return a ciphertext with an incorrect plaintext. A client holding only ciphertexts has no way to detect it. *Verifiable FHE* (VFHE) [59] or Verifiable Computation over Encrypted Data (VCoED) [27] closes this gap by additionally requiring the server to prove its honest evaluation. The natural tool for such a proof is a succinct non-interactive argument of knowledge (SNARK), which offers sublinear proofs and fast verification. Yet composing SNARKs with FHE is far from straightforward, and this has given rise to two paradigms.

A promising line, *FHE-over-SNARK*, evaluates the SNARK statement inside FHE schemes and thus transforms the proof relation to the *plaintext* level. Its structural shift makes all non-arithmetic maintenance operations fall *outside* the proven relation entirely. Works such as Blind ZK [24], Phalanx [66] and RAA [33] fall under this paradigm, and are built on a polynomial commitment-based framework, where the SIMD property is used to pack the commitment. However, as observed in [53], the SIMD slot capacity is fully consumed by polynomial commitments, forcing the payload to run without parallelism. RAA makes the commitment itself linear-time by replacing the Reed-Solomon code with a non-binary RAA code, but likewise fills the slots with the wires of a single instance and yields a proof scaling with the data slot capacity. Indeed, it is unclear how to enhance these three protocols to support SIMD FHE payload efficiently. Recent works like Laminate [53] and Lasagne [67] support native SIMD packing on FHE data and further improve the prover-side performance by building on the GKR framework, while still having non-constant proof size.

An earlier line follows the paradigm of *SNARK-over-FHE*, which tries to directly parse the homomorphic evaluation as a language and use existing SNARK frameworks to prove the statement. The adaptation is conceptually direct while the major drawback is that the proven relation is performed on *ciphertexts*. This makes the performance under this paradigm really limited, since the practical FHE schemes rely on non-arithmetic *maintenance* operations, such as relinearization, key switching, modulus switching, and bootstrapping, that must either be simulated inside the proven relation or proven separately, inflating circuit size by orders of magnitude.

Among SNARK-over-FHE, there is a line following linear probabilistically checkable proof (LPCP) [9], [35], [56] over a quadratic arithmetic program (QAP) [6], [25], [28], [52]

that offers a property the GKR line structurally cannot: *constant* proof size and verification independent of the witness size. These constructions, however, were largely set aside as impractical for verifiable FHE since, when applied naturally, an LPCP proves homomorphic evaluation at the *ciphertext* level and incurs significant overhead due to non-arithmetic maintenance operations [59]. Moreover, to support ring arithmetic, either the encoding scheme is stretched to embed the entire ciphertext space, which increases proof size [25], or the large ring operations are reduced to field operations, which increases the circuit (see § 1.2). For these reasons, the LPCP line was deemed a poor fit, and recent work moved toward GKR [53], [67] or linear-time codes [33]. Thus, we pose the following question: *Can we build an FHE-over-SNARK that has constant proof size and verification, and also natively supports SIMD FHE operations?*

We answer this question affirmatively, showing that the LPCP/QAP framework can be adapted into an FHE-over-SNARK with a circuit-dependent setup, constant proof size, and witness-independent verification. Crucially, the payload’s SIMD packing is left untouched during proof generation: each wire is a single packed BGV ciphertext occupying all D slots, so the relation is proven slot-wise without rearranging the data. The design rests on two deliberate considerations. First, the choice between the polynomial-commitment-plus-IOP approach [3], [24], [33], [66] and LPCP trades public verifiability for a designated verifier to get rid of the polynomial commitment. Under verifiable FHE, this is the natural default¹, so we adopt it. Second, against the GKR/sumcheck approach [33], [53], [67], the LPCP framework yields a constant-size proof independent of circuit depth and the number of queries.

We further extend our protocol to a distributed-prover setting to meet the throughput requirements of real deployments. Distributed SNARKs [39], [43], [54], [63], [64], [65] have long been studied, while all of them work over finite fields and rely on field-specific machinery that does not transfer to ring arithmetic, leaving distributed proving for the ring-native FHE setting open. We close this gap with an aggregation framework on top of our FHE-over-SNARK protocol that composes sub-proofs into a succinct proof to preserve the constant proof size while dramatically reducing the runtime for proof generation for large-scale circuits. A formal comparison appears in Table 1.

1.1. Our Contributions

We introduce Encore, an FHE-over-SNARK scheme with constant proof size based on the LPCP framework. We also propose d-Encore, a distributed scheme to support proving large-scale circuits. Our contributions are summarized below.

- **LPCP-based FHE-over-SNARK protocol:** We revisit the LPCP framework and devise a technique for *encoding-ciphertext multiplication* between (F)HE ciphertexts and

1. The client already holds the secret key. Public verifiability can be recovered, as in prior work [3], [24], [66], by having the verifier additionally prove in zero-knowledge that it generated the encodings honestly and accepted the proof.

LPCP query encodings; this allows the QAP identity check to be done at the (F)HE *plaintext* level. This leaves non-arithmetic FHE maintenance operations (e.g., relinearization, key switching, and bootstrapping) out of the proven relation, eliminating the circuit blow-up that rendered LPCP impractical for verifiable FHE. The resulting Encore has several advantages: the prover runtime and proof size are independent of the circuit’s multiplicative depth, it natively supports SIMD, and, unlike the prior works [53], [67], proving addition gates is free.

- **Soundness with no new assumption:** The soundness depends on a multi-key variant of the strictly linear targeted malleability (LTM) assumption. We show that it reduces to the *standard* strictly LTM assumption of prior works [9], [56] (Lemma 4.2). Thus, our construction rests on the same security foundation as prior LPCP-based SNARKs.
- **Aggregation framework for distributed prover:** We further extend Encore to the distributed prover setting and present d-Encore. The key technique is an efficient aggregation scheme that combines sub-proofs by distributed sub-provers into a succinct final proof. We introduce the novel use of *entangled* encodings in crs to ensure that sub-proofs are bound to the original witnesses.
- **Evaluation:** We compare the performance of Encore with other FHE-over-SNARK protocols (§ 6). Under representative parameters, for a circuit of size 2^{20} , with half multiplication and half addition (i.e., 2^{19} multiplication gates) and SIMD operating on plaintext messages with ring dimension 2^{14} , Encore outperforms recent FHE-over-SNARK systems Phalanx [66] and Blind ZK [24] by three to five orders of magnitude across all metrics. Compared to [33], which also does not preserve the SIMD packing of the FHE data, Encore yields five orders of magnitude smaller proofs with a $2.09\times$ faster prover. Compared to one of the latest works Lasagne [67], Encore yields $\sim 30.71\times$ smaller proofs with a $2.55\times$ faster prover. And compared to the strongest baseline Laminate [53], our protocol Encore has a $2.87\times$ smaller proof with a $1.19\times$ faster prover, and is agnostic to the circuit topology. For a circuit with up to 2^{26} multiplication gates with SIMD dimension 2^{15} , d-Encore could generate a proof in ~ 32 hours with 2^9 sub-provers.

1.2. Related Works

SNARK-over-FHE. Using state-of-the-art SNARKs, proving correctness happens at the *ciphertext* level, which forces ring operations to be simulated by field operations and incurs two compounding blow-ups: 1) each ring multiplication corresponds to $O(D \log D)$ field constraints via the NTT, where D is the ring dimension, and 2) the subsequent reduction modulo $\mathcal{R}_q$ adds $\Theta(D \log q)$ range-check constraints. A line of *ring-native* proof systems avoids the first blow-up by operating over rings directly. LaBRADOR [5] and its extensions Greyhound [51] and DV-zkSNARK [1] give Module-SIS RICS proofs, but require a short witness, which is restrictive when proving honest evaluation of a general FHE circuit.

Closer to verifiable FHE, a series of works prove ciphertext-level $\mathcal{R}_q$ computation directly: Atapoor et al. [4] exploit the double-CRT/RNS representation to prove arithmetic together with the non-arithmetic maintenance operations (key and modulus switching), while Approx-VC [13] builds $\mathcal{R}_q$ -native PIOPs and polynomial commitments handling arithmetic, range checks, and rescaling. These achieve sublinear verification but must explicitly prove the heavy maintenance operations inside the relation. ZHE [68] targets HE evaluation via VOLE-based proofs. Moreover, all the aforementioned protocols suffer from a large proof scaling at least logarithmically with the circuit size. A separate subline [42], [57] focuses specifically on honest bootstrapping. Among constant-proof-size schemes, ISW [35], LUNA [56], and Rinocchio [25] lift the LPCP/QAP framework to ring arithmetic, but prove relations at the *ciphertext* level and thus must still simulate the maintenance operations, keeping them far from practical verifiable FHE. Beyond ZKPs, integrity of outsourced FHE has been pursued via homomorphic MACs [14], [20], [40], constant-depth verifiable computation [7], [21], ABFT redundancy [55], trusted hardware [50], and MPC [34], as well as sumcheck-over-rings [58]. These are either restricted to specific computations, require per-level interaction, or rely on trust assumptions, and none yield a succinct, ring-native proof scalable to large circuits.

For distributed proving, prior systems [39], [43], [54], [60], [63], [64], [65] are all field-based and rely on KZG/-pairing machinery that does not transfer to ring arithmetic. **FHE-over-SNARK.** The FHE-over-SNARK paradigm, first proposed in [27], is a more promising line that proves at the (F)HE *plaintext* level, leaving ciphertext maintenance outside the relation. It might appear that any SNARK over ring arithmetic, once evaluated homomorphically, immediately yields an FHE-over-SNARK. Unfortunately, several obstacles remain. First, FHE computations include *non-arithmetic* operations, such as rotations, that are not expressible as plaintext ring arithmetic. Second, some FHE schemes keep their plaintexts *SIMD*-packed, which requires the SNARK to preserve this structure. Third, for efficiency, the FHE plaintext space is typically much smaller than the field size a sound SNARK requires, so soundness may need amplification. More importantly, since everything is now evaluated homomorphically, proof generation adds non-trivial noise growth on top of the underlying FHE circuit. Existing constructions navigate these obstacles in different ways. Early ones homomorphically evaluate hash-based SNARKs: FRI in HELIOPOLIS [3], Fractal in Blind ZK [24], and Spartan in Phalanx [66]. These reduce the relation to queries on a polynomial commitment derived homomorphically, and yield proofs that scale with the commitment query complexity. The polynomial commitment affords transparency and public verifiability, but binds the witness into a matrix layout that does not coincide with the native SIMD packing of the plaintext, incurring a repacking that dominates the prover in [66]. [33] keeps the Spartan-plus-code-based-PCS blueprint of Phalanx but instantiates the Ligero-style PCS with a generalized (non-binary) RAA code that encodes in linear time with nearly optimal multiplicative depth.

Like Phalanx, however, it assumes a pre-packed witness, and therefore does not preserve the SIMD structure of the plaintext evaluation. The latest GKR line [53], [67] dispenses with the hash-based IOP altogether. Laminar [53] attains succinct, SIMD-friendly proofs over layered BGV circuits and is our strongest baseline in the SIMD setting, while Lasagne [67] incurs the least noise overhead over the original FHE scheme and outperforms Laminar in the small-SIMD-width setting. Both support non-arithmetic operations such as automorphisms, but their verifier cost and proof size scale linearly with circuit *depth* (and logarithmically with size). A formal asymptotic comparison is shown in Table 1.

TABLE 1: Comparison with prior FHE-over-SNARKs. N denotes the circuit size (the number of mul. and add. gates), D the ring dimension, B the SIMD slots, dp the circuit layers, η the NTT constant ($\eta=5$ for Encore). For the concrete proof cost, we set $N = 2^{20}$, half multiplication half addition gates uniformly distributed, $D = B = 2^{14}$.

	Prover		Proof Size		SIMD
	Asymp.	Concrete (hr)	Asymp.	Concrete	
Blind ZK [24]	$\Omega(B(N \log N + \eta N^{1+1/\eta}))$	$\sim 10^5$	$\Omega(B \log N)$	~ 1.9 TB	$\times$
Phalanx [66]	$\Omega(B(N \log N + \eta N^{1+1/\eta}))$	$\sim 2 \times 10^4$	$\Omega(B(\frac{\sqrt{N}}{D} + B \log N))$	~ 0.9 TB	$\times$
RAA [33]	$\Omega(BN \log N)$	19.57	$O(B\sqrt{N})$	~ 512 GB	$\times$
Laminar [53]	$O(\frac{B}{D} N \log N)$	11.12	$O(dp \log \frac{BN}{dp})$	13.3 MB	$\checkmark$
Lasagne [67]	$O(\frac{dp N \log N}{B})$	23.97	$O(dp \log \frac{BN}{dp})$	142.5 MB	$\checkmark$
Encore	$O(\eta N^{1+1/\eta})$	9.38	$O(1)$	4.64 MB	$\checkmark$
d-Encore	$O(\frac{\eta(N/M)^{1+1/\eta}}{+ \eta M^{1+1/\eta}})$	1.36			

2. Preliminary

We use $\boxed{x}$ to denote the encoding of plaintext x . For two different encoding schemes $\Pi_E^{(1)}, \Pi_E^{(2)}$, we write encodings as $\boxed{x}_1$ and $\boxed{x}_2$ respectively. Let $\cdot$ denote the standard ring multiplication and “ $\otimes$ ” the encoding-ciphertext multiplication between an encoding under some encoding scheme Π_{enc} and a ciphertext under some encryption scheme Π_{fhe} .

Let $[n]$ denote the set $\{1, 2, \dots, n\}$. We use calligraphic capital letters (e.g., $\mathcal{I}, \mathcal{J}$) to denote sets of wire indices.

2.1. Verifiable Computation on Encrypted Data

Indexed relation. We denote by $R_{\text{SAT}} \ni (i, \mathbb{x}, \mathbb{w})$ the circuit satisfiability relation, i.e., for all $(i := (pp, C), \mathbb{x}, \mathbb{w}) \in R_{\text{SAT}}$, where pp is the public parameters and C is the description of an arithmetic circuit, $\mathbb{x}$ is the public inputs, and $\mathbb{w}$ is the secret inputs, we have that $(\mathbb{x}, \mathbb{w})$ is a satisfying assignment of the circuit C under public parameter pp , i.e., $C(pp, \mathbb{x}, \mathbb{w}) = 1$. Let $\mathcal{L}$ be the language for R_{SAT} s.t. $\mathcal{L} := \{(i, \mathbb{x}) \mid \exists \mathbb{w}, R_{\text{SAT}}(i, \mathbb{x}, \mathbb{w}) = 1\}$.

Looking ahead, in our construction, pp includes auxiliary information, such as the ring $\mathcal{R}$. Notice that for an arithmetic circuit, given the public and secret inputs, there is an efficient algorithm that recovers the complete assignment of all wires.

Definition 2.1 (Linear PCP [9]). Given an indexed relation R_{SAT} , an ℓ -query input-independent linear PCP scheme Π_{LPCP} for R_{SAT} and knowledge error ϵ is a tuple of algorithms $\Pi_{\text{LPCP}} = (\text{Query}, \text{Prove}, \text{Verify})$ defined as follows:

- $(\text{VK}, \text{Q}) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, i)$: takes in the security parameter λ and the index i ; outputs the verification key as well as the query.
- $\mathcal{O}^\pi \leftarrow \Pi_{\text{LPCP}}.\text{Prove}(1^\lambda, \mathbb{x}, \mathbb{w})$: takes in the instance and the witness; generates the linear oracle of the proof $\mathcal{O}^\pi$.
- $\{0, 1\} \leftarrow \Pi_{\text{LPCP}}.\text{Verify}(\text{VK}, \mathbb{x}, \text{ans})$: given the verification key, the instance $\mathbb{x}$ and the answer $\text{ans} \leftarrow \mathcal{O}^\pi(\text{Q})$, return 1 if accepted; 0 otherwise.

The LPCP protocol Π_{LPCP} should satisfy the following:

- **Completeness:** For all $(i, \mathbb{x}, \mathbb{w})$ s.t., $R_{\text{SAT}}(i, \mathbb{x}, \mathbb{w}) = 1$, $(\text{VK}, \text{Q}) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, i)$, $\mathcal{O}^\pi \leftarrow \Pi_{\text{LPCP}}.\text{Prove}(1^\lambda, \mathbb{x}, \mathbb{w})$ and $\text{ans} \leftarrow \mathcal{O}^\pi(\text{Q})$, we have: $\Pr[\Pi_{\text{LPCP}}.\text{Verify}(\text{VK}, \mathbb{x}, \text{ans})] = 1$.
- **Statistical Knowledge Soundness:** There exists an efficient extractor $\mathcal{E}$, s.t., for all $(i, \mathbb{x})$, where $\Pr[\text{Verify}(\text{VK}, \mathbb{x}, \text{ans}) = 1 \mid (\text{VK}, \text{Q}) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, i), \text{ans} \leftarrow \mathcal{O}^\pi(\text{Q})] > \epsilon$, we have $\mathbb{w} \leftarrow \mathcal{E}^{\mathcal{O}^\pi}(i, \mathbb{x})$ and $R_{\text{SAT}}(i, \mathbb{x}, \mathbb{w}) = 1$.

Notice that the statistical knowledge soundness could be relaxed to statistical soundness, defined as follows.

- **Statistical Soundness:** For all $(i, \mathbb{x}) \notin \mathcal{L}$, for all $\mathcal{O}^\pi \leftarrow \Pi_{\text{LPCP}}.\text{Prove}(1^\lambda, \mathbb{x}, \mathbb{w})$, we have $\Pr[\Pi_{\text{LPCP}}.\text{Verify}(\text{VK}, \mathbb{x}, \text{ans}) = 1 \mid (\text{VK}, \text{Q}) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, i), \text{ans} \leftarrow \mathcal{O}^\pi(\text{Q})] \leq \epsilon$.

We represent a circuit that evaluates ring arithmetic as a *quadratic ring program* as proposed in [25], which is adapted from its non-ring arithmetic version in [52].

Definition 2.2 (Quadratic Arithmetic Program). Let C be an arithmetic circuit over a finite commutative ring $\mathcal{R}$ with n inputs and n' outputs, and we associate each input of C and each output of a multiplication gate with an index $k \in \{0, \dots, w\}$. W.l.o.g., we assume the input wires have indices $(1, \dots, n)$ and output wires $(w - n' + 1, \dots, w)$. A *Quadratic Arithmetic Program* (QAP) encoding C consists of three sets of polynomials, $\{V_k(x)\}$, $\{W_k(x)\}$, and $\{Y_k(x)\}$ for $k \in \{0, \dots, w\}$, and a target polynomial $t(x)$. We say that the QAP program computes C if: $(c_1, \dots, c_n, c_{w-n'+1}, \dots, c_w) \in \mathcal{R}^{n+n'}$ is a valid input/output assignment of the circuit C if and only if there exists an assignment $(c_{n+1}, \dots, c_{w-n'}) \in \mathcal{R}^{w-n-n'}$ such that $t(x)$ divides $p(x) = V(x) \cdot W(x) - Y(x)$, where: $V(x) = V_0(x) + \sum_{k \in [w]} c_k V_k(x)$, $W(x) = W_0(x) + \sum_{k \in [w]} c_k W_k(x)$, $Y(x) = Y_0(x) + \sum_{k \in [w]} c_k Y_k(x)$.

As in prior work [25], [52], a QAP for a circuit C can be constructed as follows. We pick a root $r_g \in \mathcal{R}$ for each multiplication gate g and define the target polynomial as $t(x) = \prod_g (x - r_g)$. For each wire index k , a polynomial $V_k(x)$ is defined so that $V_k(r_g) = 1$ if the k -th wire is the left input of the multiplication gate g ; and $V_k(r_g) = 0$ otherwise. Similarly, $W_k(x)$ encodes the right inputs, and

$Y_k(x)$ encodes the outputs. We denote by $H(x)$ the quotient polynomial $H(x) := \frac{V(x)W(x) - Y(x)}{t(x)}$. As discussed in [6], [28], [35], QAP is a 4-query LPCP.

Verifiable Computation on Encrypted Data (VCoED) [27] captures the security model in which a client outsources homomorphic computation over encrypted data to an untrusted server while preserving both client-side privacy and soundness. We adapt the standard definition [27], [66] to the LPCP-based framework.

Definition 2.3 (Verifiable Computation on Encrypted Data). A verifiable computation on encrypted data (VCoED) protocol Π consists of the following PPT algorithms:

- $(\text{VK}, \text{PK}) \leftarrow \Pi.\text{Setup}(1^\lambda, i)$: takes in the security parameter and the index; outputs the verification key VK and the public proving key PK .
- $\text{ct} \leftarrow \Pi.\text{E}(\text{PK}, m)$: takes in the public proving key and a message m ; outputs the ciphertext ct .
- $(\pi, \text{ct}') \leftarrow \Pi.\text{Prove}(\text{PK}, \mathbb{x}, \mathbb{w})$: given the proving key, and $(\mathbb{x}, \mathbb{w})$; outputs the proof π and the evaluation output ct' .
- $m/\perp \leftarrow \Pi.\text{Verify}(\text{VK}, \mathbb{x}, \text{ct}', \pi)$: verify the proof π given the verification key VK , the statement $\mathbb{x}$ and the evaluation output; outputs the evaluation result in plain as m if the verification passes; and $\perp$ otherwise.

A VCoED should satisfy the following properties:

- **Completeness:** For the security parameter λ , for all $(i, \mathbb{x}, \mathbb{w})$ s.t. $R_{\text{SAT}}(i, \mathbb{x}, \mathbb{w}) = 1$, let $(\text{VK}, \text{PK}) \leftarrow \Pi.\text{Setup}(1^\lambda, i)$, $(\pi, \text{ct}') \leftarrow \Pi.\text{Prove}(\text{PK}, \mathbb{x}, \mathbb{w})$, we have $\Pr[\Pi.\text{Verify}(\text{VK}, \mathbb{x}, \text{ct}', \pi) = 1] = 1$.
- **Outsourceability:** A VCoED is outsourceable if $|\pi|$ and the runtime of $\Pi.\text{Verify}$ are both $o(|w|)$.
- **Client-privacy:** A VCoED is client-private if the prover knows no information of the data encrypted by the client.
- **Soundness:** For all $(i, \mathbb{x}) \notin \mathcal{L}$, $(\text{VK}, \text{PK}) \leftarrow \Pi.\text{Setup}(1^\lambda, i)$, for every PPT adversary $\mathcal{A}$, $(\pi, \text{ct}') \leftarrow \mathcal{A}(\text{PK}, \mathbb{x})$, we have $\Pr[\Pi.\text{Verify}(\text{VK}, \mathbb{x}, \text{ct}', \pi) = 1] = \text{negl}(\lambda)$.

A VCoED is *knowledge sound* if we replace the *soundness* property above with the following:

- **Knowledge soundness:** for every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$, s.t., for all $(i, \mathbb{x})$, $(\text{VK}, \text{PK}) \leftarrow \Pi.\text{Setup}(1^\lambda, i)$, $(\pi, \text{ct}') \leftarrow \mathcal{A}(\text{PK}, \mathbb{x}; z)$, $\mathbb{w} \leftarrow \mathcal{E}(1^\lambda, \text{PK}, \text{VK}, \mathbb{x}; z)$, we have $\Pr[\Pi.\text{Verify}(\text{VK}, \mathbb{x}, \text{ct}', \pi) = 1 \wedge R_{\text{SAT}}(i, \mathbb{x}, \mathbb{w}) = 0] = \text{negl}(\lambda)$.

Notice that different from the definition for SNARG, we introduce two properties: outsourceability and client-privacy for VCoED, and we do not consider the server-side privacy in this work (see Remark 4.1). We also generalize this definition to a distributed setting; see Def. A.1 in § A.

2.2. Linear Targeted Malleability

Assumption 2.1 (Strictly Linear Targeted Malleability). A vector encryption scheme $\Pi_E := (\text{Setup}, \text{E}, \text{D})$ satisfies strictly linear targeted malleability if for every PPT algorithms $\mathcal{A}$, there exists a possibly computationally unbounded

simulator $\mathcal{S}$, s.t. for all auxiliary input $z \in \{0, 1\}^{\text{poly}(\lambda)}$, where λ is the security parameter and ℓ is the vector length, the following two distributions are computationally indistinguishable:

• **Real Distribution:**

- 1) The challenger samples $(\text{st}, \{\vec{\mu}_i\}_{i \in [K]})$ for $\vec{\mu}_i \xleftarrow{\$} \mathcal{M}^\ell$;
- 2) The challenger invokes $(\text{pp}, \text{sk}) \leftarrow \Pi_E.\text{Setup}(1^\lambda)$, for $i \in [K]$, generates $\text{ct}_i \leftarrow \Pi_E.E(\text{sk}, \vec{\mu}_i)$ as the encodings;
- 3) Upon receiving all the encodings, $\text{ct}' \leftarrow \mathcal{A}(\text{pp}, \{\text{ct}_i\}_{i \in [K]}; z)$;
- 4) The challenger decrypts $m'_0 \leftarrow \Pi_E.D(\text{sk}, \text{ct}')$;
- 5) Output $(\vec{m}'_0, \{\vec{\mu}_i\}_{i \in [K]}, \text{st})$.

• **Ideal Distribution:**

- 1) The challenger samples $(\text{st}, \{\vec{\mu}_i\}_{i \in [K]})$ for $\vec{\mu}_i \xleftarrow{\$} \mathcal{M}^\ell$;
- 2) The simulator generates $\pi \leftarrow \mathcal{S}(z)$, where $\pi := \{c_i\}_{i \in [K]} \in \mathcal{M}^K$;
- 3) The challenger computes $\vec{m}'_1 := \sum_{i \in [K]} c_i \cdot \vec{\mu}_i$;
- 4) Output $(\vec{m}'_1, \{\vec{\mu}_i\}_{i \in [K]}, \text{st})$.

3. Technical Overview

We now present Encore, a SNARK for the satisfiability of an arithmetic circuit *over plaintexts of an FHE scheme*. By following the FHE-over-SNARK paradigm, Encore avoids simulating FHE maintenance operations. Encore also preserves the original SIMD structure of the payload data, and is agnostic to the circuit structure. We then extend to a distributed prover via a novel aggregation framework.

The technical overview is structured as follows. § 3.1 summarizes the linear PCP (LPCP) framework that serves as the skeleton of our protocol, and explains why a direct application of LPCP to verifiable FHE is impractical. § 3.2 introduces our key technique, the multi-key multiplication, that elevates the proven relation from ciphertext to *plaintext* level, together with major optimizations introduced in § 3.3 we deploy in our construction. § 3.4 then presents the aggregation framework for distributed proving while preserving the succinct proof with comparably small aggregation overhead.

3.1. Recap of LPCP

We first summarize the LPCP framework based on QAP from prior works [6], [28], [35]. Given an arithmetic circuit C with N multiplication gates, its QAP representation consists of a target polynomial $t(x)$ and three sets of polynomials $\{V_k(x), W_k(x), Y_k(x)\}_{k \in \mathcal{I}}$ of degree N for each wire $k \in \mathcal{I}$, where $\mathcal{I}$ is the set of wire indices². Given the witness, i.e., the wire-values $\{c_k\}_{k \in \mathcal{I}}$, we define $V(x) = \sum_{k \in \mathcal{I}} c_k V_k(x)$, and analogously W, Y . The witness satisfies C iff $t(x) \mid$

$V(x)W(x) - Y(x)$, i.e., there exists a quotient polynomial $H(x) := \frac{V(x)W(x) - Y(x)}{t(x)}$ of degree $\leq N$.

The goal is for the prover to convince the verifier that she knows such a valid witness $\{c_k\}_{k \in \mathcal{I}}$. First, the verifier samples a random s and generates the following *query matrix* with 4 rows and $(N + w + 1)$ columns where $w = |\mathcal{I}|$:

$$\underbrace{\begin{pmatrix} V_1(s) & \dots & V_w(s) & 0 & 0 & \dots & 0 \\ W_1(s) & \dots & W_w(s) & 0 & 0 & \dots & 0 \\ Y_1(s) & \dots & Y_w(s) & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & s & \dots & s^N \end{pmatrix}}_{(N+w+1) \text{ columns}}. \quad (1)$$

Each column then is encoded using a vector encoding scheme Π_{enc} satisfying the strict linear targeted malleability (LTM) property (see Assumption 2.1), which intuitively states that the prover can only compute linear combinations of the encodings [9], [25], [35], [56]. We denote the encodings with $\{\text{enc}_k\}_{k \in [N+w+1]}$. All encodings are then appended to crs. The prover uses $\{c_k\}_{k \in \mathcal{I}}$ to derive the QAP polynomials $V(x), W(x), Y(x)$ and compute the coefficients of the quotient polynomial $H(x) = h_1 + \dots + h_{N+1}x^N$. Let the linear proof of LPCP be: $\{c'_k\}_{k \in [N+w+1]} \leftarrow \{c_k\}_{k \in \mathcal{I}} \cup \{h_{k'}\}_{k' \in [N+1]}$. The prover then computes the inner product:

$$\text{ans} = \sum_k c'_k \diamond \text{enc}_k = \left(\begin{matrix} V(s) & W(s) & Y(s) & H(s) \end{matrix} \right)^\top, \quad (2)$$

where $\diamond$ denotes the *plaintext-encoding multiplications*. The verifier decodes ans to recover $V(s), W(s), Y(s), H(s)$, and checks if $H(s) \cdot t(s) = V(s) \cdot W(s) - Y(s)$.

The soundness guarantee relies on the strict LTM of Π_{enc} , which restricts the prover to linear operations w.r.t. the encodings $\{\text{enc}_k\}_{k \in [N+w+1]}$ in crs. I.e., any ans produced by the prover must be a linear combination of the crs encodings with plaintexts $\{c'_k\}$. The verifier is then convinced that these plaintexts constitute a valid witness whose corresponding QAP identity check holds at the random evaluation point s . The prover runtime for the LPCP is $O(N \log N)$, dominated by deriving the coefficients of $H(s)$ via NTT. The light verification costs simply $O(|\mathbb{X}|)$, independent of the witness size, where $|\mathbb{X}|$ is size of the public inputs and outputs, and the succinct proof is of size strictly $O(1)$.

Applying LPCP to VFHE. The above protocol is under the SNARK-over-FHE paradigm, where the wire values c_k are ciphertexts under some (F)HE scheme Π_{fhe} and the proof certifies the homomorphic evaluations, i.e., it works at the *ciphertext* level. This approach faces two obstacles. First, the (F)HE maintenance operations, such as relinearization, modulus switching, and bootstrapping, are non-arithmetic; hence, simulating them via homomorphic multiplications and additions inflates the circuit size by orders of magnitude. Second, for the typical (F)HE parameter sets, the ciphertext consists of multiple ring elements with large dimension, which complicates the choice of parameters for the encoding scheme. Therefore, we face the following dilemma: we can use an encoding scheme compatible with the large ciphertext space, which would blow up the proof size, or we can use an encoding scheme over a smaller field, but in this case, reducing the large-ring operations to field operations would dramatically increase circuit complexity.

2. In the formal protocol, $\mathcal{I}$ is split into the public input/output set $\mathcal{I}_{\text{io}}$ and the other $\mathcal{I}_{\text{mid}} := \mathcal{I} \setminus \mathcal{I}_{\text{io}}$. The verifier verifies the validity of $\{c_k\}_{k \in \mathcal{I}_{\text{mid}}}$ w.r.t. the public input/output.

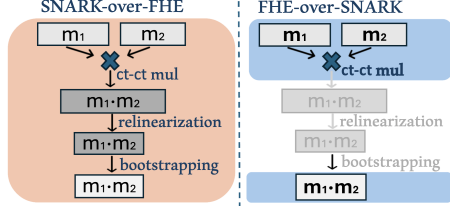


Figure 1: For a circuit consisting of a single ciphertext-ciphertext multiplication followed by a relinearization and a bootstrapping to reduce noise, the naive approach folds all ciphertext-level operations into the proof relation. In contrast, our FHE-over-SNARK design directly proves evaluation of the underlying plaintexts, i.e., only the multiplication, leaving the maintenance operations outside the relation.

3.2. Our Solution: ct-enc Multiplication

We overcome the above obstacles by transforming this approach to FHE-over-SNARK, where both the proof and the verification occur at the plaintext level of the (F)HE scheme Π_{fhe} . To achieve this, we do not treat c_k as a plaintext of the encoding scheme but view it purely and directly as a *ciphertext*³ ct_k with underlying plaintext m_k . Our goal is to design a protocol with proof and verification carried out at the plaintext level rather than to encode ct_k wholesale and operate on the ciphertext level. The first step is to show how the prover can compute an encoding of $V(s) = \sum_k m_k \cdot V_k(s)$ (similarly for $W(s), Y(s)$ and $H(s)$), for the QAP corresponding to the *plaintext computation*. In this case, the verifier decodes and certifies the QAP identity check for the plaintext computation directly, leaving the non-arithmetic maintenance operations of Π_{fhe} outside the proven relation. See an illustration in Fig. 1.

Now the prover has access to encodings (for the proof generation) and ciphertexts (for the (F)HE computation) and needs to homomorphically compute under both schemes. We take inspiration from multi-key FHE [48] and set the encoding and encryption schemes to have the same structure but different secret keys. Specifically, both the encryption and the encoding schemes follow the structure of the BFV/BGV-style FHE scheme [11], [12], [19], where the encoding scheme does not contain a relinearization key, so it satisfies strict LTM. We then replace the plaintext-encoding multiplication of the classical LPCP approach with a *ciphertext-encoding multiplication* (ct-enc multiplication), denoted by $\otimes$. To distinguish the two schemes, write $\Pi_E^{(1)} := \Pi_{\text{fhe}}$, $\Pi_E^{(2)} := \Pi_{\text{enc}}$, with $\boxed{\cdot}_1, \boxed{\cdot}_2$ denoting encodings/encryptions under each and sk_1, sk_2 denoting their secret keys. For a general encoding, we write enc , and ct for a general ciphertext.

Now, the ciphertext-encoding operation combines $\boxed{m_k}_1 \otimes \boxed{V_k(s)}_2$, where each encoding/ciphertext has the form $(a, b) \in \mathcal{R}^2$. The $\otimes$ operation acts as:

$$(a_1, b_1) \otimes (a_2, b_2) \rightarrow (a_1 a_2, a_1 b_2, a_2 b_1, b_1 b_2) \in \mathcal{R}^4. \quad (3)$$

3. Similar to prior works [53], [67], we consider the case where all inputs are provided by the clients, i.e., the case of VCoED [27].

Denote the result of form $\mathcal{R}^4$ as a *tensored encoding*. The final proof is then: $\text{ans} := \begin{pmatrix} V(s) & W(s) & Y(s) & H(s) \end{pmatrix}^\top = \sum_k \text{ct}_k \otimes \text{enc}_k$ as in Eq. (2), where $\text{ct}_k := \boxed{m_k}_1$ and enc_k as defined in Eq. (1). During verification, the verifier computes the inner product of ans with $(sk_1 sk_2, -sk_1, -sk_2, 1)$ to recover the first slot in Eq. (2) as $a_1 a_2 sk_1 sk_2 - a_1 \vec{b}_2[1] sk_1 - a_2 b_1 sk_2 + b_1 \vec{b}_2[1] = a_2 sk_2 (a_1 sk_1 - b_1) - \vec{b}_2[1] (a_1 sk_1 - b_1) = (b_1 - a_1 sk_1) (\vec{b}_2[1] - a_2 sk_2) \approx \sum_k m_k \cdot V_k(s) = V(s)$, which is precisely the plaintext-level LPCP response⁴.

In effect, the ct-enc tensor and its joint decryption under sk_1, sk_2 enable the proof of an arithmetic relation over the plaintext space, even though the underlying ciphertext transformations are non-arithmetic. In the VCoED setting, it is also the verifier that delegates the homomorphic computation to the server, so the verifier holds both secret keys.

High-level workflow. By replacing the plaintext-encoding multiplication with the ct-enc multiplication described above, we summarize the Encore protocol as follows. Given an (F)HE circuit and its QAP, the verifier, who is also the client outsourcing the (F)HE computation, instantiates the (F)HE encryption scheme $\Pi_E^{(1)}$ and the encoding scheme $\Pi_E^{(2)}$, and generates the encodings of the query matrix in Eq. (1) as $\text{crs} := \{\text{enc}_k\}$ under $\Pi_E^{(2)}$. The prover, who is also the delegated server, evaluates the (F)HE circuit on the encrypted data under $\Pi_E^{(1)}$, and derives the linear proof from the resulting intermediate ciphertexts under $\Pi_E^{(1)}$. To answer the query encoded under $\Pi_E^{(2)}$, the prover takes the inner product of crs with its linear proof via the ciphertext-encoding multiplication. On receiving the answer, the verifier decrypts it under the keys of both $\Pi_E^{(1)}$ and $\Pi_E^{(2)}$ and runs the standard LPCP verification to check the QAP identity.

Overhead. Our ciphertext-encoding multiplication $\text{ct} \otimes \text{enc}$ requires the two operands to share the same parameter sets, including the ciphertext modulus and plaintext modulus⁵. The two schemes are thus parameterized *together*: $\Pi_E^{(1)}$ is sized for the (F)HE evaluation with potentially large ciphertext modulus for deep circuits, while $\Pi_E^{(2)}$ is sized for the SNARK protocol, needing just enough noise budget for proof generation. In practice, the (F)HE ciphertexts under $\Pi_E^{(1)}$ are modulus-switched down to match the modulus level of $\Pi_E^{(2)}$ right before the ct-enc multiplication. This avoids the parameter blow-up of the naive SNARK-over-FHE approach, where $\Pi_E^{(2)}$ is forced to embed the entire ciphertext space, inflating both the ring dimension and the modulus.

On the original homomorphic evaluation side, the price is a moderately larger noise budget, since the data ciphertexts $\{\text{ct}_k\}$ must tolerate the additional homomorphic work performed during proof generation, requiring a few more limbs and possibly a larger ring dimension on top of the underlying (F)HE evaluation. Nevertheless, a typical BGV instantiation

4. Note that this direct decryption only works for (F)HE schemes that encode the message in the LSBs, such as BGV. For BFV, we need to maintain a scaling factor for recovering the correct message.

5. In § 4, we use a field extension of the underlying plaintext space $\mathcal{R}_p$ to amplify soundness.

with $\log p \approx 23$ and $\log q \approx 300$ renders the naive approach infeasible, whereas our protocol yields a proof of under 5 MB. A detailed comparison with other works appears in § 6.

Soundness. Unlike prior LPCP-based SNARKs [35], [56] where the prover performs only scalar-encoding multiplications and additions with $\Pi_E^{(2)}$, our prover performs a tensor product between the encoding and FHE scheme. Soundness therefore relies on a *multi-key* variant of the strictly linear targeted malleability property: The decryption result of the final proof ans under sk_1, sk_2 must be expressible as an inner product of a list of plaintexts with the original message encoded under $\{\text{enc}_k\}$ in the crs. While this may appear stronger than the standard LTM, we show in Lemma 4.2 that it reduces directly to the standard LTM (Assumption 2.1) of $\Pi_E^{(2)}$ alone. Intuitively, an adversary against strictly LTM of $\Pi_E^{(2)}$ could instantiate the encryption scheme $\Pi_E^{(1)}$ itself and then invoke the adversary that breaks the multi-key variant. On receiving the answer of form $\square_{1 \otimes 2}$, it reduces it to $\square_2$ given the secret key of $\Pi_E^{(1)}$, thereby breaking standard LTM. See more details in § 4. Our construction thus inherits the same security foundation as prior LPCP-based SNARKs.

A further complication arises because the (F)HE plaintext modulus p is small (e.g., $\log p = 23$ under our parameters), and the soundness error for LPCP is $2N/p$, where N is the circuit size. To amplify soundness, we lift the random evaluation point s chosen by the verifier from $\mathcal{R}_p$ to an extension ring $\mathcal{R}_{p^d}$, achieving soundness comparable to related works [53], [67]. Since $\mathcal{R}_p$ is a subring of $\mathcal{R}_{p^d}$, this amplification does not affect the plaintext of $\Pi_E^{(1)}$. The ct-enc multiplication combines a base-field ciphertext, encrypting $m_k \in \mathcal{R}_p$, with an extension-field ciphertext, encoding $V_k(s) \in \mathcal{R}_{p^d}$, and produces a result in the extension field.

3.3. Optimizations

We describe two optimizations that significantly reduce the prover runtime and proof size.

Coset and Lagrange basis evaluation. Recall from § 3.2, the prover computes the quotient $H(x) := (V(x)W(x) - Y(x))/t(x)$, which dominates the prover cost. A naive polynomial division requires $O(N^2)$ homomorphic operations. Our goal is to reduce this to $O(\eta N^{1+1/\eta})$ for a small constant η , while keeping the multiplicative depth small.

The standard route is to multiply and divide polynomials via NTT. Note that $V(x), W(x), Y(x)$ are defined in a way that their evaluations on the roots of unity $\{\omega^k\}_{k \in [N]}$ (which also index the multiplication gates) equal the corresponding wire values. Therefore, $V(x)W(x) - Y(x)$ and $t(x)$ vanish on $\{\omega^k\}$ and division on these points is undefined. We therefore shift the evaluations to a coset $\{\xi\omega^k\}_{k \in [N]}$ for a nontrivial ξ . A coset shift naively takes three passes: an INTT to recover coefficients, a layer of scalar multiplications pairing the k -th coefficient with ξ^k , and an NTT to evaluate on the coset. Following [30], we fold the ξ^k scaling into the subsequent NTT, removing the separate scalar-multiplication layer. With it, the prover obtains the encrypted coset evaluations $\{H(\xi\omega^k)\}$ using two (I)NTTs.

At this point, the prover holds H as coset evaluations, not coefficients, and the query in Eq. (1) expects coefficients, which then naively requires a third (I)NTT to convert. To eliminate this conversion entirely, instead of $\{s^k\}_{k \in [N]}$, we precompute a *Lagrange-basis query* for H on the coset by replacing s^k with $h_k := L_k(s) \cdot t(\xi\omega^k)^{-1}$, where the Lagrange basis is defined as: $L_k(s) = \prod_{j \in [N], j \neq k} \frac{s - \xi\omega^j}{\xi(\omega^k - \omega^j)} = \frac{\omega^k (s^N - \xi^N)}{N \xi^{N-1} (s - \xi\omega^k)}$, which is independent of the instance and witness. The prover then computes directly an inner product of the coset evaluations with $\{h_k\}_{k \in [N]}$ via the ciphertext-encoding multiplication, so the coset evaluations are consumed as-is. This removes the final $O(N \log N)$ INTT pass and the noise budget it would consume.

Finally, since every (I)NTT is evaluated homomorphically, a textbook $\log N$ -depth NTT with $O(N \log N)$ operations would incur $\log N$ multiplicative depth. We instead use an η -depth variant with $O(\eta N^{1+1/\eta})$ operations [26], [66], trading a higher operation count for a constant-depth circuit that confines noise growth to a few multiplication layers.

Ring switching. As noted above, the homomorphic procedure of the proof generation adds noise on top of the payload (F)HE computation. This added noise can push the required ciphertext modulus past a security threshold, forcing a larger ring dimension than the payload data alone would need. With standard parameters [2], ring dimension $D = 2^{13}$ supports $\log q \leq 218$, whereas $D = 2^{14}$ is needed if $\log q$ goes as large as 436. As the homomorphic evaluation proceeds, modulus switching gradually reduces $\log q$. Once $\log q$ falls below the threshold at which the smaller dimension is again secure, the prover performs a *ring switch* to bring down the ring dimension. This helps in two ways: all subsequent homomorphic operations run on the smaller ring at roughly half the per-operation cost; and in the end, the ring carrying the proof is realigned with the SIMD packing of the payload, so no slots are wasted.

For instance, consider an (F)HE circuit of multiplicative depth 6 with SIMD dimension 2^{14} of data payload. Although the data alone and the incurred noise of the (F)HE operations on top of it only require a $D = 2^{14}$ ring, the additional noise from the proof protocol pushes the required modulus past the threshold ($\log q \approx 436$), forcing $D = 2^{15}$, where the data fills only half the slots. The modulus switching gradually brings q back below the threshold, and a ring switch returns the computation to $D = 2^{14}$, eliminating the wasted slots and halving the proof size.

3.4. Distributed Prover: Aggregation Framework

Building on Encore (§ 3.2), we present d-Encore, a distributed prover that handles large-scale circuits. At a high level, d-Encore works as follows. Following the partitioning procedure in Hekaton [54], we split the large circuit into M sub-circuits, each of size $N' = N/M$. Each of the M sub-provers proves the honest evaluation of one sub-circuit, and an aggregator composes the M sub-proofs into a single succinct proof. The sub-provers and the aggregator are treated

as the malicious *prover side* from the perspective of the verifier and may collude. The resulting protocol has a prover-side runtime to be $O(\eta N^{1+1/\eta} + \eta M^{1+1/\eta})$ for some small constant η , with an aggregated succinct final proof. We present key design challenges for this distributed setting.

Aggregation circuit. The natural way to compose sub-proofs is for the aggregator to perform the QAP identity checks $V_i(s)W_i(s) - Y_i(s) = H_i(s)t_i(s)$ for $i \in [M]$ on behalf of the verifier, and then prove that the aggregated check was performed honestly. The verifier checks only the final proof from the aggregator, not the M sub-proofs individually. Unfortunately, the aggregator does not hold the secret keys, so it cannot directly verify the sub-proofs. In our scheme, it *homomorphically* computes an encoding of:

$$\sum_{i \in [M]} \delta_i \cdot (V_i(s)W_i(s) - Y_i(s) - H_i(s)t_i(s)), \quad (4)$$

where δ_i are random batching coefficients chosen during setup. If the evaluation result encodes zero, then with soundness error $1/|\mathbb{F}|$, all sub-proofs satisfy the QAP check.

We call the (F)HE circuit that computes Eq. (4) the *aggregation circuit* (AC). The task of the aggregator is then essentially to use Encore to prove satisfiability of AC, with sub-proofs as the wire values. Let $\bar{V}, \bar{W}, \bar{Y}, \bar{t}$ denote the QAP of AC and $\bar{N} = O(M)$ its size. The verifier samples a random $\bar{s}$ at setup, and the query encodings w.r.t. AC, analogous to Eq. (1), are included in crs. The prover sends the encoded results of Eq. (4) together with the LPCP proof of this AC to the verifier. The verifier is convinced if Eq. (4) decodes to zero and the LPCP proof verifies.

Outer LTM, inner (F)HE. Recall that the sub-proofs $V_i(s), W_i(s), Y_i(s), H_i(s)$ are encodings produced by ciphertext-encoding multiplication of the form in Eq. (3). Evaluating the products $V_i(s) \cdot W_i(s)$ and $H_i(s) \cdot t_i(s)$ therefore requires multiplying two such tensored encodings in $\mathcal{R}^4$, each decodable under the joint key $(sk_1 sk_2, -sk_1, -sk_2, 1)$. Since $\Pi_E^{(2)}$ must satisfy strictly LTM as mentioned in § 3.2, the final result of multiplying two such tensored encodings would yield an $\mathcal{R}^6$ object under the joint key $(sk_1, 1) \otimes (sk_2^2, sk_2, 1)$. Carrying out further $\otimes$ operations against the crs encodings of our aggregation circuit, which is under another encoding scheme $\Pi_E^{(3)}$, expands the result again. In short, the final proof result size grows linearly due to the non-linearizable multiplication between tensored encodings.

The natural fix is to treat the aggregation circuit as an ordinary (F)HE circuit, so that $V_i(s) \cdot W_i(s), H_i(s) \cdot t_i(s)$ are ordinary (F)HE multiplications, in which $(\mathcal{R}_q)^4 \times (\mathcal{R}_q)^4$ relinearizes back to $(\mathcal{R}_q)^4$ with no blow-up, similar to the ciphertext-ciphertext multiplication done under $\Pi_E^{(1)}$ alone. However, relinearization is clearly non-linear in the secret key sk_2 , so making the sub-proof evaluations genuinely homomorphic appears to forfeit strictly LTM on $\Pi_E^{(2)}$, the very property our soundness relies on.

Fortunately, a closer look at our multi-key version of strictly LTM shows the tension is illusory. For a single $\otimes$ operation between $\Pi_E^{(1)}$ and $\Pi_E^{(2)}$, only $\Pi_E^{(2)}$ needs to satisfy the strictly LTM. More generally, for a chain of $\otimes$ operations:

$(\Pi_E^{(1)} \otimes \Pi_E^{(2)}) \otimes \Pi_E^{(3)}$ —first a ciphertext-encoding multiplication under $\Pi_E^{(1)}$ and $\Pi_E^{(2)}$, and then a second $\otimes$ against encodings under $\Pi_E^{(3)}$, which resembles our aggregation step on top of the sub-proof tensored encodings—only the *outer* encoding scheme $\Pi_E^{(3)}$ must satisfy strictly LTM, while the *inner* encoding schemes $\Pi_E^{(1)}, \Pi_E^{(2)}$ may support arbitrary (F)HE operations (see Lemma 4.2 and Remark 4.1). This is precisely what the aggregation framework needs: the results of $V_i(s) \cdot W_i(s), H_i(s) \cdot t_i(s)$ can be relinearized back to $\mathcal{R}^4$ elements, while the soundness is guaranteed by enforcing the strictly LTM at the aggregation layer.

Unified inner scheme. So far the aggregation framework uses three encryption/encoding schemes: $\Pi_E^{(1)}$ for the underlying (F)HE on data payload, $\Pi_E^{(2)}$ for the sub-proof query encodings, and $\Pi_E^{(3)}$ for the final aggregation, with sub-proofs under the joint key $\in \mathcal{R}^4$ derived from sk_1, sk_2 and the final proofs under the joint key $\in \mathcal{R}^8$ derived from sk_1, sk_2, sk_3 . The reason the sub-proof queries were encoded under $\Pi_E^{(2)}$ rather than $\Pi_E^{(1)}$ in Encore is functional: the verifier publishes evaluation keys of $\Pi_E^{(1)}$ to support non-linear operations, whereas $\Pi_E^{(2)}$ must satisfy strictly LTM. Their roles differ.

But as explained above, based on our multi-key strictly LTM (Lemma 4.2), under the distributed setting, both $\Pi_E^{(1)}$ and $\Pi_E^{(2)}$ serve the same *inner* role within the aggregation framework. To be specific, $\Pi_E^{(1)}$ handles (F)HE evaluations on the payload data, and $\Pi_E^{(2)}$ is used for homomorphic evaluations on the sub-proofs inside the aggregation circuit; neither requires strictly LTM there. We therefore collapse them into a single inner scheme $\Pi_E^{(1)}$.

Now, the sub-proofs are plain $\Pi_E^{(1)}$ ciphertexts in $\mathcal{R}^2$, obtained by standard *ciphertext-ciphertext* multiplications with relinearization between the (F)HE payload ciphertext and the sub-circuit query ciphertexts. Only the crs encoding scheme for the aggregation circuit, which we now rename $\Pi_E^{(2)}$, must satisfy strictly LTM. The final proof is then the result of *ciphertext-encoding* multiplication of form $\boxed{\cdot}_{1 \otimes 2}$ between the sub-proofs under $\Pi_E^{(1)}$ and aggregation-circuit query encodings under $\Pi_E^{(2)}$. Due to this simplification, the decryption uses a joint key $\in \mathcal{R}^4$ formed from sk_1, sk_2 rather than a key $\in \mathcal{R}^8$ formed from sk_1, sk_2, sk_3 , thereby reducing the proof size by a factor of two.

Entangled encodings. The construction so far has a critical flaw: the sub-circuit queries and the AC queries are generated and encoded *independently*. The aggregation check never actually inspects whether the sub-proofs fed as the inputs of the aggregation circuit are real “sub-proofs”. To see why this is exploitable, note that by publishing query encodings that encode $\bar{V}_i(\bar{s})$ for AC and $V_{i,k}(s)$ for the sub-circuit i as in Eq. (1), the verifier expects to receive the answer with values $V_i(s) \cdot \bar{V}_{i'}(\bar{s})$, where i' is the wire index in AC that is supposed to carry the sub-proof value $V_i(s)$, with $V_i(s)$ honestly derived as $\sum_k m_{i,k} V_{i,k}(s)$. However, nothing stops a malicious prover from ignoring the sub-circuit queries

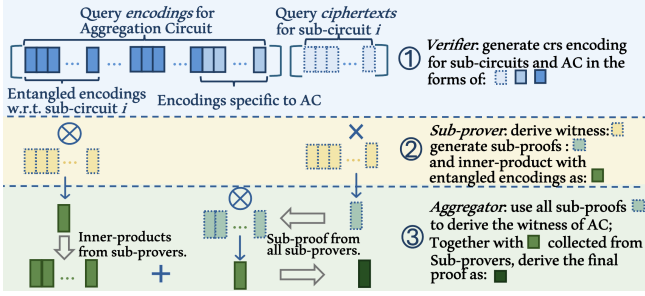


Figure 2: An illustration of the aggregation workflow. We use $\boxed{\cdot}$ to denote a *ciphertext* under $\Pi_E^{(1)}$, including the query ciphertexts for sub-circuits and sub-proofs, and $\boxed{\cdot}$ to denote an *encoding* under $\Pi_E^{(2)}$ or $\Pi_E^{(1)} \otimes \Pi_E^{(2)}$, including the query encodings for aggregation and the final proofs.

$V_{i,k}(s)$ and placing arbitrary values on wire i' in AC. For instance, it can submit sub-proofs as $V_i(s) = W_i(s) = Y_i(s) = H_i(s) = 0$, which satisfy every sub-circuit QAP check vacuously. Nothing in the construction so far binds the wire values in AC to honestly computed sub-proofs.

The fix is thus to make the two query encodings *entangled*. The key move is that the verifier never publishes the aggregation-circuit query on its own, but only in an *entangled* format, so that the aggregation circuit can only be satisfied using genuine sub-proof values. Specifically, consider an AC wire meant to carry the sub-proof value $V_i(s)$, and let $\bar{V}_{i'}(\bar{s})$ be the query polynomial for that wire. Instead of publishing encodings for $\bar{V}_{i'}(\bar{s})$ on its own, the verifier publishes only its product with the sub-circuit query $V_{i,k}(s)$, as a single encoding $\boxed{V_{i,k}(s)\bar{V}_{i'}(\bar{s})}_2$.

Now the sub-prover homomorphically derives the inner product of those encodings against its sub-circuit wire values $\boxed{m_{i,k}}_1$, and yields the encodings of $\sum_k \boxed{m_{i,k}}_1 \otimes \boxed{V_{i,k}(s)\bar{V}_{i'}(\bar{s})}_2 = \boxed{V_i(s) \cdot \bar{V}_{i'}(\bar{s})}_{1 \otimes 2}$. Decrypted under the joint key $(sk_1 sk_2, -sk_1, -sk_2, 1)$, this is $\sum_k m_{i,k} \cdot V_{i,k}(s) \cdot \bar{V}_{i'}(\bar{s}) = V_i(s) \bar{V}_{i'}(\bar{s})$, which is exactly the value the aggregation circuit expects on this wire when it is answered with a genuine sub-proof.

Hence, the entanglement forces the prover side to have the value of the AC wire $i^{(v)}$ bound to the sub-proof value $V_i(s)$. This solves the flaw and allows no shortcut for satisfying the AC check without deriving the legitimate sub-proofs. Analogous entangled encodings handle W_i, Y_i, H_i . Fig. 2 shows an illustration on the high-level workflow of our aggregation. See more details in § C.

4. Secure Encoding Scheme

In this section, we formally define the encoding scheme used in our SNARK and its properties. This encoding scheme can roughly be seen as a vector version of BGV-style encryption, stripped of the multiplicative homomorphism and all public evaluation keys, and extended to an extension

ring, i.e., the plaintext space is $\mathcal{R}_{p^d} := \mathbb{Z}_{p^d}[Y]/f(Y)$, where f is an irreducible polynomial with degree d .

Definition 4.1 (RLWE-based Vector Encoding Construction [9], [35], [56]). Let $\mathcal{R}$ be a cyclotomic ring with ring dimension D , and $p, q \in \mathbb{N}$ be two moduli associated with the plaintext and encoding space respectively. Let χ_σ be a discrete Gaussian over $\mathcal{R}$, and $\mathcal{D}$ denote the secret key distribution. Let d be the extension factor, s.t., $O(p^{-d}) = \text{negl}(\lambda)$, and τ be the sparsification factor, s.t., $p^{-D\tau} = \text{negl}(\lambda)$ for some security parameter λ . Let ℓ be the length of the message vector and set $\ell' := \ell + \tau$. The encoding scheme $\Pi_E := (\text{Setup}, E, D)$ is defined as follows.

- $(pp, sk) \leftarrow \text{Setup}(1^\lambda, 1^{\ell'})$: Sample secret keys $\vec{s}[i] \leftarrow \mathcal{D}, i \in [\ell']$, and the sparsification matrix $T \xleftarrow{\$} \mathcal{R}_{p^d}^{\ell \times \tau}$. Return $sk := (\vec{s}, T)$ and $pp := (p, q, D, d, \ell, \tau, \mathcal{D}, \sigma)$.
- $ct \leftarrow E(sk, \vec{m})$: For $\vec{m} \in \mathcal{R}_{p^d}^\ell$, compute $\vec{m}' = \vec{m}T \in \mathcal{R}_{p^d}^\tau$ and set $\vec{\mu} := \vec{m} \parallel \vec{m}' \in \mathcal{R}_{p^d}^{\ell'}$. Sample $a \xleftarrow{\$} \mathcal{R}_{q^d}$. For $i \in [\ell']$ sample $e_i \leftarrow \chi_\sigma$, and compute $b_i = a \cdot \vec{s}[i] + p \cdot e_i + \vec{\mu}[i] \in \mathcal{R}_{q^d}$. Output $ct := (a, \vec{b}) \in \mathcal{R}_{q^d} \times \mathcal{R}_{q^d}^{\ell'}$.
- $\vec{m} \leftarrow D(sk, ct := (a, \vec{b}))$: For $i \in [\ell']$, compute $\vec{\mu}[i] = \vec{b}[i] - a \cdot \vec{s}[i] \bmod \mathcal{R}_{p^d}$. Parse $\vec{\mu} = \vec{m} \parallel \vec{m}'$ with $\vec{m} \in \mathcal{R}_{p^d}^\ell, \vec{m}' \in \mathcal{R}_{p^d}^\tau$. If $\vec{m}' \neq \vec{m}T$, output $\perp$; otherwise output $\vec{m}$.

Notice that the message vector $\vec{\mu} := \vec{m} \parallel \vec{m}' \in \mathcal{R}_{p^d}^{\ell+\tau}$ has a special structure namely $\vec{m}T = \vec{m}'$, where T is the sparsification matrix. As in prior works [35], [56], this structure “sparsifies” the underlying plaintext space and prevents a malicious prover from generating encodings with valid underlying plaintexts. Thus, the above vector encoding scheme is conjectured to satisfy strictly LTM (Assumption 2.1) [35], [56]. Instead of regular homomorphic multiplication, this scheme allows *ciphertext-encoding* ($ct\text{-}enc$) multiplication, with ciphertext under $\Pi_E^{(1)}$, and encoding under $\Pi_E^{(2)}$. Note that $\Pi_E^{(1)}$ is the FHE scheme, while $\Pi_E^{(2)}$ is the encoding scheme and no relinearization key is provided for this scheme. Thus, a malicious prover can only perform one layer of multiplication w.r.t. the encodings under $\Pi_E^{(2)}$.

Definition 4.2 (Ciphertext-Encoding Multiplication). Let $\Pi_E^{(1)}$ be the BGV encryption scheme [12] and $\Pi_E^{(2)}$ be the vector encoding scheme of Def. 4.1, both with the same plaintext modulus p , ciphertext modulus q , secret key distribution $\mathcal{D}$, and noise distribution χ_σ , with $\Pi_E^{(2)}$ specialized with extension factor d , message vector length ℓ and sparsification factor τ . Set $\ell' = \ell + \tau$. Let the secret keys be $sk_1 := s_1$ and $sk_2 := (\vec{s}_2 \in \mathcal{R}_{q^d}^{\ell'}, T)$, respectively. Let $ct^{(1)}$ be a ciphertext under $\Pi_E^{(1)}$ and $enc^{(2)}$ be an encoding under $\Pi_E^{(2)}$. We define the following procedures.

- $enc^{(1) \otimes (2)} \leftarrow \text{mkMul}(ct^{(1)}, enc^{(2)})$: Given an encoding $ct^{(1)} := (a_1, b_1) \in \mathcal{R}_q \times \mathcal{R}_q$ decodable under sk_1 and an encoding $enc^{(2)} := (a_2, \vec{b}_2) \in \mathcal{R}_{q^d} \times \mathcal{R}_{q^d}^{\ell'}$ decodable under $sk_2 \in \mathcal{R}_{q^d}^{\ell'}$, lift $ct^{(1)}$ into $ct'^{(1)}$ by treating each $\mathcal{R}_q$ -element as a constant polynomial in $\mathcal{R}_{q^d}$. Compute

$d_0 = a_1 a_2, d_2 = a_2 b_1, \vec{d}_1[i] = a_1 \vec{b}_2[i], \vec{d}_3[i] = b_1 \vec{b}_2[i]$, for $i \in [\ell']$. **Output** $\text{enc}^{(1) \otimes (2)} := (d_0, \vec{d}_1, d_2, \vec{d}_3) \in \mathcal{R}_{q^d} \times \mathcal{R}_{q^d}^{\ell'} \times \mathcal{R}_{q^d} \times \mathcal{R}_{q^d}^{\ell'}$.

- $\text{enc}' \leftarrow \text{mkAdd}(\{\text{enc}_i\})$: Given $\text{enc}_i := (d_{0,i}, \vec{d}_{1,i}, d_{2,i}, \vec{d}_{3,i}) \in \mathcal{R}_{q^d} \times \mathcal{R}_{q^d}^{\ell'} \times \mathcal{R}_{q^d} \times \mathcal{R}_{q^d}^{\ell'}$, compute $\text{enc}' := (d'_0, \vec{d}'_1, d'_2, \vec{d}'_3) := (\sum_i d_{0,i}, \sum_i \vec{d}_{1,i}, \sum_i d_{2,i}, \sum_i \vec{d}_{3,i})$.
- $\vec{m} \leftarrow \text{mkDec}(\text{sk}_1 := s_1, \text{sk}_2 := (\vec{s}_2, T), \text{enc}^{(1) \otimes (2)})$: Parse $\text{enc}^{(1) \otimes (2)}$ as $(d_0, \vec{d}_1, d_2, \vec{d}_3)$, compute $\vec{\mu}[i] = d_0 \cdot s_1 \vec{s}_2[i] - \vec{d}_1[i] \cdot s_1 - d_2 \cdot \vec{s}_2[i] + \vec{d}_3[i] \pmod{\mathcal{R}_{p^d}}$. Parse $\vec{\mu} := \vec{m} \parallel \vec{m}'$, $\vec{m} \in \mathcal{R}_{p^d}^{\ell}, \vec{m}' \in \mathcal{R}_{p^d}^{\tau}$. If $\vec{m}' \neq \vec{m}T$, output $\perp$; o.w. $\vec{m}$.

Given ciphertexts $\{\text{ct}_i\}_{i \in [K]}$ under $\Pi_E^{(1)}$ and $\{\text{enc}'_i\}_{i \in [K]}$ under $\Pi_E^{(2)}$, with underlying plaintext messages $\{m_i\}$ and $\{\vec{\mu}_i\}$ respectively, let $\text{enc} = \text{mkAdd}(\{\text{mkMul}(\text{ct}_i, \text{enc}'_i)\})$, and $\vec{m}' = \sum_{i \in [K]} m_i \cdot \vec{\mu}_i$. We say that $\Pi_E^{(1)}$ and $\Pi_E^{(2)}$ satisfy the *ct-enc multiplication* if: $\Pr[\text{mkDec}(\text{sk}_1, \text{sk}_2, \text{enc}) = \vec{m}'] = 1 - \text{negl}(\lambda)$.

The semantic security and the homomorphism of this vector encoding scheme are directly inherited from prior works [9], [12]. Based on [9, Section 4.4], Def. 4.1 also satisfies strict LTM (see Assumption 2.1). Intuitively, the ct-enc multiplication resembles the ciphertext-ciphertext multiplication under BGV-style encryption scheme without relinearization. Thus, it is straightforward to see that under standard BGV parameter sets, $(\Pi_E^{(1)}, \Pi_E^{(2)})$ satisfies the *ct-enc multiplication* in Def. 4.2, which is formally captured in the following lemma.

Lemma 4.1 (Correctness of Ct-Enc Multiplication). Let $\Pi_E^{(1)}, \Pi_E^{(2)}$ be the encryption and encoding scheme as in Def. 4.2. If $\log q \gg 2 \log p + 1/2 \log K + 2 \log B$, $(\Pi_E^{(1)}, \Pi_E^{(2)})$ satisfies the ct-enc multiplication property.

Proof. Observe that the decryption of $\text{ct}^{(1)} = (a_1, b_1) \in \mathcal{R}_q \times \mathcal{R}_q$ computes $b_1 - a_1 s_1 = m + p e_1 \pmod{q}$, with the plaintext recovered performing mod p . Likewise, for $\text{enc}^{(2)} = (a_2, \vec{b}_2)$, we have $\vec{b}_2[i] - a_2 \vec{s}_2[i] = \vec{\mu}[i] + p \vec{e}_2[i] \pmod{q^d}$. Thus, for a single ct-enc multiplication, mkDec under key $(s_1 \vec{s}_2[i], -s_1, -\vec{s}_2[i], 1)$ derives:

$$\begin{aligned} \vec{m}'[i] &= d_0 s_1 \vec{s}_2[i] - \vec{d}_1[i] s_1 - d_2 \vec{s}_2[i] + \vec{d}_3[i] \\ &= a_1 a_2 s_1 \vec{s}_2[i] - a_1 \vec{b}_2[i] s_1 - a_2 b_1 \vec{s}_2[i] + b_1 \vec{b}_2[i] \\ &= (b_1 - a_1 s_1) (\vec{b}_2[i] - a_2 \vec{s}_2[i]) \\ &= m \vec{\mu}[i] + p \cdot (m \vec{e}_2[i] + \vec{\mu}[i] e_1 + p e_1 \vec{e}_2[i]). \end{aligned} \quad (5)$$

Notice that the noise term is dominated by $p e_1 \vec{e}_2[i]$. Thus, the ct-enc multiplication incurs a noise of $O(\log p + 2 \log B)$ bits, where B is the error bound on e_1, e_2 ⁶. The decryption correctness holds as long as the noise does not cause a wrap around and contaminate the message. Thus, after summing over K such results, with $\log q \gg 2 \log p + 1/2 \log K + 2 \log B$, we have that $\text{mkDec}(\text{sk}_1, \text{sk}_2, \text{ct}) = \sum_i m_i \vec{m}'_i = \vec{m}$ except with the $\text{negl}(\lambda)$ probability. $\square$

6. For fresh ciphertexts, this is bounded by $C \cdot \sigma \sqrt{D}$ for a small constant C , with error drawn from χ_σ and ring dimension D .

Concretely, with $D = 2^{14}, \sigma = 3.2, \log p = 23, K = 2^{20}$, the ct-enc multiplication roughly incurs a noise growth of < 50 bits. Different from the normal strictly LTM, our SNARK protocol relies on a *multi-key* variant, since instead of scalar-encoding multiplication operated on top of ciphertexts under Π_E , the prover actually performs ciphertext-encoding multiplication. We formalize our new multi-key strictly LTM assumption and then show that it reduces to the strict-LTM assumption for $\Pi_E^{(2)}$.

Definition 4.3 (Multi-key Strictly Linear Targeted Malleability). Let $\Pi_E^{(1)}$ be an (F)HE encryption scheme and $\Pi_E^{(2)}$ be a vector encoding scheme, satisfying the ciphertext-encoding multiplication in Def. 4.2. Let $\mathcal{R}_p$ be the underlying plaintext space of $\Pi_E^{(1)}$, and the extension space $\mathcal{R}_{p^d}$ the plaintext space of $\Pi_E^{(2)}$. We say that $\Pi_E^{(1)}$ and $\Pi_E^{(2)}$ satisfy the *Multi-key Strictly Linear Targeted Malleability*, if for any PPT adversary $\mathcal{A}$, there exists a possibly computationally unbounded simulator $\mathcal{S}$ such that for all auxiliary input $z \in \{0, 1\}^{\text{poly}(\lambda)}$, the following two distributions are computationally indistinguishable:

• **Real Distribution:**

- 1) The challenger samples $(\text{st}, \{\vec{\mu}_i\}_{i \in [K]}) \xleftarrow{\$} \mathcal{R}_{p^d}^{\ell'}$;
- 2) The challenger gets $(\text{pp}, \text{sk}_1, \text{pk}_1) \leftarrow \Pi_E^{(1)}. \text{Setup}(1^\lambda)$, $(\text{pp}, \text{sk}_2) \leftarrow \Pi_E^{(2)}. \text{Setup}(1^\lambda)$, and generates the encodings $\text{enc}_i \leftarrow \Pi_E^{(2)}. \text{E}(\text{sk}_2, \vec{\mu}_i)$ for $i \in [K]$;
- 3) Upon receiving all the encodings, $\text{ct}^{(1) \otimes (2)} \leftarrow \mathcal{A}(\text{pp}, \text{pk}_1, \{\text{enc}_i\}_{i \in [K]}; z)$;
- 4) The challenger decrypts the result as $\vec{m}'_0 \in \mathcal{R}_{p^d}^{\ell'} \leftarrow \text{mkDec}(\text{sk}_1, \text{sk}_2, \text{ct}^{(1) \otimes (2)})$;
- 5) Output $(\vec{m}'_0, \{\vec{\mu}_i\}_{i \in [K]}, \text{st})$.

• **Ideal Distribution:**

- 1) The challenger samples $(\text{st}, \{\vec{\mu}_i\}_{i \in [K]}) \xleftarrow{\$} \mathcal{R}_{p^d}^{\ell'}$;
- 2) The simulator generates $\pi \leftarrow \mathcal{S}(z)$, where $\pi := \{m_i\}_{i \in [K]} \in \mathcal{R}_{p^d}$;
- 3) The challenger computes $\vec{m}'_1 \in \mathcal{R}_{p^d}^{\ell'} := \sum_{i \in [K]} m_i \cdot \vec{\mu}_i$;
- 4) Output $(\vec{m}'_1, \{\vec{\mu}_i\}_{i \in [K]}, \text{st})$.

Lemma 4.2. Let $\Pi_E^{(1)}$ be the BGV encryption scheme, parameterized with $(p, q, D, \mathcal{D}, \sigma)$ with key pair $(\text{sk}_1, \text{pk}_1)$, and $\Pi_E^{(2)}$ be a vector encoding scheme with algorithms $(\text{Setup}, \text{E}, \text{D})$, parameterized with $(p, q, D, \mathcal{D}, \sigma, d, \ell, \tau)$ with underlying secret key sk_2 . If $\Pi_E^{(2)}$ satisfies strict-LTM (see Assumption 2.1), then $(\Pi_E^{(1)}, \Pi_E^{(2)})$ satisfies the multi-key strict LTM as in Def. 4.3.

Proof. We show that the lemma holds under Assumption 2.1 via contradiction. Assume that there exists a PPT adversary $\mathcal{A}$ for which the output is computationally distinguishable from the ideal distribution (i.e., it breaks the above multi-key strictly LTM), then we build a PPT adversary $\mathcal{A}'$ in the *real*

distribution in Assumption 2.1 to break the standard strictly LTM. W.l.o.g., let $\ell' = 1$.

- W.l.o.g., denote the original sk in Assumption 2.1 as sk_2 .
- After receiving $(pp, \{enc_i\}_{i \in [K]}; z)$ from the challenger, $\mathcal{A}'$ generates $(pp, sk_1, pk_1) \leftarrow \Pi_E^{(1)}.Setup(1^\lambda)$ and invokes $\mathcal{A}(pp, pk_1, \{enc_i\}_{i \in [K]}; z)$.
- After receiving $enc^{(1) \otimes (2)} := (d_0, d_1, d_2, d_3) \in \mathcal{R}^4$ from $\mathcal{A}$, $\mathcal{A}'$ outputs $(d_2 - d_0 \cdot s_1, d_3 - d_1 \cdot s_1) \in \mathcal{R}^2$.

Recall that for RLWE-based encryption scheme, we have the encodings $enc^{(1) \times (2)}$ derived from $mkMul$ to be of the form $(d_0, d_1, d_2, d_3) \in \mathcal{R}^4$. In the above construction, we let $\mathcal{A}'$ derive $(d_2 - d_0 \cdot s_1, d_3 - d_1 \cdot s_1) \in \mathcal{R}^2$ given $sk_1 := s_1$. On the other hand, in the real distribution generated w.r.t. $\mathcal{A}$ (for multi-key LTM), we decrypt (d_0, d_1, d_2, d_3) by performing the inner product with $(s_1 s_2, -s_1, -s_2, 1)$. It is straightforward to see that:

$$\begin{aligned} & (d_3 - d_1 s_1) \cdot 1 - (d_2 - d_0 s_1) \cdot s_2 \\ &= d_0 \cdot s_1 s_2 - d_1 \cdot s_1 - d_2 \cdot s_2 + d_3 \cdot 1 \\ &= \langle (d_0, d_1, d_2, d_3), (s_1 s_2, -s_1, -s_2, 1) \rangle. \end{aligned}$$

Therefore, both decryption approaches render the same decryption result. I.e., the final message $\vec{m}'_0$ recovered from $\mathcal{A}'$ we construct above is the same message as recovered from $\mathcal{A}$. Thus, given the same ideal distribution in Assumption 2.1 and Lemma 4.2, if $\mathcal{A}$ is able to generate some ciphertext $(d_0, d_1, d_2, d_3) \in \mathcal{R}^4$ with the underlying plaintext $\vec{m}'_0$ that breaks the multi-key strictly LTM, then the ciphertext $(d_2 - d_0 s_1, d_3 - d_1 s_1)$, encoding the same plaintext $\vec{m}'_0$, derived by $\mathcal{A}'$ would also break the standard strictly LTM.

Hence, we conclude that $\mathcal{A}'$ has the same advantage as $\mathcal{A}$. Since based on Assumption 2.1, such an $\mathcal{A}$ succeeding with non-negligible probability does not exist, we conclude that $\mathcal{A}'$ cannot exist, and our lemma holds. $\square$

Remark 4.1. From Lemma 4.2, we observe that $\Pi_E^{(1)}$ is not restricted to linear operations, i.e., (F)HE operations including ciphertext-ciphertext multiplication with relinearization are permitted. In our aggregation framework, where the original circuit is partitioned into multiple sub-circuits so that the FHE evaluation and the proof generation are distributed to sub-provers, sub-provers generate the sub-proofs under $\Pi_E^{(1)}$; that is, $\Pi_{enc} = \Pi_{fhe} = \Pi_E^{(1)}$ when computing the sub-proof. Thus, the sub-prover performs standard ciphertext-ciphertext multiplication under $\Pi_E^{(1)}$ followed by relinearization, rather than the ct-enc multiplication which would at least double the ciphertext size. In short, beyond enabling the outer-layer (F)HE circuit that performs the batched verification as in § C, this property preserves the compactness of the final proof by allowing relinearization of internal multiplications.

5. LPCP-based Aggregatable VCoED

The formal protocol of our FHE-over-SNARK scheme, Encore, is given in Fig. 7, and its distributed variant, d-Encore, is given in Fig. 8.

Theorem 5.1. Let $\Pi_E^{(1)}$ be an (F)HE encryption scheme, $\Pi_E^{(2)}$ be an encoding scheme, both satisfying semantic security. If Π_{LPCP} satisfies statistical soundness against linear provers, and $\Pi_E^{(1)}, \Pi_E^{(2)}$ satisfy the *multi-key strictly linear targeted malleability* (Def. 4.3), the Encore protocol given in Fig. 7 is a *sound* VCoED in the preprocessing model.

We defer the formal proof to § B. The completeness and outsourceability are straightforward, and the client-privacy is guaranteed by the semantic security of the encryption scheme $\Pi_E^{(1)}$. For soundness, at a high level, if the prover is able to convince the verifier for $(i, x) \notin \mathcal{L}$, then based on the multi-key strictly LTM, the simulator should be able to simulate a linear proof, from which we extract the witness w , and contradict the statistical soundness of LPCP.

Similar to the LPCP-based prior works [9], [56], if we assume a stronger strictly LTM assumption with *computationally efficient simulation* in Def. 4.3, and the underlying LPCP satisfies knowledge soundness, then Encore is knowledge sound. The multi-key strictly LTM with computationally bounded simulator can also be directly reduced to the standard strictly LTM with computationally bounded simulator. Both definitions of the standard strictly LTM are conjectured to be true and remain secure under the latest attack in [17].

Theorem 5.2. Let $\Pi_E^{(1)}$ be an encryption scheme, $\Pi_E^{(2)}$ be an encoding scheme, both satisfying semantic security. If Π_{LPCP} satisfies knowledge soundness against linear provers, and $\Pi_E^{(1)}, \Pi_E^{(2)}$ satisfy the *multi-key strictly linear targeted malleability with computationally efficient simulation*, the Encore protocol given in Fig. 7 is a *knowledge sound* VCoED in the preprocessing model.

Theorem 5.3 (Distributed VCoED). Let $\Pi_E^{(1)}$ be an encryption scheme, $\Pi_E^{(2)}$ be an encoding scheme, both satisfying semantic security. If Π_{LPCP} satisfies knowledge soundness, and $\Pi_E^{(1)}, \Pi_E^{(2)}$ satisfy the *multi-key strictly linear targeted malleability with computationally efficient simulation*, the d-Encore protocol given in Fig. 8 is a *knowledge sound distributed* VCoED in the preprocessing model.

6. Evaluation

Throughout, we fix the underlying (F)HE plaintext modulus $p = 7340033 = 7 \cdot 2^{20} + 1$, so $\log p = 23$, and sample secret keys from a ternary distribution with fixed Hamming weight $h = 32$. For the query encoding scheme, set the Galois extension degree $d = 6$ over $\mathbb{Z}_p$ to guarantee > 100 bits soundness, with message-vector length $\ell = 4$ and sparsification factor $\tau = 1$, yielding $2 + 2(\ell + \tau) = 12$ ring elements per multi-key ciphertext. A prototype [61] is implemented on top of SEAL [49] for noise analysis.

Cost breakdown. We record the per-step noise consumption and the operation counts of proof generation in Table 2. Notice that step 4 in Table 2 multiplies normal (F)HE ciphertexts by crs extended vector encodings, i.e., $(\mathcal{R}_q)^2 \times (\mathcal{R}_{q^d})^6 \rightarrow (\mathcal{R}_{q^d})^{12}$, and step 5 operates over encodings in $(\mathcal{R}_{q^d})^{12}$. The total noise budget incurred

for the data (F)HE chain due to our SNARK protocol is approximately 360 bits with $\eta = 5$ (i.e., we adapt a 5-layered NTT to optimize the runtime). When applicable, a ring switch can be used to halve the operation cost.

TABLE 2: Per-step noise consumption and operation counts for proof generation, where N is the circuit size.

	Step Description	Noise (bits)	Operations
1	INTT to obtain coefficients of V, W, Y	23η	$O(\eta N^{1+1/\eta}) \mathbb{Z}_p \times \text{ct}$
2	Coset-NTT to obtain coset evaluations of V, W, Y		$O(\eta N^{1+1/\eta}) \text{ct-ct add}$
3	ct-ct mul to obtain $VW - Y$ at coset	50	N (ct-ct mult+relin) N ct-ct add
4	Multi-key mul between (F)HE ciphertexts and crs	50	$2N$ multi-key mul
5	Add all multi-key ct to obtain the final proof ans	10	$2N$ multi-key add

TABLE 3: Microbenchmark of (F)HE operations for one limb with ≈ 50 bits and ring dimension $D = 16384$.

Operation	ct-ct Add	ct-ct Mul	$\mathbb{Z}_p \times \text{ct}$	Relin.	Dec.
Time (ms)	0.008	0.045	0.016	0.775	0.258

Main comparison. For a fair comparison, we adopt the microbenchmark (see Table 3) of prior work [67], which is conducted on a machine with an Intel Xeon Platinum 8592V processor, 256 logical cores, and 512 GB memory, to estimate the runtime. All numbers of prior works are rescaled accordingly. We fix the circuit size to be $N = 2^{20}$ with half multiplication and half addition gates, matching the setup by prior works [53, Table 20]. For our protocol, which handles additions for free, the effective circuit size is 2^{19} (i.e., 2^{19} multiplication gates). For all schemes, the decryption time dominates the verification and thus the verifier time is estimated based on the proof size if not reported by prior works. A detailed derivation of these estimates is deferred to the full version.

Our prover cost is dominated by the (I)NTT of steps 1,2. Since in the non-distributed setting, Encore needs not homomorphically aggregate sub-proofs, it admits a more aggressive layered NTT with $\eta = 5$ over 7 limbs of 50 bits each. The last two steps 4 and 5 are performed w.r.t. the crs encodings, where each encoding has $\ell + \tau = 5 \mathcal{R}_{p^6}$ elements, with extension factor $d = 6$. Table 4 reports prover runtime, proof size, and verifier runtime for ring dimensions $D \in \{2^{13}, 2^{14}, 2^{15}\}$, where the data payload utilizes the full SIMD slots, i.e., $B = D$. The first three works, Blind ZK, Phalanx, RAA, all utilize the SIMD structure for proof evaluations but not for the native FHE parallelism, and thus the cost scales linearly with D .⁷ Our protocol improves on Blind ZK and Phalanx by three to five orders of magnitude across all three metrics. Against RAA, our proof size and verification are

7. These works could potentially be optimized to handle SIMD payloads; since they do not discuss it, we apply a direct $B \times$ factor to their reported numbers, following Laminat [53]. Note that RAA states that its machine is 1.5–2 $\times$ faster than Xeon baselines for SEAL. Thus, our rescaling is conservative in its favour.

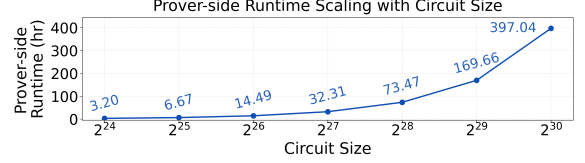


Figure 3: The prover-side time for d-Encore, with a fixed aggregation circuit size of 2^{11} . The original circuit size ranges from 2^{24} to 2^{30} and the sub-circuit size ranges from 2^{14} to 2^{20} correspondingly, ring dimension $D = 2^{15}$.

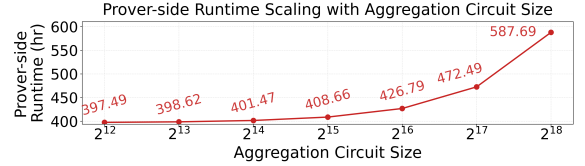


Figure 4: The prover-side time for d-Encore, with a fixed sub-circuit size of 2^{20} , ring dimension $D = 2^{15}$.

also approximately five orders of magnitude better, while the prover runtime is $2.09\times$ faster for $D = 2^{15}$. Against Lasagne, both proof size and verification time are an order of magnitude smaller, while prover runtime is $2.55\times$ faster for $D = 2^{15}$. Against the strongest baseline Laminat, our proof and verification are $\sim 6\times$ smaller with $1.53\times$ slower prover for $D = 2^{13}$, and $1.18\times$ faster with $\sim 1.4\times$ improvements in both proof size and verification for $D = 2^{15}$.

Distributed setting. In Table 4, we instantiate d-Encore with $M = 2^7$ sub-provers, each proving a sub-circuit of size 2^{12} , and an instance (public input/output) of size 2^{12} dominating the aggregation circuit. For $D = 2^{15}$, prover-side finishes in 2.11 hours, $10.54\times$ faster than Laminat, $18.55\times$ faster than RAA, $22.72\times$ faster than Lasagne, while proof size and verification time retain the same advantage.

We further evaluate circuits with 2^{24} – 2^{30} multiplication gates with $D = 2^{15}$ and the number of sub-provers $M = 2^{10}$ in Fig. 3, representing the number of field multiplications up to 2^{45} . Our prover finishes in 400 single-core hours. Moreover, since our proof size and verification time stay constant and all prior works scale at least logarithmically to the circuit size, our proof size and verification are then at least $22.63\times$ smaller than Lasagne and $2.15\times$ smaller than Laminat. We also show the scaling of the d-Encore w.r.t. the aggregation circuit size in Fig. 4 and the scaling across $M \in \{2^9, \dots, 2^{15}\}$ for a circuit of size 2^{26} in Fig. 5.

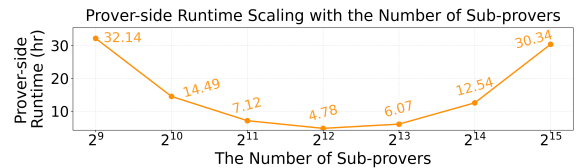


Figure 5: The prover-side time for d-Encore. The original circuit size is 2^{26} . The number of sub-provers ranges from 2^9 to 2^{15} ; the sub-circuit size ranges from 2^{17} to 2^{11} .

TABLE 4: Comparison of prover runtime, proof size, and verifier runtime with plaintext space $\mathcal{R}_p$ ring dimension $D \in \{2^{13}, 2^{14}, 2^{15}\}$, $N = 2^{20}$ for half multiplication gates and half addition gates. The noise overhead column reports the additional bits required by each scheme to support the proof generation on the underlying (F)HE. d-Encore runs in a distributed manner with 128 sub-provers. *We compare with $\text{Laminate}_{\text{lane}}$, which handles SIMD data without automorphism.

Scheme	D = 2 ¹³			D = 2 ¹⁴			D = 2 ¹⁵		
	Prover (hr)	Proof (MB)	Verifier (ms)	Prover (hr)	Proof (MB)	Verifier (ms)	Prover (hr)	Proof (MB)	Verifier (ms)
Blind ZK [24]	~ 53568	~ 950272	~ 10 ⁶	~ 107136	~ 1.9 × 10 ⁶	~ 2.1 × 10 ⁶	~ 214272	~ 3.8 × 10 ⁶	~ 4.2 × 10 ⁶
Phalanx [66]	~ 13248	~ 475955	~ 523931	~ 26496	~ 951910	~ 10 ⁶	~ 52992	~ 1.9 × 10 ⁶	~ 2.1 × 10 ⁶
RAA [33]	~ 9.78	~ 262144	> 250000	~ 19.57	~ 524288	> 500000	~ 39.14	~ 10 ⁶	> 10 ⁶
Laminate* [53]	~ 5.6	13.3	~ 26.06	11.12	13.3	~ 26.06	~ 22.24	13.3	~ 26.06
Lasagne [67]	11.61	135	~ 148.61	23.97	142.5	188.24	~ 47.94	150	~ 165.12
Encore	8.56	2.32	~ 4.32	9.38	4.64	~ 9.29	18.77	9.28	~ 19.9
d-Encore	1.34			1.36			2.11		

Overhead on (F)HE. We finally quantify the cost of embedding our SNARK atop the base (F)HE scheme.

From Table 2, proof generation consumes ~ 360 bits of noise across 7 limbs, versus ≈ 100 bits for RAA, 150 bits for Laminate, 250 bits for Blind ZK, and 360 bits for Phalanx. Our noise budget stems from the (I)NTT needed to derive the quotient polynomial $H(s)$ and our larger plaintext modulus ($\log p = 23$ versus 17 for Laminate), which raises per-level multiplication noise. For multiplicative depth $\text{dp} \in \{6, 8, 10\}$, our overhead on the base (F)HE is $4.32\times, 1.78\times, 1.70\times$, against $2.86\times, 1.33\times, 1.27\times$ for Laminate.

Crucially, proof-generation runtime dominates the end-to-end cost. We compare with our strongest baseline Laminate as follows. For a circuit of size $N = 2^{20}$ with $B = D = 2^{14}$, $\text{dp} = 8$, half multiplications, half additions gates distributed uniformly, an honest evaluation takes 0.93 hours [53, Table 23]. Integrated with Laminate, the total server time is ~ 12.36 hours, while with Encore it is $0.93 \cdot 1.78 + 9.38 = 11.04$ hours, $1.12\times$ faster. Since our protocol is agnostic to circuit structure and computation-free for addition gates, the gap widens on addition-heavy circuits: If the circuit contains 90% addition gates ($< 2^{17}$ multiplications, ~ 0.21 honest hours), Laminate needs ~ 3.55 hours while Encore takes $0.21 \cdot 1.78 + 1.70 = 2.07$ hours, a $1.71\times$ improvement. Conversely, for a circuit with 90% multiplication gates, Laminate takes ~ 9.76 hours, and we require $1.65 \cdot 1.78 + 18.00 = 20.94$ hours, $2.15\times$ slower. Meanwhile, our proof size and verification time retain their advantage across all settings.

The distributed variant, d-Encore, trades a larger noise budget for smaller circuit sizes (thus cheaper NTT), requiring up to 560 bits across 11 limbs. This leaves roughly 4 multiplication levels to the base (F)HE (without bootstrapping), and incurs a $4\times$ overhead. With $M = 128$ sub-provers, the total server time of d-Encore is $0.93/M \cdot 4 + 1.36 = 1.39$ hours for half-half circuits ($8.89\times$ faster than Laminate), 0.14 hours at 90% additions ($25.36\times$ faster, since now the effective sub-circuit size for sub-provers is $< 2^{10}$), and 2.34 hours at 90% multiplications (still $4.17\times$ faster than Laminate).

When our protocol shines. Since additions are proven for free, our protocol is best suited to circuits that are not multiplication-dominated. The distributed variant further favors *wide and shallow* circuits, with a huge amount of

multiplication gates yet shallow multiplicative depths. A natural fit is verifiable private analytics over large encrypted datasets, e.g., computing statistics or weighted aggregates over millions of records.

7. Conclusion and Open Problems

In this work we present Encore and its distributed variant d-Encore, an FHE-over-SNARK that achieves constant proof size independent of the circuit size, while natively supporting SIMD FHE operations. By revisiting the LPCP/QAP framework long set aside as impractical for verifiable FHE and lifting the proven relation to the plaintext level via the ciphertext-encoding multiplication, Encore inherits the same security foundation as prior LPCP-based SNARKs without introducing new assumptions. On circuits with multiplication and addition gates evenly distributed, our protocol outperforms prior work in a preprocessing model, and the advantage widens on addition-heavy, wide-and-shallow circuits.

As a new route to FHE-over-SNARK, our work leaves several problems open. First, the prover cost is dominated by the (I)NTT used to derive the quotient polynomial; whether this can be further optimized or replaced by a fundamentally cheaper representation remains an interesting question. Second, our SNARK proves arithmetic relations over the plaintext, and thus cannot efficiently certify non-arithmetic operations such as Galois automorphisms. Similarly, our protocol works in the exact relation and it seems that to support approximate FHE schemes such as CKKS [15] requires new techniques. Third, Encore operates in a preprocessing model: the verifier performs a one-time, circuit-dependent setup to generate the query encodings, which, as in prior LPCP-based work, amortizes across repeated evaluations of the same circuit but is a real cost for one-shot computations. Finally, supporting server-side input privacy requires additional engineering. It would also be interesting to adapt the noise-smudging technique of LUNA [56] to avoid the exponential noise involved for smudging alone.

Acknowledgments

This work was supported in part by the Ethereum Foundation (Grant FY25-2130).

References

- [1] Mohammad Sadegh Ahmadi, Taraneh Eghlidos, Behzad Abdolmaleki, and Ngoc Khanh Nguyen. A lattice-based designated verifier zkSNARK from standard assumptions. *Cryptology ePrint Archive*, 2025.
- [2] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Cryptology ePrint Archive*, 2015.
- [3] Diego F Aranha, Anamaria Costache, Antonio Guimarães, and Eduardo Soria-Vazquez. Heliopolis: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In *ASIACRYPT 2024*, 2024.
- [4] Shahla Atapoor, Karim Bagheri, Hilder VL Pereira, and Jannik Spiessens. Verifiable fhe via lattice-based snarks. *IACR Communications in Cryptology*, 2024.
- [5] Ward Beullens and Gregor Seiler. Labrador: compact proofs for r1cs from module-sis. In *CRYPTO 2023*, 2023.
- [6] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. *Cryptology ePrint Archive*, Paper 2012/718, 2012.
- [7] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. *Cryptology ePrint Archive*, Paper 2020/1526, 2020.
- [8] Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. Single secret leader election. In *ACM AFT 2020*, 2020.
- [9] Dan Boneh, Yuval Ishai, Amit Sahai, and David J Wu. Lattice-based snarks and their application to more efficient obfuscation. In *EUROCRYPT 2017*, 2017.
- [10] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO 2012*, 2012.
- [11] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in lwe-based homomorphic encryption. In *PKC 2013*, 2013.
- [12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS 2012*, 2012.
- [13] Ignacio Cascudo, Anamaria Costache, Daniele Cozzo, Dario Fiore, Antonio Guimarães, and Eduardo Soria-Vazquez. Verifiable computation for approximate homomorphic encryption schemes. *Cryptology ePrint Archive*, 2025.
- [14] Sylvain Chatel, Christian Knabenhans, Apostolos Pyrgelis, Carmela Troncoso, and Jean-Pierre Hubaux. Verifiable encodings for secure homomorphic analytics. *arXiv:2207.14071*, 2022.
- [15] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT 2017*, 2017.
- [16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT 2016*, 2016.
- [17] Thomas Debris-Alazard, Pouria Fallahpour, and Damien Stehlé. Quantum oblivious lwe sampling and insecurity of standard model lattice-based snarks. In *STOC 2024*, 2024.
- [18] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In *EUROCRYPT 2015*, 2015.
- [19] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, Paper 2012/144, 2012.
- [20] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In *ACM CCS 2014*, 2014.
- [21] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In *PKC 2020*, 2020.
- [22] Ben Fisch, Arthur Lazzaretti, Zeyu Liu, and Charalampos Papamanthou. Thorpir: Single server pir via homomorphic thorp shuffles. In *ACM CCS 2024*, 2024.
- [23] Ben Fisch, Zeyu Liu, Eran Tromer, and Yunhao Wang. UnifOMR: Oblivious message retrieval with near-optimal concrete efficiency. *Cryptology ePrint Archive*, Paper 2026/910, 2026.
- [24] Mariana Gama, Emad Heydari Beni, Jiayi Kang, Jannik Spiessens, and Frederik Vercauteren. Blind zkSNARKs: for private proof delegation and verifiable computation over encrypted data. *IACR Communications in Cryptology*, 2025.
- [25] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: Snarks for ring arithmetic. *Journal of Cryptology*, 2023.
- [26] Sanjam Garg, Aarushi Goel, and Mingyuan Wang. How to prove statements obliviously? In *CRYPTO 2024*, 2024.
- [27] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO 2010*, 2010.
- [28] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT 2013*, 2013.
- [29] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. Stanford University, 2009.
- [30] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved RNS variant of the BFV homomorphic encryption scheme. In *CT-RSA 2019*, 2019.
- [31] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nikolai Zeldovich. Private web search with tiptoe. In *SOSP 2023*, 2023.
- [32] Alexandra Henzinger, Matthew M Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast Single-Server private information retrieval. In *USENIX Security 2023*, 2023.
- [33] Kexi Huang, Yanpei Guo, Wenjie Qu, and Jiaheng Zhang. Linear-time, constant-depth blind polynomial commitments from generalized RAA codes, with an end-to-end blind SNARK implementation. *Cryptology ePrint Archive*, Paper 2026/487, 2026.
- [34] Janik Huth, Antoine Joux, and Giacomo Santato. Fherret: Proof of the correct-and-honest evaluation with circuit privacy from mpcith. *Cryptology ePrint Archive*, 2025.
- [35] Yuval Ishai, Hang Su, and David J Wu. Shorter and faster post-quantum designated-verifier zkSNARKs from lattices. In *ACM CCS 2021*, 2021.
- [36] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakanan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security 2018*, 2018.
- [37] Joon-Woo Lee, Hyungchul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, and Jong-Seon No. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 2022.
- [38] Keewoo Lee and Yongdong Yeo. SophOMR: Improved oblivious message retrieval from SIMD-aware homomorphic compression. *Cryptology ePrint Archive*, Paper 2024/1814, 2024.
- [39] Chongrong Li, Pengfei Zhu, Yun Li, Cheng Hong, Wenjie Qu, and Jiaheng Zhang. Hyperpianist: Pianist with linear-time prover and logarithmic communication cost. In *IEEE S&P 2025*, 2025.
- [40] Shimin Li, Xin Wang, and Rui Zhang. Privacy-preserving homomorphic macs with efficient verification. In *ICWS 2018*, 2018.
- [41] Haofei Liang, Zeyu Liu, Yunhao Wang, Xiang Xie, Yu Yu, and Fan Zhang. Relect: Single secret leader election via FHE with reduced computation and communication and transparent setup. *Cryptology ePrint Archive*, Paper 2026/1619, 2026.

- [42] Fengrun Liu, Haofei Liang, Tianyu Zhang, Yuncong Hu, Xiang Xie, Haisheng Tan, and Yu Yu. Hasteboots: Proving the bootstrapping in seconds. *Cryptology ePrint Archive*, 2025.
- [43] Tianyi Liu, Tiancheng Xie, Jiaheng Zhang, Dawn Song, and Yupeng Zhang. Pianist: Scalable zkrollups via fully distributed zero-knowledge proofs. In *IEEE S&P 2024*, 2024.
- [44] Zeyu Liu, Katerina Sotiraki, Eran Tromer, and Yunhao Wang. Snake-eye resistant PKE from LWE for oblivious message retrieval and robust encryption. In *EUROCRYPT 2025*, 2025.
- [45] Zeyu Liu and Eran Tromer. Oblivious message retrieval. In *CRYPTO 2022*, 2022.
- [46] Zeyu Liu, Eran Tromer, and Yunhao Wang. Group oblivious message retrieval. In *IEEE S&P 2024*, 2024.
- [47] Zeyu Liu, Eran Tromer, and Yunhao Wang. PerfOMR: Oblivious message retrieval with reduced communication and computation. *USENIX Security 2024*, 2024.
- [48] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC 2012*, 2012.
- [49] Microsoft SEAL (release 3.6). <https://github.com/Microsoft/SEAL>, 2020.
- [50] Deepika Natarajan, Andrew Loveless, Wei Dai, and Ronald Dreslinski. Chex-mix: Combining homomorphic encryption with trusted execution environments for two-party oblivious inference in the cloud. *Cryptology ePrint Archive*, 2021.
- [51] Ngoc Khanh Nguyen and Gregor Seiler. Greyhound: Fast polynomial commitments from lattices. In *CRYPTO 2024*, 2024.
- [52] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. *Communications of the ACM*, 2016.
- [53] Kabir Peshawaria, Zeyu Liu, Ben Fisch, and Eran Tromer. Laminate: Succinct simd-friendly verifiable fhe. *Cryptology ePrint Archive*, 2025.
- [54] Michael Rosenberg, Tushar Mopuri, Hossein Hafezi, Ian Miers, and Pratyush Mishra. Hekaton: Horizontally-scalable zksnarks via proof aggregation. *Cryptology ePrint Archive*, 2024.
- [55] Muhammad Husni Santrijaji, Jiaqi Xue, Yancheng Zhang, Qian Lou, and Yan Solihin. Dataset: Ensuring the verifiability of private computation on encrypted data. In *IEEE S&P 2025*, 2025.
- [56] Ron Steinfeld, Amin Sakzad, Muhammed F Esgin, Veronika Kuchta, Mert Yassi, and Raymond K Zhao. Luna: quasi-optimally succinct designated-verifier zero-knowledge arguments from lattices. In *ACM CCS 2024*, 2024.
- [57] Louis Tremblay Thibault and Michael Walter. Towards verifiable fhe in practice: Proving correct execution of tfhe’s bootstrapping using plonky2. *Cryptology ePrint Archive*, 2024.
- [58] Louis Tremblay Thibault, Michael Walter, and Jiapeng Zhang. Practical snarks for matrix multiplications over encrypted data. *Cryptology ePrint Archive*, 2026.
- [59] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption. *arXiv:2301.07041*, 2023.
- [60] Wenhao Wang, Fangyan Shi, Dani Vilardell, and Fan Zhang. Cirrus: Performant and accountable distributed snark. *Cryptology ePrint Archive*, Paper 2024/1873, 2024.
- [61] Yunhao Wang. Encore repo, 2026. <https://github.com/wyuhao/mk-fhe-zkp>.
- [62] Yunhao Wang and Fan Zhang. Qelett: Lattice-based single secret leader election made practical. *USENIX Security 2025*, 2025.
- [63] Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. DIZK: A distributed zero knowledge proof system. In *USENIX Security 2018*, 2018.
- [64] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. zkBridge: Trustless cross-chain bridges made practical. In *ACM CCS 2022*, 2022.
- [65] Hua Xu, Mariana Gama, Emad Heydari Beni, and Jiayi Kang. Frittata: Distributed proof generation of fri-based snarks. *Cryptology ePrint Archive*, 2025.
- [66] Xinxuan Zhang, Ruida Wang, Zeyu Liu, Binwu Xiang, Yi Deng, and Xianhui Lu. Fhe-snark vs. snark-fhe: From analysis to practical verifiable computation. *Cryptology ePrint Archive*, 2025.
- [67] Xinxuan Zhang, Ruida Wang, Qingyun Niu, Peixin Liu, Xianhui Lu, Lutan Zhao, Rui Hou, and Yi Deng. Lasagne: Practical verifiable computation over encrypted data. *Cryptology ePrint Archive*, 2026.
- [68] Zhelei Zhou, Yun Li, Yuchen Wang, Zhaomin Yang, Bingsheng Zhang, Cheng Hong, Tao Wei, and Wenguang Chen. Zhe: Efficient zero-knowledge proofs for he evaluations. In *IEEE S&P 2025*, 2025.

Appendix A.

Supplementary Materials of Preliminary

(Fully) homomorphic encryption. A (fully) homomorphic encryption scheme, first constructed by [29], consists of algorithms (GenParam, KeyGen, E, D, Eval). The GenParam interface takes as input the security parameter and outputs the (F)HE parameters pp , which include the ring dimension D , the ciphertext modulus q , the plaintext modulus t , the secret key distribution $\mathcal{D}$ and the error distribution χ . The parameters pp are input to all other algorithms. The KeyGen algorithm generates the public key, the secret key, and the evaluation key (pk, sk, ek_{eval}) . The E interface takes a public key pk and a plaintext message $m \in \mathcal{R}_t$ as inputs, and outputs a ciphertext $ct := \overline{m} \in \mathcal{R}_q$. Decryption recovers the plaintext accordingly: $m \leftarrow D(ct, sk)$. Given an arithmetic function f and ciphertexts $(ct_1, \dots, ct_n) := (\overline{m_1}, \dots, \overline{m_n})$, $Eval(f, (ct_1, \dots, ct_n), ek_{eval})$ outputs ct' , s.t., $D(ct') = f(m_1, \dots, m_n)$. To be more specific:

- $ct \leftarrow Eval(+, \{ct_i\}_{i \in [m]})$: given a list of ciphertexts, outputs a single ciphertext ct , s.t., $D(ct) = \sum_{i \in [m]} D(ct_i)$.
- $ct \leftarrow Eval(\cdot, ct_1, ct_2)$: given two input ciphertexts, outputs a single ciphertext ct , s.t., $\forall i \in [N], D(ct)[i] = D(ct_1)[i] \cdot D(ct_2)[i]$.

In BFV, an input message is treated as a vector in $\mathbb{Z}_t^D$, which is then encoded into a ring element $\in \mathcal{R}_t$ via the Inverse Number Theoretic Transform (INTT) before encryption. Similarly, decryption produces a ring element that is decoded back into a vector in $\mathbb{Z}_t^D$. For simplicity and readability, we treat the encoding and decoding procedures as being wrapped inside the E and D interfaces. Moreover, BFV supports Single Instruction Multiple Data (SIMD), enabling homomorphic operations to be applied component-wise to all entries (slots) of the encrypted vector.

Definition A.1 (Distributed VCoED). A distributed verifiable computation on encrypted data (d-VCoED) protocol Π consists of the PPT algorithms (Setup, E, Distribute, SubProve, AggProve, Verify), with Setup, E, Verify defined in the same way as in Def. 2.3 and other procedures defined as follows.

- $(\overline{\text{PK}}, \{(\text{PK}_i, \mathbb{x}_i, \mathbb{w}_i)\}_{i \in [M]}) \leftarrow \Pi.\text{Distribute}(\text{PK}, \mathbb{x}, \mathbb{w})$: create M sub-circuits $\{(\text{PK}_i, \mathbb{x}_i, \mathbb{w}_i)\}_{i \in [M]}$ and a master proving key $\overline{\text{PK}}$ based on $(\text{PK}, \mathbb{x}, \mathbb{w})$.
- $(\pi_i, \text{ct}'_i) \leftarrow \text{SubProve}(\text{PK}_i, \mathbb{x}_i, \mathbb{w}_i)$: generate proof π_i and the evaluation result ct'_i for sub-circuit $(\mathbb{x}_i, \mathbb{w}_i)$ using the proving key PK_i .
- $(\pi, \text{ct}') \leftarrow \text{AggProve}(\overline{\text{PK}}, \{(\pi_i, \text{ct}'_i)\}_{i \in [M]})$: generate aggregated proof π and the final evaluation given the master proving key $\overline{\text{PK}}$, the sub-proofs and the evaluations of the sub-circuits $\{(\pi_i, \text{ct}'_i)\}_{i \in [M]}$.

With *outsourcability* and *client-privacy* as defined in Def. 2.3, a d-VCoED for R_{SAT} also needs to satisfy the following properties:

- **Completeness.** For all $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \text{R}_{\text{SAT}}$, let $(\text{VK}, \text{PK}) \leftarrow \text{Setup}(1^\lambda, \mathbb{i})$, $(\overline{\text{PK}}, \{(\text{PK}_i, \mathbb{x}_i, \mathbb{w}_i)\}_{i \in [M]}) \leftarrow \text{Distribute}(\text{PK}, \mathbb{x}, \mathbb{w})$, $(\pi_i, \text{ct}'_i) \leftarrow \text{SubProve}(\text{PK}_i, \mathbb{x}_i, \mathbb{w}_i)$, $i \in [M]$ and $(\pi, \text{ct}') \leftarrow \text{AggProve}(\overline{\text{PK}}, \{(\pi_i, \text{ct}'_i)\}_{i \in [M]})$, we have that $\text{Verify}(\text{VK}, \mathbb{x}, \text{ct}', \pi) = 1$.
- **Knowledge soundness.** For any PPT adversary $\mathcal{A}$ there is a PPT extractor $\mathcal{E}$ such that if $(\mathbb{i}', \text{aux}) \leftarrow \mathcal{A}(1^\lambda)$, $(\text{VK}, \text{PK}) \leftarrow \text{Setup}(1^\lambda, \mathbb{i}')$, and $(\mathbb{x}', \mathbb{w}', \pi', \text{ct}') \leftarrow \mathcal{E}^\mathcal{A}(\text{PK}, \text{aux})$, where the extractor might have access to the random tape of $\mathcal{A}$, it holds that: $\Pr \left(\begin{array}{c} \text{Verify}(\text{VK}, \mathbb{x}', \text{ct}', \pi') = 1 \\ \wedge (\mathbb{i}', \mathbb{x}', \mathbb{w}') \notin \text{R}_{\text{SAT}} \end{array} \right) = \text{negl}(\lambda)$.
- **Succinctness.** For any $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \text{R}_{\text{SAT}}$, we have that $|\pi| = \text{poly}(\lambda) \cdot \text{polylog}(|\mathbb{x}| + |\mathbb{w}|)$, which is independent of M .

Appendix B.

LPCP-based FHE-over-SNARK

In this section, we formally present our FHE-over-SNARK protocol *Encore* in Fig. 7 based on LPCP, for the indexed relation $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \text{R}_{\text{SAT}}$ for circuit satisfiability.

We first construct the underlying Π_{LPCP} in Fig. 6, which is a standard 4-query LPCP based on QAP as in [35]. The completeness and knowledge soundness follow directly from [35]. With the Π_{LPCP} in hand, we construct our FHE-over-SNARK protocol *Encore* in Fig. 7.

Remark B.1. Standard strictly LTM (Assumption 2.1) simulates the linear proof over the extension field $\mathcal{R}_{p^d}$ whereas the (F)HE witness should live in $\mathcal{R}_p$. For client-only inputs this is immediate: All inputs are $\mathcal{R}_p$ elements and homomorphic evaluation keeps every intermediary wire in $\mathcal{R}_p$. For circuits with server-side inputs $\{\nu_i\}$ potentially in $\mathcal{R}_{p^d}$, we enhance the original (F)HE to additionally compute $\sum_{i \in [K']} [r_i]_1 \cdot [\nu_i]_1$, where K' is the total number of server inputs, and $\{r_i \in \mathcal{R}_p\}$ are secret random elements chosen by the verifier. I.e., the prover computes the random linear combination of all server-side inputs, and a well-formed $\mathcal{R}_{p^d}^2$ output α certifies all $\nu_i \in \mathcal{R}_p$ with soundness error $1/p$, amplified to $1/p^\rho$ over ρ repetitions. With $\log p = 23$, $\rho = 6$ suffices for 128-bit soundness, enlarging the proof by

$(\text{st}, \text{Q}) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, \mathbb{i})$:

- 1) Parse $\mathbb{i}$ as $(\text{pp} := (\mathcal{R}, \lambda), C := (\{V_k(x), W_k(x), Y_k(x)\}_{k \in \mathcal{I}}, t(x)))$ (see Def. 2.2), where $\mathcal{I} := \mathcal{I}_{\text{io}} \cup \mathcal{I}_{\text{mid}}$ is the set of all wires, $\mathcal{I}_{\text{io}}$ denotes the wire indices set for public inputs/outputs. Let N be the number of multiplication gates in the circuit.
- 2) Define the $|N|$ -th root of unity $\{\omega^j\}$, with $N \mid p - 1$. Sample a coset $\{\xi \omega^j\}$ for $\xi \xleftarrow{\$} \mathcal{R}_p$. Sample the random evaluation point $s \xleftarrow{\$} \mathcal{R}_{p^d} \setminus \{\omega^j, \xi \omega^j\}_{j \in [N]}$.
- 3) Let $L_j(s) := \frac{(s^N - \xi^N) \cdot \omega^j}{N \cdot \xi^{N-1} (s - \xi \cdot \omega^j)}$, $j \in [N]$ be the Lagrange basis. Define $H_j(s) = t(\xi \omega^j)^{-1} \cdot L_j(s)$.
- 4) Let $w' := |\mathcal{I}_{\text{io}}|$ be the total number of public input/output wires and w be the total number of wires in each sub-circuit C_i . Construct the query matrix $Q \in \mathcal{R}_p^{4 \times (N + |\mathcal{I}_{\text{mid}}| + 1)}$ as:
$$\begin{pmatrix} V_{|\mathcal{I}_{\text{io}}|+1}(s) & \dots & V_{|\mathcal{I}|}(s) & 0 & \dots & 0 \\ W_{|\mathcal{I}_{\text{io}}|+1}(s) & \dots & W_{|\mathcal{I}|}(s) & 0 & \dots & 0 \\ Y_{|\mathcal{I}_{\text{io}}|+1}(s) & \dots & Y_{|\mathcal{I}|}(s) & 0 & \dots & 0 \\ 0 & \dots & 0 & H_1(s) & \dots & H_N(s) \end{pmatrix}. \quad (6)$$
- 5) Output $(\text{st} := s, \text{Q})$.

$\pi \leftarrow \Pi_{\text{LPCP}}.\text{Prove}(1^\lambda, \mathbb{i}, \mathbb{x}, \mathbb{w})$:

- 1) Evaluate C based on the circuit structure $\mathbb{i}$ and the encrypted FHE data of $\mathbb{x}$ and derive all wire values homomorphically, denoted as $\{\text{ct}_k\}_{k \in [|\mathcal{I}_{\text{mid}}|]}$.
- 2) Treat $\{\text{ct}_{v_j}\}, \{\text{ct}_{w_j}\}, \{\text{ct}_{y_j}\} \subset \{\text{ct}_k\}_{k \in [|\mathcal{I}_{\text{mid}}|]}$ as the evaluations of polynomial $V_{\text{mid}}, W_{\text{mid}}, Y_{\text{mid}}$ at the root of unit ω^j for $j \in [N]$. Use a η -layered INTT to homomorphically compute $\{\text{ct}'_{v'_j}\}, \{\text{ct}'_{w'_j}\}, \{\text{ct}'_{y'_j}\}$, denoting ciphertexts encrypting the coefficients of $V_{\text{mid}}, W_{\text{mid}}, Y_{\text{mid}}$. And then use an η -layered NTT to homomorphically compute $\{\text{ct}''_{v'_j}\}, \{\text{ct}''_{w'_j}\}, \{\text{ct}''_{y'_j}\}$, encrypting the evaluations of $V_{\text{mid}}, W_{\text{mid}}, Y_{\text{mid}}$ on the coset points $\{\xi \omega^j\}$.
- 3) Derive $\text{ct}_{h_j} \leftarrow \Pi_E^{(1)}.\text{Eval}(+, -\text{ct}''_{y'_j}, \Pi_E^{(1)}.\text{Eval}(\times, \text{ct}''_{v'_j}, \text{ct}''_{w'_j}))$.
- 4) Output $\pi := \{\text{ct}_k\}_{k \in [|\mathcal{I}_{\text{mid}}|]} \cup \{\text{ct}_{h_j}\}_{j \in [N+1]}$.

$\{0, 1\} \leftarrow \Pi_{\text{LPCP}}.\text{Verify}(\text{st}, \mathbb{x}, \pi)$:

- 1) Parse $\text{st} := (V_{\text{io}}(s), W_{\text{io}}(s), Y_{\text{io}}(s))$, and $\mathbb{x} := \{m_\kappa\}$ as the decryption of the FHE evaluation outputs. Derive $V'_{\text{io}}(s) \leftarrow V_{\text{io}}(s) + \sum_\kappa m_\kappa \cdot V_\kappa(s)$, similar to $W'_{\text{io}}(s), Y'_{\text{io}}(s)$.
- 2) Parse $\text{ans} := \bar{m} \parallel \bar{m}'$, where $\bar{m} \in \mathcal{R}_{p^d}^\ell$ and $\bar{m}' \in \mathcal{R}_{p^d}^\tau$ and return $\perp$ directly if $\bar{m} \cdot T \neq \bar{m}'$.
- 3) Parse $\text{ans} := (v', w', y', h')$, and return 1 if $(V'_{\text{io}}(s) + v')(W'_{\text{io}}(s) + w') - (Y'_{\text{io}}(s) + y') = h' t(s)$; 0 otherwise.

Figure 6: The formal protocol of Π_{LPCP} .

a few hundred KB. We consider the incurred prover side cost negligible for a moderate number of server inputs ($\lesssim 10\%$ of the circuit). Matching prior work [53], [67], we focus concrete evaluations on the client-only setting in this work.

Proof of Theorem 5.1.

Proof. The completeness directly follows from the results in prior works [9, Theorem 4.6], [6, Lemma 6.3]. We prove the soundness by contradiction. Let $\mathcal{A}$ be the malicious prover that convinces the verifier of some false statement $(\mathbb{i}, \mathbb{x}) \notin \mathcal{L}$ with non-negligible probability $\epsilon(\lambda)$. Since $\Pi_E^{(1)}, \Pi_E^{(2)}$ satisfy the multi-key strictly LTM as defined in Def. 4.3, then there exists a simulator $\mathcal{S}$ such that the following two distributions are computationally indistinguishable:

- **Real Distribution:** Invoke $\text{Encore}.\text{Setup}(1^\lambda, \mathbb{i})$ to get $\text{VK} := (\text{sk}_1, \text{sk}_2, \cdot)$ and $\text{PK} := (\text{crs}, \text{pp}, \text{pk}_1, \text{evk}, \mathbb{i})$. Let $(\text{st}, \text{Q}) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, \mathbb{i})$ be the underlying invocation on Π_{LPCP} inside $\text{Encore}.\text{Setup}$, where $Q \in \mathcal{R}_{p^d}^{4 \times (N + |\mathcal{I}_{\text{mid}}| + 1)}$.

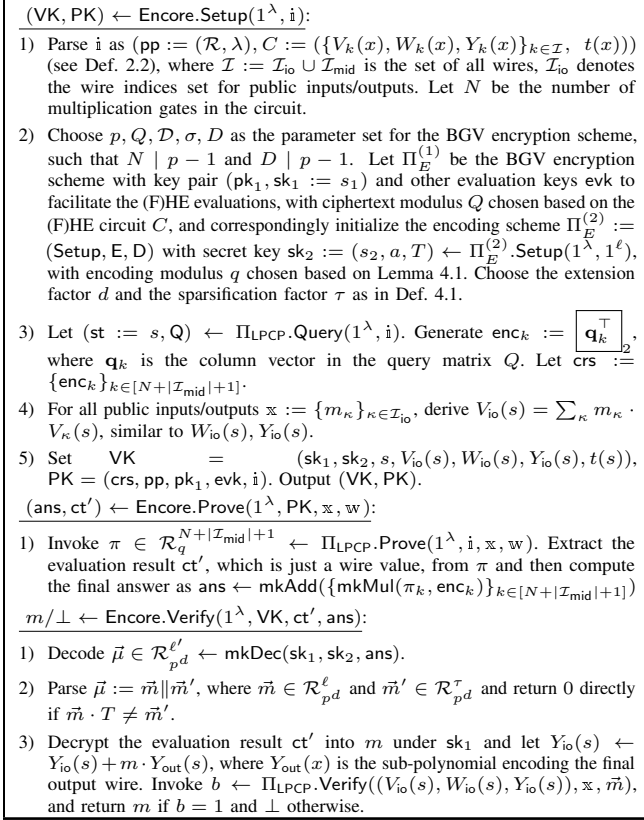


Figure 7: The formal protocol of Encore.

Let $enc_i \leftarrow \Pi_E^{(2)}.E(sk, \vec{q}_i)$, where $\vec{q}_i$ is the i -th column of the query matrix Q .

$enc' \leftarrow \mathcal{A}(pp, \{enc_i\}, pk_1, i, \mathbb{x})$, s.t., $\text{mkDec}(sk_1, sk_2, enc') \neq \perp$.

$\vec{m}'_0 \in \mathcal{R}_{p^d}^5 \leftarrow \text{mkDec}(sk_1, sk_2, enc')$. Output $(Q, st, \vec{m}'_0)$.

- **Ideal Distribution:** $(st, Q) \leftarrow \Pi_{LPCP}.Query(1^\lambda, i)$, where $Q \in \mathcal{R}_{p^d}^{4 \times (N+|\mathcal{I}_{mid}|+1)}$.

$\pi \leftarrow \mathcal{S}(i)$, and $\vec{m}'_1 \in \mathcal{R}_{p^d}^5 \leftarrow \pi^\top Q$. Output $(Q, st, \vec{m}'_1)$

By assumption, $\mathcal{A}$ convinces an honest verifier with probability $\epsilon(\lambda)$, i.e., we have the LPCP verifier output $\Pi_{LPCP}.Verify(Q, st, \vec{m}'_0) = 1$ with probability at least $\epsilon(\lambda)$. Based on the computational indistinguishability of the two distributions, we should then have $\Pi_{LPCP}.Verify(Q, st, \vec{m}'_1) = 1$ with probability at least $\epsilon(\lambda) - \text{negl}(\lambda)$ by treating $\Pi_{LPCP}.Verify$ as our PPT distinguisher between two distributions.

Now, notice that in the *ideal distribution*, the linear proof $\pi \in \mathcal{R}_{p^d}^{N+|\mathcal{I}_{mid}|+1}$ is generated independently of the query matrix Q and the state st of Π_{LPCP} . Therefore, by an averaging argument, there exists some π , s.t., on input $(i, \mathbb{x}) \notin \mathcal{L}$, $\Pi_{LPCP}.Verify(Q, st, \vec{m}'_1) = 1$. On the other hand, based on the statistical soundness of the underlying Π_{LPCP} , for all $\pi \in \mathcal{R}_p^{N+|\mathcal{I}_{mid}|+1}$ (i.e., the non-extended linear proof), we have $\Pi_{LPCP}.Verify(st, \mathbb{x}, w) = 1$ with probability less

than ϵ , where w is the first $|\mathcal{I}_{mid}|$ ring elements of π , while for all $\pi \in \mathcal{R}_{p^d}^{N+|\mathcal{I}_{mid}|+1} \setminus \mathcal{R}_p^{N+|\mathcal{I}_{mid}|+1}$ (i.e., the witness strictly lying in the extended ring), we clearly have $R_{SAT}(i, \mathbb{x}, w) = 0$ under the case of no-server-side inputs (i.e., all encrypted inputs and evaluation outputs $\in \mathbb{x}$ of the FHE scheme are non-extended $\mathcal{R}_p$ elements).

Therefore, we have a contradiction and conclude that Encore is non-adaptively sound. $\square$

Proof of Theorem 5.2.

Proof. The proof follows from [6] and is similar to Theorem 5.1. Essentially, if the malicious prover is able to convince the verifier with some non-negligible probability ϵ , then given the computational indistinguishability, we would have a PPT simulator in the *ideal* distribution and output some $(Q, st, \vec{m}'_1)$. We then invoke the extractor $\mathcal{E}(i, \mathbb{x})$ of the underlying Π_{LPCP} to extract the witness w . Similar to before, under a no-server-input setting (i.e., VCoED), we have the extracted $w \in \mathcal{R}_p^{|\mathcal{I}_{mid}|}$ as the valid witness lying in the non-extended ring. $\square$

Setup cost. Like all prior works, including those [53], [67] based on holographic GKR, our protocol requires a circuit-dependent setup that amortizes across evaluations of the same circuit. This setup includes deriving the sub-polynomials $V_k(s), W_k(s), Y_k(s)$ for $k \in [3N]$ and $H_i(s)$ for $i \in [N]$. By the definition of a QAP (Def. 2.2), the evaluations of $V_k(x)$ on the roots of unity are all zero except for a single slot, which is one; the same holds for $W_k(x)$ and $Y_k(x)$. Therefore, we use $O(N)$ operations to compute the Lagrange basis $L_i(s) = \frac{\omega^i \cdot (s^N - 1)}{N \cdot (s - \omega^i)}$ for $i \in [N]$, which directly yields the evaluations $V_k(s), W_k(s), Y_k(s)$. A similar procedure is taken for deriving $H_i(s)$ in $O(N)$. The dominant cost comes from the encoding step, which, based on the microbenchmark from the prior work [67], takes ~ 84.3 hours on a single core. This cost is a circuit-dependent preprocessing and can be amortized across multiple evaluations. As in all the prior works, we focus the comparison on the online prover cost.

Appendix C.

Aggregation Framework of Distributed Prover

Our distributed protocol d-Encore consists of five phases: (1) a setup phase that generates the verification and proving keys, which also specify the structure of all sub-circuits and the aggregation circuit, (2) a distribution phase to generate sub-witnesses based on the partitioned sub-circuits and distribute the proving keys to the prover side, (3) the parallel computation of sub-provers, (4) the aggregation performed by the master prover, which generates a succinct proof, and (5) the verification checks computed by the verifier.

Definition C.1 (Aggregation Circuit). Let $\Pi_E^{(1)}, \Pi_E^{(2)}$ be as defined in Def. 4.2. For $M \in \mathbb{Z}$, and $\vec{a}_i \in (\mathcal{R}_q^2)^8$ with $i \in [M]$, the aggregation circuit (AC) is described by the QAP $C_{AC} := \{\{V_k(x), W_k(x), Y_k(x)\}_{k \in \mathcal{I}}, \vec{t}(x)\}$, where $\mathcal{I}$ is the index set of all wires in C_{AC} , and evaluates the

functions: $f_{AC}(\{\vec{a}_i\}_{i \in [M]}) = \sum_{i \in [M]} (a_{i,1} + a_{i,2}) \otimes (a_{i,3} + a_{i,4}) - (a_{i,5} + a_{i,6}) - a_{i,7} \otimes a_{i,8}$. We denote the output of f_{AC} by out_1 . We denote by $\bar{\mathcal{I}}_{io}$ the set of indices for wires carrying the values $\{a_{i,1}, a_{i,3}, a_{i,5}, a_{i,8}\}_{i \in [M]}$, $\bar{\mathcal{L}}$ the set of indices for wires carrying the values of $\{a_{i,2}, a_{i,4}, a_{i,6}\}_{i \in [M]}$, and $\bar{\mathcal{J}} = \bar{\mathcal{I}} \setminus (\bar{\mathcal{I}}_{io} \cup \bar{\mathcal{L}})$. We define the index-mapping function $\text{ld}_{x_v} : [M] \rightarrow \bar{\mathcal{L}}$, which, with $i \in [M]$, outputs the wire index in AC that carries $a_{i,2}$. Similar to ld_{x_w} , ld_{x_y} , ld_{x_h} .

Lemma C.1 (Circuit Partition [54]). For the indexed relation $(\mathbf{i} = (\text{pp}, C), \mathbb{X}, \mathbb{W}) \in \text{RSAT}$, for some public parameter pp and a circuit C , there exists an efficient transformation $f_{\text{part}} = (f_i, f_x, f_w, f_{\text{map}})$: $(\{\mathbf{i}_i := (\text{pp}, C_i)\}_{i \in [M]}, \mathbf{i}_{AC}) \leftarrow f_i(\mathbf{i}, m)$, $\{\mathbb{X}_i\}_{i \in [M]} \leftarrow f_x(\mathbf{i}, \{\mathbf{i}_i\}_{i \in [M]}, \mathbb{X})$, $\{\mathbb{W}_i\}_{i \in [M]} \leftarrow f_w(\mathbf{i}, \{\mathbf{i}_i\}_{i \in [M]}, \mathbb{W})$, and $k' \leftarrow f_{\text{map}}(\mathbf{i}, \{\mathbf{i}_i\}_{i \in [M]}, i, k)$ maps the wire index $k \in \mathcal{I}_i$ in sub-circuit C_i to a wire index $k' \in \mathcal{I}$ in the original circuit C , such that:

- **Completeness:** If $(\mathbf{i}, \mathbb{X}, \mathbb{W}) \in \text{RSAT}$, then for all $i \in [M]$, $(\mathbf{i}_i, \mathbb{X}_i, \mathbb{W}_i) \in \text{RSAT}$.
- **Knowledge soundness:** Let $(\{\mathbf{i}_i\}_{i \in [M]}, \mathbf{i}_{AC}) \leftarrow f_i(\mathbf{i}, m)$, for every PPT algorithm $\mathcal{A}$, $\{\{\mathbb{X}_i, \mathbb{W}_i\}\}_{i \in [M]} \leftarrow \mathcal{A}(\{\mathbf{i}_i\}_{i \in [M]}, \mathbf{i}_{AC}, m)$, if for all $i, i' \in [M]$, $k \in \mathcal{I}_i$, $k' \in \mathcal{I}_{i'}$ s.t. $f_{\text{map}}(\mathbf{i}, \{\mathbf{i}_i\}_{i \in [M]}, i, k) = f_{\text{map}}(\mathbf{i}, \{\mathbf{i}_i\}_{i \in [M]}, i', k')$, we have $c_k = c_{k'}$, i.e., the k -th wire in C_i carries the same value as the k' -th wire in $C_{i'}$ if they are stemmed from the same wire in the original circuit C , then there exists a PPT extractor $\mathcal{E}$, $(\mathbb{X}, \mathbb{W}) \leftarrow \mathcal{E}(\mathbf{i}, \{\{\mathbf{i}_i, \mathbb{X}_i, \mathbb{W}_i\}\}_{i \in [M]})$, such that: $\Pr \left[\begin{array}{l} \forall i, (\mathbf{i}_i, \mathbb{X}_i, \mathbb{W}_i) \in \text{RSAT} \\ \wedge (\mathbf{i}, \mathbb{X}, \mathbb{W}) \notin \text{RSAT} \end{array} \right] = \text{negl}(\lambda)$.

Lemma C.2 (Batched Satisfiability). For $\delta_i \xleftarrow{\$} \mathcal{R}_{p^d}$, let $F(x) = \sum_{i \in [M]} \delta_i \cdot f_i(x)$ and $f_i(x) = V_i(x)W_i(x) - Y_i(x) - H_i(x)t_i(x)$. For $s \xleftarrow{\$} \mathcal{R}_{p^d}$, if $\forall i \in [M]$, $f_i(s) = 0$, then we have that $F(s) = 0$. If $F(s) = 0$ and $f_i(x)$ has degree at most N' for all $i \in [M]$, then $\Pr[\forall i, f_i(s) = 0] \geq 1 - \frac{2N'}{|p^d|}$.

Proof. It is clear to see that if $\forall i \in [M]$, $f_i(s) = 0$, then $F(s) = 0$. The other direction is straightforward by applying the generalized Schwartz-Zippel lemma [25, Lemma 2]. $\square$

Proof of Theorem 5.3.

Proof. Similar to Theorem 5.2, the completeness, outsourceability and client privacy are straightforward. In the end, the aggregator sends a single ans w.r.t. the query of the aggregation circuit, and thus the succinctness is also preserved. Notice that the evaluation output of AC is expected to be zero and thus the aggregator does not need to send that. Thus, we have the proof size of d-Encore to be the same as our non-distributed protocol Encore. The knowledge soundness follows a similar idea as in Theorem 5.2. Essentially, based on Lemma 4.2, a PPT simulator simulates a linear proof used to perform the inner product with the entangled query matrix $\bar{Q}'$ as defined in Fig. 8. And then based on the knowledge soundness of Π_{LPCP} , the extractor $\mathcal{E}^{\text{OP}}(\mathbf{i}_{AC}, \mathbb{X})$ is able to extract all sub-witnesses $\{\mathbb{W}_i\}_{i \in [M]}$ from the linear proof simulated. Based on Lemma C.2, given the evaluation result

$(\text{VK}, \text{PK}) \leftarrow \text{d-Encore.Setup}(1^\lambda, \mathbf{i})$:

- 1) Parse $\mathbf{i}$ as $(\text{pp} := (\mathcal{R}, \lambda), C := (\{V_k(x), W_k(x), Y_k(x)\}_{k \in \mathcal{I}}, t(x)))$ as in Fig. 7. And let $\{\mathbf{i}_i := (\text{pp}, C_i = (\{V_{i,k}, W_{i,k}, Y_{i,k}\}_{k \in \mathcal{I}_i}, t_i(x)))\}_{i \in [M]}$, $\mathbf{i}_{AC} := (\text{pp}, C_{AC} = (\{\bar{V}_k, \bar{W}_k, \bar{Y}_k\}_{k \in \bar{\mathcal{I}}}, \bar{t}(x))) \leftarrow f_i(\mathbf{i}, m)$. Let N' be the maximum number of multiplication gates in the sub-circuits and let $\bar{N}$ be the number of multiplication gates in AC.
- 2) Instantiate $\Pi_E^{(1)}, \Pi_E^{(2)}$ with parameters $p, Q, \mathcal{D}, \sigma, D, d, \ell, \tau$ as in Fig. 7.
- 3) Let $(\text{st} := s, Q_i) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, \mathbf{i}_i)$ and $(\text{st} := \bar{s}, \bar{Q}) \leftarrow \Pi_{\text{LPCP}}.\text{Query}(1^\lambda, \mathbf{i}_{AC})$. Let $w'_i := |\mathcal{I}_{io,i}|$ and w_i be the total number of wires in each sub-circuit C_i . Similarly, let $\bar{w}$ be the total number of wires in the aggregation circuit, and $\bar{w}' := |\bar{\mathcal{I}}_{io} \cup \bar{\mathcal{L}}|$. Based on the two query matrices $Q, \bar{Q}$, form the entangled query $\bar{Q}'$ as:
$$\begin{pmatrix} \bar{\Psi}^v & \bar{\mathcal{H}}^v & \bar{V}_{\bar{w}'}(\bar{s}) & \cdots & \bar{V}_{\bar{w}}(\bar{s}) & 0 & \cdots & 0 \\ \bar{\Psi}^w & \bar{\mathcal{H}}^w & \bar{W}_{\bar{w}'}(\bar{s}) & \cdots & \bar{W}_{\bar{w}}(\bar{s}) & 0 & \cdots & 0 \\ \bar{\Psi}^y & \bar{\mathcal{H}}^y & \bar{Y}_{\bar{w}'}(\bar{s}) & \cdots & \bar{Y}_{\bar{w}}(\bar{s}) & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \bar{H}_1(\bar{s}) & \cdots & \bar{H}_{\bar{N}}(\bar{s}) \end{pmatrix}, \quad (7)$$
where $\bar{\Psi}^v := (\Psi_{1,w'_1}^v \cdots \Psi_{1,w'_M}^v \cdots \Psi_{M,w'_M}^v \cdots \Psi_{M,w_M}^v)$, s.t. each element $\Psi_{i,k}^v := V_{i,k}(s) \cdot \bar{V}_{\text{ld}_{x_v}(i)}(\bar{s}) + W_{i,k}(s) \cdot \bar{V}_{\text{ld}_{x_w}(i)}(\bar{s}) + Y_{i,k}(s) \cdot \bar{V}_{\text{ld}_{x_y}(i)}(\bar{s})$, and $\bar{\mathcal{H}}^v := \bar{\mathcal{H}}_1^v(\bar{s}) \parallel \cdots \parallel \bar{\mathcal{H}}_M^v(\bar{s})$, s.t. $\bar{\mathcal{H}}_i^v(\bar{s}) := (H_{i,1}(s) \cdot \bar{V}_{\text{ld}_{x_h}(i)}(\bar{s}) \cdots H_{i,N_i}(s) \cdot \bar{V}_{\text{ld}_{x_h}(i)}(\bar{s}))$. Similar to $\bar{\Psi}^w, \bar{\Psi}^y, \bar{\mathcal{H}}^w, \bar{\mathcal{H}}^y$.

Generate $\text{ct}_{i,k} := \begin{bmatrix} \mathbf{q}_{i,k}^\top \end{bmatrix}_1$, where $\mathbf{q}_{i,k}$ is the column vector in the query matrix Q_i . And then generate $\text{enc}_{\bar{k}} := \begin{bmatrix} \bar{\mathbf{q}}_{\bar{k}}^\top \end{bmatrix}_2$, where $\bar{\mathbf{q}}_{\bar{k}}$ is the column vector in the query matrix $\bar{Q}'$. Let $\text{crs} := \{\text{ct}_{i,k}\} \cup \{\text{enc}_{\bar{k}}\}$.

- 4) Construct $\bar{V}_{io}(\bar{s}), \bar{W}_{io}(\bar{s}), \bar{Y}_{io}(\bar{s})$ with $\mathbb{X}$ as in Fig. 7.
- 5) Set $\text{VK} = (\text{sk}_1, \text{sk}_2, s, \bar{s}, \bar{V}_{io}(\bar{s}), \bar{W}_{io}(\bar{s}), \bar{Y}_{io}(\bar{s}), \bar{t}(\bar{s}))$, $\text{PK} = (\text{crs}, \text{pp}, \text{pk}_1, \text{evk}, \{\mathbf{i}_i\}, \mathbf{i}_{AC})$. Output (VK, PK) .

$(\bar{\text{PK}}, \{(\text{PK}_i, \mathbb{X}_i, \mathbb{W}_i)\}_{i \in [M]}) \leftarrow \text{d-Encore.Distribute}(\text{PK}, \mathbb{X}, \mathbb{W}, \text{st})$:

- 1) Parse $\text{PK} \rightarrow (\text{crs}, \text{pp}, \text{pk}_1, \{\mathbf{i}_i\}_{i \in [M]}, \mathbf{i}_{AC})$. Compute $\{\mathbb{X}_i\}_{i \in [M]} \leftarrow f_x(\{\mathbf{i}_i\}_{i \in [M]}, \mathbb{X})$ and $\{\mathbb{W}_i\}_{i \in [M]} \leftarrow f_w(\{\mathbf{i}_i\}_{i \in [M]}, \mathbb{W})$.
- 2) Split the public proving keys into: $\text{PK} := \{\text{enc}_{\bar{k}}\}$, and $\text{PK}_i := \{\text{ct}_{i,k}\} \cup \{\text{enc}_{i,k}'\}$ where $\text{enc}_{\bar{k}}$ denotes the encodings of the last $\bar{w} - \bar{w}' + \bar{N}$ columns in $\bar{Q}'$ matrix, and $\text{enc}_{i,k}'$ denotes the entangled encodings w.r.t. sub-circuit i .
- 3) Output $(\bar{\text{PK}}, \{(\text{PK}_i, \mathbb{X}_i, \mathbb{W}_i)\}_{i \in [M]})$.

$(\text{ans}_i, \text{ct}'_i) \leftarrow \text{d-Encore.SubProve}(1^\lambda, \text{PK}_i, \mathbb{X}_i, \mathbb{W}_i)$:

- 1) Invoke $\pi_i \leftarrow \Pi_{\text{LPCP}}.\text{Prove}(1^\lambda, \mathbf{i}_i, \mathbb{X}_i)$. Extract ct'_i from π_i as the evaluation result and compute the answer as $\text{ans}'_i \leftarrow \text{mkAdd}(\{\text{mkMul}(\pi_{i,k}, \text{ct}_{i,k})\})$, with $\{\text{ct}_{i,k}\}$ parsed from PK_i . Moreover, compute the inner product w.r.t. the entangled encodings $\{\text{enc}_{i,k}'\}$ and denote the result as ans''_i . Let $\text{ans}_i := (\text{ans}'_i, \text{ans}''_i)$.

$(\text{ans}, \text{ct}') \leftarrow \text{d-Encore.AggProve}(1^\lambda, \bar{\text{PK}}, \{\text{ans}_i, \text{ct}'_i\})$:

- 1) Invoke $\pi \leftarrow \Pi_{\text{LPCP}}.\text{Prove}(1^\lambda, \mathbf{i}_{AC}, \mathbb{X}, \{\text{ans}'_i, \text{ct}'_i\})$. Extract ct' as the evaluation result of the original circuit, and compute $\text{ans}' \leftarrow \text{mkAdd}(\{\text{mkMul}(\pi_{\bar{k}}, \text{enc}_{\bar{k}})\})$. And then derive the final answer $\text{ans} := \text{ans}' + \sum_{i \in [M]} \text{ans}''_i$.

$m/\perp \leftarrow \text{d-Encore.Verify}(1^\lambda, \text{VK}, \text{ct}', \text{ans})$:

- 1) Decode $\bar{\mu} \in \mathcal{R}_{p^d}^{\ell'} \leftarrow \text{mkDec}(\text{sk}_1, \text{sk}_2, \text{ans})$.
- 2) Parse $\bar{\mu} := \bar{m} \parallel \bar{m}'$, where $\bar{m} \in \mathcal{R}_{p^d}^{\ell'}$ and $\bar{m}' \in \mathcal{R}_{p^d}^{\tau}$ and return 0 directly if $\bar{m} \cdot T \neq \bar{m}'$.
- 3) Decrypt the evaluation result ct' into m under sk_1 and let $\bar{V}_{io}(\bar{s}) \leftarrow \bar{V}_{io}(\bar{s}) + m \cdot \Psi_{i^*,k^*}^V$, where k^* is the wire index in the sub-circuit i^* that carries the final output. Similar to $\bar{W}_{io}(\bar{s}), \bar{Y}_{io}(\bar{s})$. Invoke $b \leftarrow \Pi_{\text{LPCP}}.\text{Verify}((\bar{V}_{io}(\bar{s}), \bar{W}_{io}(\bar{s}), \bar{Y}_{io}(\bar{s})), \mathbb{X}, \bar{m})$, and return m if $b = 1$ and $\perp$ otherwise.

Figure 8: The formal protocol of d-Encore.

of AC is zero, $(\mathbf{i}_i, \mathbb{X}_i, \mathbb{W}_i) \in \text{RSAT}$ for all i with probability $1 - \text{negl}(\lambda)$. And then based on the knowledge soundness of the underlying partition function (Lemma C.1), a PPT $\mathcal{E}'$ outputs $(\mathbb{X}, \mathbb{W})$ for the original circuit C . $\square$

Appendix D. Ethics Considerations

In this paper, we propose Encore, a SNARK protocol designed for ring arithmetic, specifically tailored to verifiable FHE, by following the FHE-over-SNARK paradigm. The stakeholders potentially affected by our work include users who seek efficient proof schemes for verifiable FHE tasks, the researchers working in lattice-based cryptography or zero-knowledge proofs who may gain insights from our framework, and society at large w.r.t. the potential misuse of proof systems.

We have considered the Menlo Report principles: Beneficence, Respect for Persons, Justice, and Respect for Law and Public Interest. Our evaluations are purely theoretical: no human participants or sensitive data were involved. Based on our thorough analysis, our protocol improves efficiency and scalability, benefiting the verifiable FHE and broader ZKP communities, which aligns with the principle of Beneficence. The protocol is designed to be broadly applicable and unbiased, consistent with the principle of Justice.

Potential harms include misuse of the protocol to generate proofs for malicious homomorphic evaluations. However, this risk is mitigated by the soundness of our construction. We note that side-channel attacks are outside the scope of this work; practitioners deploying the protocol must account for such risks.

In summary, we believe that publishing our protocol, Encore, is ethically justifiable. Its benefits for privacy-preserving computation and verifiable FHE outweigh the limited, mitigated risks. No actual experiments involving end-to-end attacks or sensitive data were performed.

Appendix E. Meta-Review

The following meta-review was prepared by the program committee for the 2027 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

E.1. Summary

This paper introduces Encore, a new FHE over SNARK scheme where the underlying SNARK is based on LPCP (QAP). Encore results in constant proof size (while downgrading the LPCP scheme to a variant with designated verifier). The paper discusses various optimizations, including extending the scheme to distributed proof generation for additional performance gains.

E.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field.
- Establishes a New Research Direction.

E.3. Reasons for Acceptance

- 1) Encore provides constant-proof size and depth-independent verification while natively supporting SIMD packing. This has the potential to be a useful tool in future research on verifiable encrypted computation.
- 2) The core technique of aligning the witness encoding with the QAP SRS polynomials to achieve an FHE-over-SNARK scheme is quite interesting.

E.4. Noteworthy Concerns

- 1) The evaluation largely relies on estimation and would benefit from more actual experiments.

Appendix F. Response to the Meta-Review

We provide a reproducible derivation of every reported runtime in the full version, including a table of measured per-operation costs for the underlying primitives and a table of exact operation counts for each phase of our protocol.