

Multi-Key FHE Almost as Fast as Single-Key FHE

Abtin Afshar
UW-Madison*

Rishab Goyal
UW-Madison[†]

Abstract

Multi-key fully homomorphic encryption (MKFHE) supports homomorphic computation over ciphertexts encrypted under independently generated keys. Known constructions accommodate independent keys by expanding every ciphertext under the concatenation of the participating secret keys, and then evaluating the circuit in that expanded dimension. A concatenated key is N times longer than an individual key, so the ciphertext on each wire and the running time of each gate grow with N , the number of users. López-Alt, Tromer, and Vaikuntanathan, who introduced MKFHE, asked for a scheme in which all algorithms are independent of N . More than a decade later, no MKFHE scheme has a homomorphic evaluation whose cost is even sublinear in N .

We construct the first MKFHE whose homomorphic evaluation asymptotically matches that of single-key FHE, while users still generate their keys and encrypt independently and the participating set is chosen only after encryption. We insert a public, deterministic, and non-interactive preprocessing phase between encryption and evaluation: key aggregation combines the selected public keys, and ciphertext preparation converts each independently encrypted input into a short single-key ciphertext under a *virtual key*, namely the sum of the participating secret keys. Evaluation then runs on ordinary single-key ciphertexts, so every wire ciphertext, every gate, and the decryptable output are independent of N .

We give two instantiations, a leveled GSW-based scheme over $\mathbb{Z}_q$ from standard LWE and a leveled BFV-based scheme over cyclotomic rings from circular Ring-LWE. Both retain single-round distributed decryption with statistically simulatable partial decryptions, where the published key material grows with N . As a proof of concept, we implement our ring-based construction in Go. For 32 users, its multiplication is $275.13\times$ and $45.50\times$ faster than CDKS and KKLSS, and its ciphertexts are $16.5\times$ smaller than both.

*Email: abtin@cs.wisc.edu.

[†]Email: rishab@cs.wisc.edu.

Contents

1	Introduction	3
1.1	Our Results	4
2	Technical Overview	6
2.1	GSW and Multi-Key Expansion	7
2.2	Optimal Evaluation Over Fields	8
2.3	From Fields to Rings	10
2.4	Further Discussions	12
3	Preliminaries	14
3.1	Lattice Preliminaries	15
3.2	Multi-Key FHE	17
4	Fast GSW-Based Multi-Key FHE from Plain LWE	20
4.1	Compressing an Inner-Product Ciphertext	20
4.2	Column Selection and Ciphertext Preparation	21
4.3	Construction	22
4.4	Analysis	24
4.5	Distributed Decryption	26
5	Fast BFV-Based Multi-Key FHE from Ring-LWE	26
5.1	Preparing BFV Ciphertexts by Compression	26
5.2	Relinearization under the Sum Secret	27
5.3	Construction	28
5.4	Analysis	30
5.5	Distributed Decryption	33
A	Background on (Multi-Key) FHE	38
A.1	MKFHE over Fields: GSW and MW	38
A.2	MKFHE over Rings: BFV and CDKS	41
B	Implementation and Evaluation	43
C	Removing User Indices	45

1 Introduction

Fully homomorphic encryption (FHE) [RAD78, Gen09, BV11, BGV12] enables computation on encrypted data. In standard FHE, however, every input must be encrypted under the same key. Multi-key FHE (MKFHE) removes this restriction: users generate keys and encrypt *independently*, and an evaluator may later choose a set of users, compute jointly on their ciphertexts, and produce an output that those users can decrypt together [LATV12, CM15, MW16]. The users need not know one another when they encrypt, participate in a joint key-generation protocol, or remain online while the computation is chosen. This flexibility underlies applications including round-optimal multiparty classical and quantum computation [AJLA⁺12, LATV12, MW16, AJJM20, Goy18, DGJ⁺20], multi-key homomorphic secret sharing [DHRW16, CDH⁺25], communication-preserving compilers for general MPC [DGMR21, ACG⁺24], threshold variants of cryptographic primitives [BGG⁺18, AG26, AGY26], and privacy-preserving distributed machine learning and classification tasks [BPTG14, WFNL15, GBDL⁺16, JVC18].

The concatenation barrier. Existing MKFHE schemes typically accommodate independently generated keys by “expanding” each ciphertext under the concatenation $\hat{\mathbf{t}} = (\mathbf{t}_1, \dots, \mathbf{t}_N)$ of the participating secret keys. The circuit is then evaluated in this expanded dimension. Because $\hat{\mathbf{t}}$ is N times longer than an individual key, ciphertext size and evaluation complexity grow with N , the number of users. This dependence holds for all existing MKFHE schemes.

In the GSW-based [GSW13] constructions of Clear-McGoldrick [CM15] and Mukherjee-Wicks [MW16], an expanded ciphertext is an $nN \times mN$ matrix over $\mathbb{Z}_q$ with $O(nm^2N^3)$ field operations for multiplication. In Ring-LWE constructions based on BFV [Bra12, FV12], ciphertexts contain $N + 1$ ring elements, and multiplication requires $O(N^2)$ work [CDKS19]. Several recent works have reduced the dependence in particular components, but homomorphic evaluation has continued to operate in a representation whose dimension grows with the participating set. The common denominator in all prior schemes is that the ability to support homomorphic evaluation on multi-source encrypted data comes at an exorbitantly high cost (i.e., per-gate evaluation $\propto N$).

López-Alt, Tromer, and Vaikuntanathan, who introduced MKFHE, asked whether an MKFHE scheme exists “where all algorithms are independent of N ” [LATV12, §3.1]. However, despite significant research efforts for over a decade, this question has remained completely open. Remarkably, we still do not have any MKFHE scheme where the dependence is even a sub-linear function of N , let alone independent of N .

Recently, Afshar-Goyal [AG26] designed the first MKFHE scheme where the evaluated ciphertext can be made sublinear in N . In fact, the size of evaluated ciphertexts in their scheme is $\ell + \text{poly}(\lambda, d, \log N)$, where ℓ is the length of the encrypted message. However, their scheme works as a ‘ciphertext-compression’ wrapper on prior MKFHE schemes. In other words, homomorphic evaluation in [AG26] still works on *expanded ciphertext* and their compression is applied only after evaluation finishes. So, the running time and working space of the evaluator retain the polynomial dependence on N even for a single gate.

In this work, we revisit the above open question [LATV12], and ask:

Can multi-key homomorphic evaluation have the same per-gate cost as single-key evaluation?

More precisely, can every evaluated ciphertext, every homomorphic gate, and the decryptable output be independent of N , while users still generate keys independently and the participating set is chosen only after encryption?

1.1 Our Results

We answer this question affirmatively. Assuming standard LWE [Reg09], we design an MKFHE scheme where the cost of homomorphic evaluation (asymptotically) matches that of single-key FHE. In particular, we design a new process to take a MKFHE ciphertext and (non-interactively) transform it into a *short* single-key ciphertext under a *compact virtual key*. Here the virtual key can be deterministically derived as a simple function of secret keys of participating users.

Essentially, our MKFHE construction moves the party-dependent work into a public preprocessing phase performed after the user set is chosen but before circuit evaluation begins. This phase aggregates the selected public keys and converts each independently encrypted input into a ciphertext under a compact virtual key. Once this conversion is complete, the evaluator uses ordinary single-key ciphertexts and single-key gate operations. The preprocessing is non-interactive and uses only public information. In our construction, the size of the public material grows with N , but the size of ciphertexts before, during, and after evaluation are independent of N . The only operation whose complexity scales with N is the preprocessing phase that transforms individual MKFHE ciphertexts into short evaluation-ready ciphertexts. This work is performed before evaluation starts, can be reused across circuits for the same user set, and is never repeated during the evaluation process.

To present our ideas in full generality, we formalize the preprocessing phase through two new algorithms between encryption and evaluation:

Key aggregation takes the public keys of participating users and produces a preparation key `PrepKey` and an evaluation key `EvalKey`.

Ciphertext preparation uses `PrepKey` to convert a fresh ciphertext into a compact ciphertext under the virtual key associated with those users.

Both algorithms are deterministic and do not need any secret keys. Encryption remains independent of the eventual participating set. Evaluation receives only prepared ciphertexts and `EvalKey`, and its complexity is independent of N .

We give two instantiations. The first builds on GSW over $\mathbb{Z}_q$ [GSW13] and converts expanded [MW16] ciphertexts into ordinary GSW matrices. The second starts from BFV over a cyclotomic ring [FV12, Bra12] and converts each input into an ordinary two-component BFV ciphertext. The constructions require different techniques because GSW multiplication needs no evaluation key, whereas BFV multiplication requires relinearization under the virtual secret.

Theorem 1.1 (Fast GSW-based MKFHE, informal; Theorem 4.3 and Corollary C.3). Assuming LWE, there is a leveled MKFHE in which every prepared or evaluated ciphertext is an $n \times m$ matrix over $\mathbb{Z}_q$. Homomorphic addition costs $O(nm)$ field operations, homomorphic multiplication costs $O(nm^2)$ field operations, and no evaluation key is required. We obtain two variants. In the indexed variant key aggregation and ciphertext preparation cost $O(N^2n)$ and $O(Nnm^2)$ field operations, respectively. In the unindexed variant, these costs are $O(N^2n^2m^2)$ and $O(n^2m^2)$, respectively. Key aggregation is performed once per participating set, and each prepared input can be reused for further computations on that set.

Here $n - 1$ is the LWE dimension and $m = n \lceil \log_2 q \rceil + 2\lambda$. After preparation, evaluation uses the unmodified GSW operations $\text{CT}_1 + \text{CT}_2$ and $\text{CT}_1 \mathbf{G}^{-1}(\text{CT}_2)$. Thus, the $nN \times mN$ expanded representation is used only within preprocessing; every circuit wire thereafter carries an $n \times m$ matrix. The main ingredient is a public compression procedure for the [MW16] expansion. It maps

the expanded secret-key relation to a GSW relation under a virtual key formed from the selected users' secrets. For fresh inputs, we exploit the sparsity of the expanded ciphertext and perform this transformation without materializing the full $nN \times mN$ matrix. This yields preparation time linear, rather than quadratic, in N in the indexed variant.

Theorem 1.2 (Fast BFV-based MKFHE, informal; Theorem 5.2 and Corollary C.3). Assuming circular Ring-LWE, there is a leveled MKFHE in which every prepared or evaluated ciphertext consists of two elements of R_q . Homomorphic addition and multiplication both cost $O(1)$ ring operations. In both our indexed and unindexed variants, key aggregation and ciphertext preparation cost $O(N^2)$ and $O(1)$ ring operations, respectively. Key aggregation is performed once per participating set, and each prepared input can be reused for further computations on that set.

The difficulty in the ring setting is relinearization. If the virtual secret is $\tilde{\sigma} = \sum_{i=1}^N \sigma_i$, then multiplication produces a term involving $\tilde{\sigma}^2$, including cross-products between different users' secrets. Ordinary single-user BFV relinearization keys do not provide these cross-products and cannot simply be added. We address this problem by aggregating masked [CDKS19] key vectors whose defining equations are linear in each user's secret contribution. Their sum gives a constant-size relinearization key for $\tilde{\sigma}$, allowing every subsequent multiplication to use an ordinary two-component BFV representation.

Both constructions also support single-round distributed decryption with statistically simulatable partial decryptions, following the standard smudging approach [MW16, CDKS19].

As per our terminology (see Section 3.2), both constructions achieve full succinctness and optimal evaluation: after public preprocessing, ciphertext size, evaluation-key size, per-gate running time, and decryptable payload are bounded by $\text{poly}(\lambda, d, \log N)$, with no explicit enumeration of the participating users at a circuit gate; see Definitions 3.7 and 3.8. The aggregated preparation key is linear in N , however, it can be reused for all inputs and circuits involving the same user set. Per-input preparation need not read the entire preparation key. The resulting compact ciphertext can be reused for different computations involving the same user set.

Asymptotics. Table 1 separates our indexed constructions, given in Theorems 4.3 and 5.2 over fields and rings respectively, from their unindexed variants obtained by the generic transformation in Corollary C.3.¹ It compares our constructions with the best known multi-key FHE costs for each row metric separately, noting that all those metrics do not occur together in one construction; they are a collection of best existing parameters across multi-key FHE literature. These are mostly the costs of [MW16] over fields and [CDKS19] over rings, with the following improvements. Over fields, [BP16] reduces wire ciphertexts from N^2nm to Nn , and fresh ciphertexts from nm^3 to n , and [AG26] reduces the payload size from Nn to n . For the plain-LWE analogue of [CZW17],² our accounting gives reductions from Nnm^2 to n for ciphertext preparation, from N^2nm to Nn^2 for addition, from N^3nm^2 to N^2n^3 for multiplication, and from N^2nm to Nn for wire ciphertext size; however, these optimizations come at the cost of large keys and heavy preprocessing. Over rings,

¹The reported preparation-key sizes includes material for ciphertext preparation for all N users. However, preparing one user's ciphertext accesses only $O(1)$ ring elements in either ring construction or $O(n^2m)$ field elements in the unindexed field construction.

²The construction of [CZW17] is presented over rings, and the authors note that the field construction is analogous. However, they provide neither an explicit field construction nor measurements for it. The corresponding efficiency improvements considered here are therefore our estimates, without a formal proof or empirical validation.

[KKL⁺23, ZHW25] reduce multiplication from N^2 to N , and the no-CRS construction of [MPS26] removes the setup material.

	Efficiency metric	Fields			Rings		
		SotA	This work		SotA	This work	
			Indexed	Unindexed		Indexed	Unindexed
Run-time	Addition	Nn^2	nm	nm	N	1	1
	Multiplication	N^2n^3	nm^2	nm^2	N	1	1
	Key aggregation	—	N^2n	$N^2n^2m^2$	—	N^2	N^2
	Ciphertext preparation	n	Nnm^2	n^2m^2	1	1	1
Sizes	Ciphertext on a wire	Nn	nm	nm	N	1	1
	Decryptable payload	n	n	n	N	1	1
	Fresh ciphertext	n	nm^3	n	1	1	1
	Evaluation key	—	—	—	N	1	1
	Preparation key	—	Nm	Nn^2m	—	N	N
	Public key/user	m	$Nn + m$	$N^2n + Nn^2m^3$	1	N	N^2
	Setup material	—	Nn^2	Nn^2	—	N	N

Table 1: Asymptotic costs at fixed global parameters. SotA reports optimal State-of-the-Art values per metric across all constructions; these metrics do not co-exist in a single construction. Preparation times are per user and assume random access. Sizes and arithmetic operations are counted over $\mathbb{Z}_q$ per plaintext bit or R_q per plaintext polynomial. Since element sizes and arithmetic costs also depend on $\log q$ we consider that as part of fixed parameters and suppress factors polynomial in $\log q$, including gadget lengths.

Implementation. We implement our ring-based construction in Go as a proof of concept (see Appendix B). At fixed parameters and after input preparation, multiplication time stays nearly constant for $N \in \{2, 4, 8, 16, 32\}$ users, with both operands involving all N users. Across these user counts, multiplication is 1.83 to 275.13 times faster than our CDKS baseline and 3.02 to 45.50 times faster than KKLSS, while taking about 1.4 times the time of single-key BFV. Our ciphertexts are 1.5 to 16.5 times smaller than those of CDKS and KKLSS, and the same size as single-key BFV ciphertexts. Table 2 reports the measured times and sizes.

2 Technical Overview

Our starting point is the approximate eigenvalue relation of GSW [GSW13] and the way CM [CM15] and MW [MW16] generalize it to many keys, by concatenating the participating secret keys (Section 2.1). That one decision is what makes every cost in a multi-key scheme grow with N . We replace the concatenation by a sum, and turn each independently encrypted input into an ordinary single-key ciphertext under that sum before evaluating any gate (Section 2.2); over rings the preparation gets easier and homomorphic multiplication gets harder (Section 2.3). Section 2.4 collects the costs, the optimizations they admit, and the questions we leave open.

2.1 GSW and Multi-Key Expansion

The GSW relation. A GSW ciphertext of a bit μ is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ whose defining property is stated against the augmented secret key $\mathbf{t} = (-\mathbf{s}^\top, 1)^\top$,

$$\mathbf{t}^\top \mathbf{C} = \mu \mathbf{t}^\top \mathbf{G} + \mathbf{e}^\top, \quad (1)$$

where $\mathbf{e}$ is short and $\mathbf{G} = [I_n \otimes \mathbf{g}^\top \mid \mathbf{0}]$ is built from $\mathbf{g} = (1, 2, \dots, 2^{\log q - 1})^\top$. The encrypted bit is an approximate eigenvalue of $\mathbf{C}$ with approximate eigenvector $\mathbf{t}$, which is why homomorphism is free of any evaluation key: for the public binary decomposition $\mathbf{G}^{-1}$ with $\mathbf{G}\mathbf{G}^{-1}(\mathbf{x}) = \mathbf{x}$, we get $\mathbf{t}^\top (\mathbf{C}\mathbf{G}^{-1}(\mathbf{x})) = \mu \mathbf{t}^\top \mathbf{x} + \mathbf{e}^\top \mathbf{G}^{-1}(\mathbf{x})$, so $\mathbf{C}_1 + \mathbf{C}_2$ and $\mathbf{C}_1 \mathbf{G}^{-1}(\mathbf{C}_2)$ add and multiply the encrypted bits. Nothing but ciphertext matrices and the public gadget takes part.

Viewing the relation through “phases”. It helps to read Eq. (1) one column at a time. Call $\mathbf{t}^\top \mathbf{c}$ the *phase* of a column $\mathbf{c}$, and say that a column *encodes* a value when its phase is that value up to a small error; we write $\approx$ for equalities that hide such an error. Writing $\mathbf{c}_{k,\alpha}$ for the column of $\mathbf{C}$ aligned with $2^\alpha \mathbf{e}_k$ in $\mathbf{G}$, Eq. (1) reads

$$\mathbf{t}^\top \mathbf{c}_{k,\alpha} \approx \mu 2^\alpha t_k = \begin{cases} -\mu 2^\alpha s_k, & k < n, \\ \mu 2^\alpha, & k = n. \end{cases} \quad (2)$$

The block $k = n$ carries the message, since $t_n = 1$; the blocks $k < n$ carry the same bit scaled by the individual secret coordinates, and they are what multiplication consumes.

The MW expanded ciphertext. In the multi-key setting, each user i independently generates a secret $\mathbf{s}_i$ and encrypts under its own public key. A fresh ciphertext contains a GSW matrix $\mathbf{C}$ together with auxiliary encryptions U of its randomness. Given the selected public keys, this auxiliary material supports public expansion to a matrix $\hat{\mathbf{C}} \in \mathbb{Z}_q^{nN \times mN}$ satisfying

$$\hat{\mathbf{t}}^\top \hat{\mathbf{C}} \approx \mu \hat{\mathbf{t}}^\top \hat{\mathbf{G}}_N, \quad \text{s.t.} \quad \hat{\mathbf{t}} = (\mathbf{t}_1^\top, \dots, \mathbf{t}_N^\top)^\top, \quad \hat{\mathbf{G}}_N = I_N \otimes \mathbf{G}. \quad (3)$$

The matrix $\hat{\mathbf{C}}$ has one block row and one column group per user. Reading Eq. (3) at the column (j, k, α) of group j gives the multi-key analogue of Eq. (2),

$$\hat{\mathbf{t}}^\top \hat{\mathbf{c}}_{j,k,\alpha} \approx \begin{cases} -\mu 2^\alpha s_{j,k}, & k < n, \\ \mu 2^\alpha, & k = n. \end{cases} \quad (4)$$

For homomorphism, since only Eq. (1) was used to justify the gate algorithms, the same two formulas $\hat{\mathbf{C}}_1 + \hat{\mathbf{C}}_2$ and $\hat{\mathbf{C}}_1 \hat{\mathbf{G}}_N^{-1}(\hat{\mathbf{C}}_2)$ evaluate any circuit on expanded ciphertexts.

The concatenation is the cost. The paradigm is natural and the price is structural. The key against which the relation holds is a concatenation of N local keys, so the relation needs N block rows; and each user needs its own group of correlated columns, so the relation needs N column groups. Hence ciphertext size grows with N^2 , and the time for one multiplication grows with N^3 . Translating to rings replaces the matrix by a vector of $N + 1$ ring elements, and the cost reappears in the relinearization of the tensor products, which grows with N^2 in [CDKS19], and with N in [KKL+23]. Every one of these costs traces to the same decision: *a single-key ciphertext is lifted to a key that is a concatenation of N keys, and the circuit is evaluated after the lift.*

2.2 Optimal Evaluation Over Fields

The goal. Replace the concatenation by a sum, and publicly turn every input into a matrix $\mathbf{CT} \in \mathbb{Z}_q^{n \times m}$ satisfying

$$\tilde{\mathbf{t}}^\top \mathbf{CT} \approx \mu \tilde{\mathbf{t}}^\top \mathbf{G}, \quad \text{s.t.} \quad \tilde{\mathbf{s}} = \sum_{i=1}^N \mathbf{s}_i, \quad \tilde{\mathbf{t}} = (-\tilde{\mathbf{s}}^\top, 1)^\top. \quad (5)$$

We call $\tilde{\mathbf{t}}$ a *virtual key*: it is a well-defined secret determined by the chosen users, no algorithm in the scheme ever reconstructs it, and no user knows it. The question is whether this target key can be reached from independently generated single-key ciphertexts, publicly, after the fact, by an untrusted evaluator holding nothing but published keys. Since Eq. (5) is Eq. (1) verbatim, every input that satisfies it is an ordinary GSW ciphertext, and the circuit runs at single-key dimensions with no evaluation key at all.

Two public algorithms. The work that precedes evaluation splits into the two algorithms of Definition 3.6. **KeyAgg** turns the chosen users' published keys into **PrepKey** and **EvalKey**, one fixed cost per user set that is reused for every input and every circuit of that set; **CTPrep** then turns one independently generated ciphertext into a ciphertext satisfying Eq. (5). Both are deterministic and read only published data, so an untrusted evaluator runs them after the inputs are encrypted.

CTPrep, in three steps. Our preparation composes three operations on one input ciphertext:

$$(U, \mathbf{C}) \xrightarrow{\text{(i) expand}} \underbrace{\hat{\mathbf{C}}}_{\in \mathbb{Z}_q^{nN \times mN}} \xrightarrow{\text{(ii) compress rows}} \underbrace{\mathbf{Z}}_{\in \mathbb{Z}_q^{n \times mN}} \xrightarrow{\text{(iii) reduce columns}} \underbrace{\mathbf{CT}}_{\in \mathbb{Z}_q^{n \times m}}.$$

The input is a fresh ciphertext under $\mathbf{t}_i$, the output of step (i) is read under $\hat{\mathbf{t}}$, and the outputs of steps (ii) and (iii) are read under $\tilde{\mathbf{t}}$. Step (i) is MW expansion, used unchanged and used only as a stepping stone; no gate is ever applied to $\hat{\mathbf{C}}$. Step (ii) shortens every column of $\hat{\mathbf{C}}$ from nN to n coordinates while preserving its phase, producing $\mathbf{Z}$ and changing the key on the left of the relation. Step (iii) turns the N user-indexed groups of columns into the one group that $\mathbf{G}$ prescribes, which changes the encoded values on the right of the relation. Steps (ii) and (iii) are the contribution of this section, and they are independent of each other: neither reads the message, and neither needs to know what its input encodes. The construction of Section 4 applies them in the opposite order, for a reason we give at the end of Section 2.2.2.

2.2.1 Step (ii): Vector Compression

Consider one expanded column $\mathbf{c} \in \mathbb{Z}_q^{nN}$, with user blocks $\mathbf{c}_j \in \mathbb{Z}_q^n$. Its phase is

$$\hat{\mathbf{t}}^\top \mathbf{c} = \sum_j c_{j,n} - \sum_{j,k < n} s_{j,k} c_{j,k}.$$

The first sum is public. The second is not, and this is the whole difficulty: an evaluator knows every coefficient $c_{j,k}$ and none of the secrets $s_{j,k}$. To reproduce that sum under the shorter key $\tilde{\mathbf{t}}$, we need public vectors whose phases encode the individual $s_{j,k}$.

The cancellation. As a first attempt, fix a common public mask $\mathbf{a}_{j,k} \in \mathbb{Z}_q^{n-1}$. Each user i publishes

$$w_{i,j,k} = \langle \mathbf{s}_i, \mathbf{a}_{j,k} \rangle + \delta_{i,j} s_{i,k} + \varepsilon_{i,j,k}, \quad (6)$$

where $\delta_{i,j}$ is 1 when $i = j$ and 0 otherwise, and $\varepsilon_{i,j,k}$ is small. Every user contributes its own part of the mask, while only user j inserts the secret coordinate to be encoded. Summing these entries gives

$$W_{j,k} = \sum_i w_{i,j,k} = \langle \tilde{\mathbf{s}}, \mathbf{a}_{j,k} \rangle + s_{j,k} + \varepsilon_{j,k}, \quad \tilde{\mathbf{t}}^\top \begin{pmatrix} \mathbf{a}_{j,k} \\ W_{j,k} \end{pmatrix} = s_{j,k} + \varepsilon_{j,k}, \quad (7)$$

where $\varepsilon_{j,k} = \sum_i \varepsilon_{i,j,k}$. The mask cancels in the phase under the sum secret, although no user needed to know that secret to generate its contribution. Write $\mathbf{K}_{j,k} = (\mathbf{a}_{j,k}^\top, W_{j,k})^\top$. The natural compressed column is

$$\mathbf{c}' = \begin{pmatrix} \mathbf{0}_{n-1} \\ \sum_j c_{j,n} \end{pmatrix} - \sum_{j,k < n} c_{j,k} \mathbf{K}_{j,k}.$$

It has only n coordinates, and its phase is $\tilde{\mathbf{t}}^\top \mathbf{c}' = \hat{\mathbf{t}}^\top \mathbf{c} - \sum_{j,k < n} c_{j,k} \varepsilon_{j,k}$. This gives the desired cancellation. However, a ciphertext coefficient $c_{j,k}$ can have magnitude comparable to q , so the last sum can amplify even a small encoding error beyond the decryption margin.

Controlling the error with gadget decomposition. We instead publish encodings of $2^\alpha s_{j,k}$, using the same construction with an independent mask for each power of two. Denote the resulting public columns by $\mathbf{K}_{j,k,\alpha}$. Their phases are $\tilde{\mathbf{t}}^\top \mathbf{K}_{j,k,\alpha} = 2^\alpha s_{j,k} + \varepsilon_{j,k,\alpha}$. Now write $c_{j,k} = \sum_\alpha 2^\alpha \beta_{j,k,\alpha} \pmod{q}$, with binary digits $\beta_{j,k,\alpha}$. Compression becomes

$$\text{Compress}(\mathbf{c}) = \begin{pmatrix} \mathbf{0}_{n-1} \\ \sum_j c_{j,n} \end{pmatrix} - \sum_{j,k < n, \alpha} \beta_{j,k,\alpha} \mathbf{K}_{j,k,\alpha}.$$

The same cancellation recovers the original phase, but each aggregate error is now multiplied by 0 or 1 which controls error growth. Decoding then succeeds under the chosen noise budget.

Compression shortens a column from nN to n coordinates while preserving its phase under $\tilde{\mathbf{t}}$, up to controlled error. It does not depend on the encoded value, so it applies equally to an expanded column, a sum of such columns, or a decryption vector. Applying it to every column of $\hat{\mathbf{C}}$ gives $\mathbf{Z} \in \mathbb{Z}_q^{n \times mN}$ satisfying

$$\tilde{\mathbf{t}}^\top \mathbf{Z} \approx \mu \hat{\mathbf{t}}^\top \hat{\mathbf{G}}_N. \quad (8)$$

The key on the left is now the virtual key, but the encoded values on the right still correspond to the individual local keys.

2.2.2 Step (iii): Column Reduction

The matrix $\mathbf{Z}$ still has N groups of columns. Write $\mathbf{z}_{j,k,\alpha} = \text{Compress}(\hat{\mathbf{c}}_{j,k,\alpha})$ for its column at position (j, k, α) . For $k < n$, its phase under $\tilde{\mathbf{t}}$ is still approximately $-\mu 2^\alpha s_{j,k}$, whereas Eq. (5) requires $-\mu 2^\alpha \tilde{s}_k$, the same multiple of the *summed* coordinate.

Adding columns adds their phases, and that is the entire step. Fix $k < n$ and α , and take two users. Their compressed columns at that position encode $-\mu 2^\alpha s_{1,k}$ and $-\mu 2^\alpha s_{2,k}$, so their sum has phase $\tilde{\mathbf{t}}^\top (\mathbf{z}_{1,k,\alpha} + \mathbf{z}_{2,k,\alpha}) \approx -\mu 2^\alpha (s_{1,k} + s_{2,k})$, and for N users the same sum gives $-\mu 2^\alpha \tilde{s}_k$. The

operation reads no secret and no plaintext. It is an addition of public vectors. For the constant coordinate, every group already has phase $\mu 2^\alpha$ at $k = n$, because every local augmented key ends in 1, so adding all N of them would produce $N\mu 2^\alpha$ while the last coordinate of $\tilde{\mathbf{t}}$ is still 1. We keep one group and discard the rest. Setting

$$\mathbf{y}_{k,\alpha} = \begin{cases} \sum_{j=1}^N \mathbf{z}_{j,k,\alpha}, & k < n, \\ \mathbf{z}_{1,n,\alpha}, & k = n, \end{cases} \quad (9)$$

placing these columns in the coordinate-and-power order of $\mathbf{G}$, and appending $m - n \log q$ zero columns to match its padding gives the prepared ciphertext $\mathbf{CT} = \mathbf{Y} \in \mathbb{Z}_q^{n \times m}$ with $\tilde{\mathbf{t}}^\top \mathbf{Y} \approx \mu \tilde{\mathbf{t}}^\top \mathbf{G}$, as required by Eq. (5). The remaining error comes from the expansion and compression errors of the columns we retained or added.

In the construction in Section 4, we reverse these two steps. We first add or select the corresponding expanded columns, leaving m columns of length nN , and then compress each one. This requires m compressions instead of mN and introduces one compression error per output column. Both orders produce the required phases. They need not produce identical ciphertexts, since binary decomposition is not additive. We use the reversed order to reduce both preparation work and the added error.

2.2.3 Putting the Plain-LWE Construction Together

Each user publishes its MW public key $\mathbf{pk}_j$ and its independent compression contributions. **KeyAgg** sums the latter into compression keys $\mathbf{CK}$ and outputs $\mathbf{PrepKey} = ((\mathbf{pk}_j)_{j \in [N]}, \mathbf{CK})$ and $\mathbf{EvalKey} = \perp$. The source public keys support expansion, while $\mathbf{CK}$ supports compression. For each fresh input, **CTPrep** performs MW expansion, column reduction, and vector compression, and its output satisfies $\tilde{\mathbf{t}}^\top \mathbf{CT}^{\text{prep}} = \mu \tilde{\mathbf{t}}^\top \mathbf{G} + \mathbf{e}_{\text{prep}}^\top$. All prepared inputs use the same virtual key, so **Eval** applies ordinary GSW homomorphic gate operations throughout the circuit. The chosen parameters allow for preparation error followed by the usual depth-dependent evaluation error.

For decryption, the constant-coordinate columns yield a vector $(\mathbf{u}^\top, v)^\top$ whose phase $v - \langle \tilde{\mathbf{s}}, \mathbf{u} \rangle$ encodes the output bit at scale $\Delta = (q - 1)/2$. Joint decryption forms the sum secret and rounds this phase. Alternatively, each user contributes $-\langle \mathbf{s}_i, \mathbf{u} \rangle$, with the public term v included once. Adding these contributions recovers the same phase. Fresh smudging noise in each contribution gives single-round distributed decryption with simulatable partial decryptions, using an enlarged modulus to accommodate that noise. The formal construction proves the security of linear key-dependent compression entries under plain LWE by an affine-KDM argument.

2.3 From Fields to Rings

Over rings preparation is simpler, while multiplication gets harder. We now turn to BFV [FV12, Bra12] and its multi-key extension by CDKS [CDKS19], over $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ with n a power of two, messages in R_t , and $\Delta = (q - 1)/t$.

One phase instead of a matrix of phases. A BFV ciphertext is a pair (c_0, c_1) whose phase is $c_0 + c_1 s \approx \Delta \mu$, and CDKS expands it to a vector $(c_0, c_1, \dots, c_N)^\top$ against the concatenated key $(1, s_1, \dots, s_N)$, and with phase $c_0 + \sum_{j=1}^N c_j s_j \approx \Delta \mu$. A fresh ciphertext owned by j has only its

owner coordinate nonzero among the secret-dependent coordinates. Expansion simply inserts zeros at the other positions.

The expanded object represents one message through one phase, so there are no correlated groups of columns and no analogue of Section 2.2.2: preserving that single phase under a single secret already produces a two-element ciphertext. Everything reduces to a ring version of Section 2.2.1.

Compression over the ring. Each user samples a compression secret σ_i independently of its BFV secret s_i , and the virtual secret is $\tilde{\sigma} = \sum_i \sigma_i$; compression is then a key switch from the source secrets $(s_j)_j$ to $\tilde{\sigma}$. Here all secrets are sampled from a short distribution. With uniform masks $\mathbf{a}_j \in R_q^k$ from setup, a compression gadget $(h, \mathbf{g})$ of length k s.t. $\langle h(c), \mathbf{g} \rangle = c$, and contributions

$$\mathbf{w}_{i,j} = \sigma_i \mathbf{a}_j + \delta_{i,j} s_i \mathbf{g} + \mathbf{e}_{i,j}, \quad \text{so that} \quad \mathbf{W}_j = \sum_{i=1}^N \mathbf{w}_{i,j} = \tilde{\sigma} \mathbf{a}_j + s_j \mathbf{g} + \mathbf{E}_j,$$

the pattern of Eqs. (6) and (7) repeats with ring multiplication in place of scalar multiplication. Compression computes

$$u = \sum_j \langle h(c_j), \mathbf{a}_j \rangle, \quad v = c_0 + \sum_j \langle h(c_j), \mathbf{W}_j \rangle,$$

whose phase is $v - u\tilde{\sigma} \approx c_0 + \sum_j c_j s_j$. Thus $(v, -u)$ is a BFV pair under the virtual secret $\tilde{\sigma}$.

The new obstacle is relinearization. Addition of prepared pairs is componentwise. Multiplication is not free, because BFV evaluation, unlike GSW, uses secret-dependent public material. Scaling and rounding the product of two phases leaves three coefficients,

$$(c_0 + c_1 \tilde{\sigma})(c'_0 + c'_1 \tilde{\sigma}) \rightsquigarrow T_0 + T_1 \tilde{\sigma} + T_2 \tilde{\sigma}^2,$$

and relinearization has to replace $T_2 \tilde{\sigma}^2$ by something linear in $\tilde{\sigma}$. A conventional BFV relinearization key encodes gadget multiples of the square of the decryption secret, and here the square does not decompose along the users: already for two,

$$(\sigma_1 + \sigma_2)^2 = \sigma_1^2 + 2\sigma_1 \sigma_2 + \sigma_2^2,$$

so the sum of locally generated square keys misses the cross term, and no user can compute that term from its own secret. Aggregating the usual keys is not merely lossy; it is wrong.

Aggregating masked relinearization keys. What we need is a set of public equations that (a) are satisfiable by each user from its own secrets alone, (b) are linear in those secrets, so that summing the contributions satisfies the same equations for the sums, and (c) let the evaluator manufacture a quadratic term at gate time. The masked representation of CDKS has exactly this shape. Specialized to a single secret σ with an auxiliary secret r and common public masks $\mathbf{a}, \mathbf{u} \in R_q^k$, it asks for

$$\mathbf{B} \approx -\sigma \mathbf{a}, \quad \mathbf{D} \approx -r \mathbf{a} + \sigma \mathbf{g}, \quad \mathbf{V} \approx -\sigma \mathbf{u} - r \mathbf{g}, \quad (10)$$

so that $(\mathbf{D}, \mathbf{a})$ encodes multiples of σ under r and $(\mathbf{V}, \mathbf{u})$ encodes multiples of $-r$ under σ . Every secret appears to degree one, multiplied by public coefficients. User i publishes these vectors using only σ_i and a freshly sampled r_i , and **KeyAgg** adds the three families to obtain Eq. (10) for $(\sigma, r) = (\tilde{\sigma}, \sum_i r_i)$. Common masks are what make this work, and the mask $\mathbf{u}$ is where we depart from CDKS, which samples it per user; with a common $\mathbf{u}$ the terms $\sigma_i \mathbf{u}$ add to $\tilde{\sigma} \mathbf{u}$. We note that the common-mask aggregation technique of the CDKS relinearization vectors was previously considered by [KLSW21].

The quadratic secret term appears when the evaluator uses these vectors. Decompose T_2 as $\mathbf{z} = h(T_2)$ and compute $x = \langle \mathbf{z}, \mathbf{B} \rangle \approx -\sigma \langle \mathbf{z}, \mathbf{a} \rangle$. Then the pair $(\langle h(x), \mathbf{V} \rangle, \langle \mathbf{z}, \mathbf{D} \rangle + \langle h(x), \mathbf{u} \rangle)$ has phase

$$\underbrace{\sigma \langle \mathbf{z}, \mathbf{D} \rangle}_{\approx \sigma^2 T_2 - \sigma r \langle \mathbf{z}, \mathbf{a} \rangle} + \underbrace{\langle h(x), \mathbf{V} + \sigma \mathbf{u} \rangle}_{\approx -rx \approx \sigma r \langle \mathbf{z}, \mathbf{a} \rangle} \approx \sigma^2 T_2.$$

The first vector produces the term we want together with a mask in r ; the public value x and the vector $\mathbf{V}$ produce the opposite mask. Adding T_0 and T_1 to the two components returns a BFV pair encrypting the product, and the result is ready for the next gate. Both decompositions use small digits, so the aggregated key errors enter weighted by digits.

Accordingly, ring key aggregation outputs the compression vectors in **PrepKey** and the three masked relinearization vectors in **EvalKey**. Ciphertext preparation produces a BFV pair, and evaluation maintains that representation using the aggregated vectors. The final phase is linear in $\tilde{\sigma}$, so joint and distributed decryption proceed by summing the users' secret contributions, as in the plain-LWE construction. The published ring key material is covered by the circular Ring-LWE assumption stated in Section 5.

2.4 Further Discussions

Here we discuss the costs of our constructions, an index-independent variant, related work, and open problems.

Our asymptotics over fields. Over fields (and in the indexed variant), each user publishes $N(n-1) \log q$ compression entries in addition to its m -element source public key. **KeyAgg** sums N contributions to each entry, giving the quadratic aggregation cost and a preparation key of $Nm + N(n-1) \log q = O(Nm)$ field elements. A fresh MW expansion has only $2N - 1$ nonzero blocks. Computing these blocks costs $O(Nnm^2 \log q)$ with random access to U . We compute these blocks one at a time, accumulate the columns needed for column reduction, and compress the resulting m vectors in $O(Nn^2m \log q)$ work. Thus preparation is linear in N without materializing the full expanded matrix. Our gates then act on $n \times m$ matrices, giving $O(nm)$ addition and $O(nm^2)$ multiplication, with no evaluation key.

Our asymptotics over rings. Over rings (and in the indexed variant), with binary gadgets, each user publishes $N \log q$ compression entries, three length- $\log q$ relinearization vectors, and a two-element source public key. Summing these contributions takes $O(N^2 \log q)$ additions and leaves $N \log q$ elements in **PrepKey** and $3 \log q$ in **EvalKey**. A fresh ciphertext has only one nonzero secret-dependent coordinate, so preparation uses its owner's compression vector, with one decomposition and two length- $\log q$ inner products. Homomorphic addition uses two ring additions, while multiplication uses two decompositions and four length- $\log q$ inner products in addition to the fixed-size

BFV tensor calculation. These give the $O(1)$ and $O(\log q)$ gate costs. The $O(1)$ CDKS preparation entry likewise uses an implicit zero-padded representation of a fresh ciphertext.

Global parameters. Noise still grows with N during aggregation and preparation. The global parameters account for it, with $n, m, \log q \leq \text{poly}(\lambda, d, \log N)$ over fields and, for the ring efficiency guarantee, binary gadgets and $n, \log q \leq \text{poly}(\lambda, d, \log N)$. The table separates this parameter dependence from the explicit work over users.

From indexed to plain MKFHE. Our construction assigns each user a unique index at key generation, whereas the standard MKFHE notion does not require this. A folklore transformation for slotted primitives removes this requirement at the cost of larger keys and fresh ciphertexts. Each user samples N independent secrets $\mathbf{s}_i^{(1)}, \dots, \mathbf{s}_i^{(N)}$ and publishes the corresponding public keys and compression entries for each candidate position ℓ . For block j , these entries use $\mathbf{s}_i^{(\ell)}$ in Eq. (6), with the inserted term present only when $j = \ell$. Fresh encryption encrypts the same message under all N keys. Once a participating set is chosen, we rank its members, say lexicographically, and use each user’s key material and ciphertext for its rank. The virtual key is the sum of the selected secrets. This increases fresh ciphertext size and per-user published material by a factor of N , while leaving prepared ciphertexts, gate costs, and decryption unchanged. A simple application of key switching (see Appendix C) makes the transformed construction comparable in efficiency to its indexed counterpart. Namely, encryption produces a single small fresh ciphertext at the cost of larger keys. Moreover, the potentially computationally heavy part of preprocessing moves from ciphertext preparation to key aggregation, allowing this work to be amortized across inputs. The indexed and unindexed columns of Table 1 quantify this tradeoff. Finally, a PRG can reduce the secret-key size.

Related work. MKFHE began with an NTRU-based construction for an a-priori bounded number of keys [LATV12], followed by LWE-based schemes with quadratic expanded ciphertexts [CM15, MW16]. Subsequent work achieves multi-hop evaluation without bootstrapping [PS16], linear ciphertext length and working space through branching programs [BP16], and batched multi-hop evaluation over rings [CZW17]. CDKS and KKLSS give practical ring-based evaluation [CDKS19, KKL+23].

Other work studies distributed setup for MKFHE [BHP17], plain-model constructions [AJJM20, MPS26], and circuit privacy even for maliciously generated keys and ciphertexts [CO17]. DHRW’s additive-function-sharing spooky encryption [DHRW16] produces ciphertexts under the individual keys whose plaintexts sum to the function output.

The closest work is that of Afshar-Goyal, who obtain, under plain LWE, evaluated ciphertexts of size $\ell + \text{poly}(\lambda)$ for an ℓ -bit output and public and secret keys of size $\text{poly}(\lambda)$ [AG26], by compressing the decryption information of an expanded ciphertext through succinct oblivious tensor evaluation [AMR25]. Their compressed object supports plaintext recovery but does not satisfy Eq. (1) or the BFV relation, and evaluation runs in the expanded dimension. Ours preserves these relations and evaluates at single-key dimensions, at the price of N compression entries per user. Their route is stated for plain LWE, and we are not aware of a ring analogue of the oblivious tensor evaluation it uses, whereas ours applies in both settings.

Summed keys already appear in threshold and multiparty HE when the party set is fixed before encryption [AJLA+12, BGG+18, MTPBH21, KLSW21]. Multi-key TFHE reduces per-gate costs

from quadratic to quasi-linear in N [CCS19, KMS24, AKÖ23, XMH⁺26]. In particular, [AKÖ23] uses a summed ring key with a concatenated LWE key and joint pre-computation for a fixed party set, while bootstrapping remains linear in N . Other compact schemes use interactive key generation [AJLA⁺12, DMR22], user-generated key-switching material for each joint key [ZCC⁺21], or two non-colluding servers [Jia22]. Our KeyAgg uses only previously published material, and CTPrep publicly converts ciphertexts encrypted before the participating set was chosen.

On shrinking the public material. Public material is where our constructions are weakest. In the indexed variants, each user publishes $N(n-1) \log q$ compression entries or $N \log q$ ring elements, so the published material is quadratic in the user count even though KeyAgg reduces the working material to linear. The construction of [AG26] keeps public and secret keys of size $\text{poly}(\lambda)$ while paying $\text{poly}(N)$ per gate; we do the opposite. Obtaining both at once is the natural next target, and the compression material is where the N sits in our scheme.

On multi-hop preparation and malicious keys. Our prepared wires are closed under the gate algorithms but not under the arrival of a new user: adding a party to a wire that has already been prepared requires returning to the retained inputs in our construction. A multi-hop version of preparation, and a treatment that survives maliciously generated keys rather than assuming honest key generation, both remain open.

Roadmap. Section 3 fixes notation, the lattice assumptions, and the MKFHE syntax and security definitions, including the two efficiency properties that our theorems establish. Sections 4 and 5 develop the two constructions, each starting with its compression tools and ending with correctness, cost, security, and distributed decryption. A reader who is less familiar with prior MKFHE schemes can find a more detailed overview of the base constructions in Appendix A. Appendix B presents our Ring-LWE implementation and compares its gate costs with CDKS, KKLSS, and single-key BFV. Appendix C gives a transformation from indexed to unindexed MKFHE with comparable asymptotic efficiency.

3 Preliminaries

Notations. We denote $[N] = \{1, \dots, N\}$ and $[M, N] = \{M, \dots, N\}$. We write λ for the security parameter, d for a circuit-depth bound, and $\text{negl}(\lambda)$ for a negligible function. All algorithms run in probabilistic polynomial time. All vectors are column vectors. For $\mathbf{v} \in \mathbb{Z}_q^n$, we write $\mathbf{v} = (v_1, \dots, v_n)^\top$, and a row is written explicitly as $\mathbf{v}^\top$. For vectors $\mathbf{v}_1, \dots, \mathbf{v}_t$, their concatenation is the column $(\mathbf{v}_1^\top, \dots, \mathbf{v}_t^\top)^\top$. We denote the inner product by $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}$ and coordinate-wise multiplication by $\mathbf{x} \circ \mathbf{y}$. The Kronecker delta $\delta_{i,j}$ is 1 if $i = j$ and 0 otherwise. We write I_n for the identity matrix, $\mathbf{e}_k$ for a standard basis column of the dimension indicated by its context, and $\otimes$ for the Kronecker product. For a vector, $\|\cdot\|_2$ and $\|\cdot\|_\infty$ denote its Euclidean and maximum-entry norms. For a matrix, $\|\cdot\|_\infty$ denotes its maximum absolute entry.

We use $\lfloor x \rfloor$ to denote rounding to the nearest integer, with ties toward positive infinity. For integers $p, q \geq 2$, scaled rounding from $\mathbb{Z}_p$ to $\mathbb{Z}_q$ maps x to $\lfloor (q/p)[x]_p \rfloor \bmod q$, where $[x]_p$ is the centered representative in $(-p/2, p/2]$. We also denote this map by $\lfloor x \rfloor_q$ when the source modulus p is clear. Centered reduction and rounding act entry-wise on vectors and matrices, and coefficient-wise on polynomials. When the modulus is clear from the context, we omit $\bmod q$. All logarithms

are base two. We write $\log q$ for $\lceil \log_2 q \rceil$ when it specifies a binary-decomposition length, dimension, or index range.

The statistical distance of two distributions X, Y over a countable domain Ω is $\Delta(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |X(\omega) - Y(\omega)|$. For distribution ensembles indexed by λ , we write $X \stackrel{\text{stat}}{\approx} Y$ if this distance is negligible, and $X \stackrel{\text{comp}}{\approx} Y$ for computational indistinguishability. The two-argument notation $\Delta(X, Y)$ is distinct from a plaintext scaling factor Δ . We write $x \leftarrow \mathcal{D}$ for sampling from a distribution $\mathcal{D}$, and $x \leftarrow S$ for uniform sampling from a finite set S . A distribution χ over the integers is B -bounded if $\Pr_{x \leftarrow \chi}[|x| \leq B] = 1$.

3.1 Lattice Preliminaries

An m -dimensional lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. Given positive integers n, m, q and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we denote

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\}, \quad \Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\}$$

for $\mathbf{u} \in \mathbb{Z}_q^n$. The latter is a coset of $\Lambda_q^\perp(\mathbf{A})$ when it is nonempty.

Discrete Gaussians. For $\sigma > 0$, define $\rho_\sigma(\mathbf{x}) = \exp(-\pi \|\mathbf{x}\|_2^2 / \sigma^2)$. For a discrete set $\mathcal{L} \subseteq \mathbb{R}^m$ with finite, nonzero Gaussian mass, let $\rho_\sigma(\mathcal{L}) = \sum_{\mathbf{x} \in \mathcal{L}} \rho_\sigma(\mathbf{x})$. The discrete Gaussian distribution $\mathcal{D}_{\mathcal{L}, \sigma}$ assigns probability $\rho_\sigma(\mathbf{x}) / \rho_\sigma(\mathcal{L})$ to each $\mathbf{x} \in \mathcal{L}$.

Truncated discrete Gaussians. The distribution $\tilde{\mathcal{D}}_{r, \mathbb{Z}, \sigma}$ samples $x \leftarrow \mathcal{D}_{\mathbb{Z}, \sigma}$ and outputs 0 whenever $|x| > \sqrt{r} \sigma$. It is $\sqrt{r} \sigma$ -bounded. For $\sigma \geq 1$ and $r = \omega(\log \lambda)$, the Gaussian tail is negligible, so $\tilde{\mathcal{D}}_{r, \mathbb{Z}, \sigma} \stackrel{\text{stat}}{\approx} \mathcal{D}_{\mathbb{Z}, \sigma}$. Vector errors are sampled from the corresponding product distribution.

Learning with errors (LWE). The LWE problem was introduced by Regev [Reg09]. It has four parameters, the dimension n , the number of samples m , the modulus q , and the error distribution χ .

Assumption 3.1 (Learning with errors). Let n, m, q be positive integers depending on λ , with $q \geq 2$, and let χ be a noise distribution over $\mathbb{Z}$, reduced modulo q . The (n, m, q, χ) -LWE assumption states that the following distributions are computationally indistinguishable:

$$\left\{ (\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top) : \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{s} \leftarrow \mathbb{Z}_q^n, \quad \mathbf{e} \leftarrow \chi^m \end{array} \right\} \stackrel{\text{comp}}{\approx} \left\{ (\mathbf{A}, \mathbf{u}^\top) : \begin{array}{l} \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \\ \mathbf{u} \leftarrow \mathbb{Z}_q^m \end{array} \right\}.$$

We write $\text{LWE}_{n, q, \chi}$ when this holds for every polynomial sample count m . Regev gives a quantum reduction from worst-case lattice problems for appropriate Gaussian errors [Reg09]. Classical reductions are given in [Pei09, BLP⁺13]. The binary-secret variant instead samples $\mathbf{s} \leftarrow \{0, 1\}^n$; whenever we use this variant, we explicitly assume hardness for its parameters [BLP⁺13].

Lemma 3.2 (Leftover hash lemma for binary randomness). Let $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{R} \leftarrow \{0, 1\}^{m \times k}$, and $\mathbf{U} \leftarrow \mathbb{Z}_q^{n \times k}$ be independent. Then

$$\Delta((\mathbf{A}, \mathbf{A}\mathbf{R}), (\mathbf{A}, \mathbf{U})) \leq \frac{k}{2} \sqrt{\frac{q^n}{2^m}}.$$

In particular, $m \geq n \log q + 2\lambda$ gives distance at most $k2^{-\lambda-1}$.

Proof. For distinct binary columns $\mathbf{r}, \mathbf{r}'$, their difference has an entry equal to 1 or -1 . Thus $\Pr_{\mathbf{A}}[\mathbf{A}\mathbf{r} = \mathbf{A}\mathbf{r}'] = q^{-n}$, so this is a universal hash family. The leftover hash lemma gives the bound for one column; a hybrid over the k independent columns gives the stated bound. This is the statistical step used in GSW encryption [MW16]. $\square$

Next we recall the binary gadget decomposition and its matrix form [MP12, MW16] over fields.

Definition 3.3 (Gadget decomposition over fields). For a prime q , let $\mathbf{g} = (1, 2, \dots, 2^{\log q - 1})^\top \in \mathbb{Z}_q^{\log q}$. The map $g^{-1} : \mathbb{Z}_q \rightarrow \{0, 1\}^{\log q}$ returns the binary digits of the representative in $\{0, \dots, q-1\}$, least significant first. It satisfies

$$\mathbf{g}^\top g^{-1}(a) = a \bmod q, \quad g^{-1}(0) = \mathbf{0}.$$

For $m \geq n \log q$, the padded gadget is $\mathbf{G}_{n,m} = [I_n \otimes \mathbf{g}^\top \mid 0_{n \times (m - n \log q)}] \in \mathbb{Z}_q^{n \times m}$. For $\mathbf{x} = (x_1, \dots, x_n)^\top$, its block structure and decomposition are

$$\mathbf{G}_{n,m} = \left[\begin{array}{ccc|c} \mathbf{g}^\top & \cdots & 0 & \\ \vdots & \ddots & \vdots & \\ 0 & \cdots & \mathbf{g}^\top & \end{array} \middle| 0_{n \times (m - n \log q)} \right], \quad \mathbf{G}_{n,m}^{-1}(\mathbf{x}) = \begin{pmatrix} g^{-1}(x_1) \\ \vdots \\ g^{-1}(x_n) \\ \mathbf{0}_{m - n \log q} \end{pmatrix} \in \{0, 1\}^m.$$

For a matrix $\mathbf{X} \in \mathbb{Z}_q^{n \times r}$, apply this map to each column, obtaining $\mathbf{G}_{n,m}^{-1}(\mathbf{X}) \in \{0, 1\}^{m \times r}$. Then

$$\mathbf{G}_{n,m} \mathbf{G}_{n,m}^{-1}(\mathbf{X}) = \mathbf{X} \bmod q.$$

The superscript -1 denotes this fixed decomposition map, not a matrix inverse. We omit the subscripts of $\mathbf{G}$ when its dimensions are clear.

Polynomial rings. Let n be a power of two. In the ring setting, n denotes the ring degree, and for an integer modulus $q \geq 2$, we write

$$R = \mathbb{Z}[X]/(X^n + 1), \quad R_q = R/qR \cong \mathbb{Z}_q[X]/(X^n + 1).$$

Every $f \in R$ has a unique representative $f = \sum_{i=0}^{n-1} f_i X^i$ with $f_i \in \mathbb{Z}$. Its coefficient norm is $\|f\|_\infty = \max_{0 \leq i < n} |f_i|$. For $f \in R_q$, we use the centered coefficient representatives in $(-q/2, q/2]$. Multiplication is modulo $X^n + 1$, and for $a, b \in R$,

$$\|ab\|_\infty \leq \delta_R \|a\|_\infty \|b\|_\infty, \quad \delta_R = n. \tag{11}$$

For a vector of ring elements, $\|\mathbf{x}\|_\infty = \max_i \|x_i\|_\infty$. Hence

$$\|\langle \mathbf{x}, \mathbf{y} \rangle\|_\infty \leq k \delta_R \|\mathbf{x}\|_\infty \|\mathbf{y}\|_\infty \quad \text{for } \mathbf{x}, \mathbf{y} \in R^k. \tag{12}$$

All ring scaling and rounding are coefficient-wise.

Ring-LWE with short secrets. Ring-LWE was introduced in [LPR10]. Its normal-form variant samples the secret from a discretized error distribution [LPR13, Lemma 2.24, full version]. We state the short-secret assumption with separate secret and error distributions, as used by our construction.

Assumption 3.4 (Ring-LWE with short secrets). Let n, m, q be positive integers depending on λ , with n a power of two and $q \geq 2$. Let χ and $\mathcal{D}_e$ be efficiently sampleable distributions over R for the secret and error, respectively, whose coefficient norms are bounded by B_χ and B_e except with negligible probability. The $(n, m, q, \chi, \mathcal{D}_e)$ -RLWE assumption states that the following distributions are computationally indistinguishable:

$$\left\{ (\mathbf{a}, s\mathbf{a} + \mathbf{e}) : \begin{array}{l} \mathbf{a} \leftarrow R_q^m, \\ s \leftarrow \chi, \quad \mathbf{e} \leftarrow \mathcal{D}_e^m \end{array} \right\} \stackrel{comp}{\approx} \left\{ (\mathbf{a}, \mathbf{z}) : \begin{array}{l} \mathbf{a} \leftarrow R_q^m, \\ \mathbf{z} \leftarrow R_q^m \end{array} \right\}.$$

All sampling is independent, the same secret s is used in all m coordinates, and arithmetic is over R_q .

We write $\text{RLWE}_{n,q,\chi,\mathcal{D}_e}$ when this holds for every polynomial sample count m . The construction specifies the secret and error distributions and their bounds. Its additional circular-security requirement is stated with the auxiliary keys that require it.

Definition 3.5 (Gadget decomposition over rings). A gadget is an efficiently computable deterministic map $h : R_q \rightarrow R^k$ and a public vector $\mathbf{g} \in R_q^k$ such that, for every $x \in R_q$,

$$\langle h(x), \mathbf{g} \rangle = x \pmod{q}, \quad \|h(x)\|_\infty \leq B_h.$$

For $\mathbf{y} \in R_q^k$, the external product $x \boxtimes \mathbf{y}$ denotes $\langle h(x), \mathbf{y} \rangle \pmod{q}$.

Ordinary base decomposition applies the scalar digit map to each coefficient. For base $b \geq 2$, it has $k = \lceil \log_b q \rceil$, $\mathbf{g} = (1, b, \dots, b^{k-1})^\top$, and $B_h = b - 1$.

3.2 Multi-Key FHE

Here we define leveled multi-key fully homomorphic encryption for N users, with public key aggregation and ciphertext preparation. Each user independently generates a key pair and encrypts its messages. Key aggregation produces separate keys for preparing these ciphertexts and evaluating a circuit on the prepared inputs.

Syntax. A multi-key (leveled) fully homomorphic encryption scheme for message space $\mathcal{M}$ consists of the following polynomial-time algorithms.

Setup($1^\lambda, 1^d, 1^N$) $\rightarrow$ PP. The probabilistic setup algorithm takes as input a security parameter 1^λ , a maximum circuit depth d , and the number of users N , both in unary, and outputs the public parameters PP.

KeyGen(PP, i) $\rightarrow$ (PK $_i$, SK $_i$). The probabilistic key generation algorithm takes as input the public parameters PP and an index $i \in [N]$, and outputs a public and secret key pair (PK $_i$, SK $_i$).

Enc(PP, PK $_i$, μ) $\rightarrow$ CT. The probabilistic encryption algorithm takes as input the public parameters PP, a public key PK $_i$, and a message $\mu \in \mathcal{M}$, and outputs a ciphertext CT.

$\text{KeyAgg}(\text{PP}, (\text{PK}_i)_{i \in [N]}) \rightarrow (\text{PrepKey}, \text{EvalKey})$. The public deterministic key aggregation algorithm takes as input the public parameters PP and a set of public keys $(\text{PK}_i)_{i \in [N]}$, and outputs a preparation key PrepKey and an evaluation key EvalKey .

$\text{CTPrep}(\text{PP}, \text{PrepKey}, \text{CT}) \rightarrow \text{CT}^{\text{prep}}$. The public deterministic ciphertext preparation algorithm takes as input the public parameters PP , a preparation key PrepKey , and a ciphertext CT , and outputs a prepared ciphertext CT^{prep} .

$\text{Eval}(\text{PP}, \text{EvalKey}, C, (\text{CT}_i^{\text{prep}})_{i \in [N]}) \rightarrow \text{CT}_C$. The public deterministic evaluation algorithm takes as input the public parameters PP , an evaluation key EvalKey , a circuit C of depth at most d , and prepared ciphertexts $(\text{CT}_i^{\text{prep}})_{i \in [N]}$, and outputs an evaluated ciphertext CT_C .

$\text{Dec}(\text{PP}, (\text{SK}_i)_{i \in [N]}, \text{CT}_C) \rightarrow \mu$. The decryption algorithm takes as input the public parameters PP , a set of secret keys $(\text{SK}_i)_{i \in [N]}$, and an evaluated ciphertext CT_C , and outputs a message μ .

Definition 3.6 (MKFHE). A multi-key fully homomorphic encryption scheme $\text{MKFHE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{KeyAgg}, \text{CTPrep}, \text{Eval}, \text{Dec})$ for message space $\mathcal{M}$ is required to satisfy the following properties.

Correctness. For any polynomial functions $N = N(\lambda)$ and $d = d(\lambda)$, there exists a negligible function $\text{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, messages $\mu_i \in \mathcal{M}$ for $i \in [N]$, supported N -ary circuit C of depth at most d , $\text{PP} \leftarrow \text{Setup}(1^\lambda, 1^d, 1^N)$, all $(\text{PK}_i, \text{SK}_i) \leftarrow \text{KeyGen}(\text{PP}, i)$, all $\text{CT}_i \leftarrow \text{Enc}(\text{PP}, \text{PK}_i, \mu_i)$, $(\text{PrepKey}, \text{EvalKey}) = \text{KeyAgg}(\text{PP}, (\text{PK}_i)_{i \in [N]})$, all $\text{CT}_i^{\text{prep}} = \text{CTPrep}(\text{PP}, \text{PrepKey}, \text{CT}_i)$, and $\text{CT}_C = \text{Eval}(\text{PP}, \text{EvalKey}, C, (\text{CT}_i^{\text{prep}})_{i \in [N]})$, we have

$$\Pr[\text{Dec}(\text{PP}, (\text{SK}_i)_{i \in [N]}, \text{CT}_C) = C((\mu_i)_{i \in [N]})] \geq 1 - \text{negl}(\lambda).$$

Semantic Security. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr \left[\begin{array}{l} (1^d, 1^N, i) \leftarrow \mathcal{A}(1^\lambda), \quad i \in [N], \\ b \leftarrow \{0, 1\}, \quad \text{PP} \leftarrow \text{Setup}(1^\lambda, 1^d, 1^N), \\ (\text{PK}_i, \text{SK}_i) \leftarrow \text{KeyGen}(\text{PP}, i), \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}(\text{PP}, \text{PK}_i), \quad |\mu_0| = |\mu_1|, \\ \text{CT}_b \leftarrow \text{Enc}(\text{PP}, \text{PK}_i, \mu_b) \end{array} : b = \mathcal{A}(\text{CT}_b) \right] \leq \frac{1}{2} + \text{negl}(\lambda).$$

Here $\mu_0, \mu_1 \in \mathcal{M}$, and PK_i includes all public key material generated by KeyGen .

Compactness. There exists a fixed polynomial $\text{poly}(\cdot, \cdot, \cdot, \cdot)$ such that for every choice of parameters, keys, messages, circuit, and ciphertexts in the correctness property, the evaluated ciphertext satisfies $|\text{CT}_C| \leq \text{poly}(\lambda, d, N, |C((\mu_i)_{i \in [N]}|))$, independently of the size of C .

Additional Properties. We consider two additional efficiency properties concerning the ciphertext size and running time of evaluation after key aggregation and ciphertext preparation.

Definition 3.7 (Full Succinctness [AG26]). An MKFHE scheme is said to be fully succinct if there exists a fixed polynomial $\text{poly}(\cdot, \cdot, \cdot, \cdot)$ such that the evaluated ciphertext size in the compactness property satisfies $|\text{CT}_C| \leq \text{poly}(\lambda, d, \log N, |C((\mu_i)_{i \in [N]}|))$.

Definition 3.8 (Optimal Evaluation). An MKFHE scheme is said to have optimal evaluation if there exists a fixed polynomial $\text{poly}(\cdot, \cdot, \cdot)$ such that $|\text{EvalKey}|$ and the running time of each gate evaluation are both bounded by $\text{poly}(\lambda, d, \log N)$, and the total evaluation running time is bounded by $|C| \cdot \text{poly}(\lambda, d, \log N)$ for every choice of parameters, keys, circuit, and prepared ciphertexts in the correctness property. Here $|C|$ counts the input, internal, and output gates of C . The time spent on key aggregation and ciphertext preparation may still grow polynomially with N .

Remark 3.9 (Dynamic choice of users). We use $[N]$ for all participating users to simplify the notation. However, we consider constructions that support a dynamic choice of these users. Namely, setup may fix a bound $N_{\max}$ on the number of users, with key generation and parameter selection using this bound. Users generate their keys and encrypt independently, before a participating set $S \subseteq [N_{\max}]$ or a circuit is chosen. For a computation involving $N = |S|$ users, we restrict aggregation and preparation to S and relabel its users as $[N]$. This relabeling applies consistently to their existing indexed public masks and key entries, preserving the original correspondence between users and slots. It requires no new setup or key generation. The same fresh ciphertext can be prepared for different participating sets, each with its own pair $(\text{PrepKey}, \text{EvalKey})$. Within one evaluation, all prepared inputs use the same participating set. This gives the dynamic choice of users and circuits needed for MKFHE while keeping only N in the definition and constructions.

3.2.1 Single-Round Distributed Decryption

We also consider MKFHE with single-round distributed decryption, following [MW16].

Syntax. An MKFHE scheme with single-round distributed decryption consists of the following additional polynomial-time algorithms.

PartDec(PP, CT_C , $(\text{PK}_j)_{j \in [N]}$, i , SK_i) $\rightarrow p_i$. The probabilistic partial decryption algorithm takes as input the public parameters PP, an evaluated ciphertext CT_C , a set of public keys $(\text{PK}_j)_{j \in [N]}$, an index $i \in [N]$, and a secret key SK_i , and outputs a partial decryption p_i .

FinDec(PP, $(p_i)_{i \in [N]}$) $\rightarrow \mu$. The final decryption algorithm takes as input the public parameters PP and a set of partial decryptions $(p_i)_{i \in [N]}$, and outputs a message μ .

Definition 3.10 (MKFHE with single-round distributed decryption). An MKFHE scheme with single-round distributed decryption is required to satisfy Definition 3.6 and the following additional properties.

Correctness of Partial Decryption. For every choice of parameters, keys, messages, circuit, and ciphertexts in the correctness property of Definition 3.6, let each p_i be computed independently by **PartDec** as specified above. Then

$$\Pr[\text{FinDec}(\text{PP}, (p_i)_{i \in [N]}) = C((\mu_i)_{i \in [N]})] \geq 1 - \text{negl}(\lambda).$$

Simulatability of Partial Decryption. There exists a PPT simulator $\mathcal{S}$ such that for every choice of parameters, keys, messages, circuit, and ciphertexts in the correctness property of Definition 3.6, and every $i \in [N]$, letting $\mu = C((\mu_j)_{j \in [N]})$, we have $p_i \stackrel{\text{stat}}{\approx} p'_i$, where

$$\begin{aligned} p_i &\leftarrow \text{PartDec}(\text{PP}, \text{CT}_C, (\text{PK}_j)_{j \in [N]}, i, \text{SK}_i), \\ p'_i &\leftarrow \mathcal{S}(\text{PP}, \mu, \text{CT}_C, C, i, (\text{SK}_j)_{j \in [N] \setminus \{i\}}). \end{aligned}$$

The statistical distance is negligible in λ . The randomness in these distributions is only over the coins of PartDec and $\mathcal{S}$; all other values are fixed.

4 Fast GSW-Based Multi-Key FHE from Plain LWE

In this section, we construct fast multi-key FHE from plain LWE using MW expansion and GSW evaluation. Before giving the construction, we define the vector-compression algorithm and the column-selection matrix used by ciphertext preparation.

We work over $\mathbb{Z}_q$ for an odd prime q , let $\Delta = (q-1)/2$, and use $\mathbf{G} = \mathbf{G}_{n,m}$ from Definition 3.3. The source public keys use a common matrix $\mathbf{B} \in \mathbb{Z}_q^{(n-1) \times m}$. For secrets $\mathbf{s}_i \in \mathbb{Z}_q^{n-1}$, define $\mathbf{t}_i = (-\mathbf{s}_i^\top, 1)^\top$, $\hat{\mathbf{t}} = (\mathbf{t}_1^\top, \dots, \mathbf{t}_N^\top)^\top$, $\tilde{\mathbf{s}} = \sum_{i=1}^N \mathbf{s}_i$, and $\tilde{\mathbf{t}} = (-\tilde{\mathbf{s}}^\top, 1)^\top$. Our preparation produces a ciphertext matrix $\mathbf{CT} \in \mathbb{Z}_q^{n \times m}$ satisfying

$$\tilde{\mathbf{t}}^\top \mathbf{CT} = \mu \tilde{\mathbf{t}}^\top \mathbf{G} + \mathbf{e}_{\mathbf{CT}}^\top. \quad (13)$$

4.1 Compressing an Inner-Product Ciphertext

For an input $\mathbf{c} \in \mathbb{Z}_q^{nN}$, the compression algorithm returns $(\mathbf{u}^\top, v)^\top \in \mathbb{Z}_q^n$ whose phase under $\tilde{\mathbf{t}}$ equals $\hat{\mathbf{t}}^\top \mathbf{c}$ up to the error bounded in Lemma 4.1. Write the input in N blocks, $\mathbf{c} = (\mathbf{c}_1^\top, \dots, \mathbf{c}_N^\top)^\top$, where $\mathbf{c}_j = (c_{j,1}, \dots, c_{j,n})^\top$. Here j indexes the row blocks of the input vector. Its phase is

$$\hat{\mathbf{t}}^\top \mathbf{c} = \sum_{j=1}^N c_{j,n} - \sum_{j=1}^N \sum_{k=1}^{n-1} s_{j,k} c_{j,k}. \quad (14)$$

Public compression material. Setup samples independent uniform columns $\mathbf{a}_{j,k,\alpha} \in \mathbb{Z}_q^{n-1}$, independently of $\mathbf{B}$, for every $j \in [N]$, $k \in [n-1]$, and $0 \leq \alpha < \log q$. User i publishes

$$w_{i,j,k,\alpha} = \langle \mathbf{s}_i, \mathbf{a}_{j,k,\alpha} \rangle + \delta_{i,j} 2^\alpha s_{i,k} + \varepsilon_{i,j,k,\alpha}. \quad (15)$$

The independent errors have distribution χ_{cmp} and bound B_{cmp} . Key aggregation for N users computes

$$W_{j,k,\alpha} = \sum_{i=1}^N w_{i,j,k,\alpha} = \langle \tilde{\mathbf{s}}, \mathbf{a}_{j,k,\alpha} \rangle + 2^\alpha s_{j,k} + \varepsilon_{j,k,\alpha}, \quad \varepsilon_{j,k,\alpha} = \sum_{i=1}^N \varepsilon_{i,j,k,\alpha}. \quad (16)$$

The aggregated public pairs satisfy $\tilde{\mathbf{t}}^\top \begin{pmatrix} \mathbf{a}_{j,k,\alpha} \\ W_{j,k,\alpha} \end{pmatrix} = 2^\alpha s_{j,k} + \varepsilon_{j,k,\alpha}$.

Compressing the ciphertext. Write $g^{-1}(c_{j,k}) = (\beta_{j,k,0}, \dots, \beta_{j,k,\log q-1})^\top$ for $j \in [N]$ and $k \in [n-1]$, so $c_{j,k} = \sum_{\alpha=0}^{\log q-1} 2^\alpha \beta_{j,k,\alpha} \pmod{q}$. Using the aggregated public pairs, compute

$$\begin{aligned} \text{Compress}(\mathbf{c}) &= \begin{pmatrix} \mathbf{0}_{n-1} \\ \sum_{j=1}^N c_{j,n} \end{pmatrix} - \sum_{j,k,\alpha} \beta_{j,k,\alpha} \begin{pmatrix} \mathbf{a}_{j,k,\alpha} \\ W_{j,k,\alpha} \end{pmatrix} \\ &= \begin{pmatrix} -\sum_{j,k,\alpha} \beta_{j,k,\alpha} \mathbf{a}_{j,k,\alpha} \\ \sum_{j=1}^N c_{j,n} - \sum_{j,k,\alpha} \beta_{j,k,\alpha} W_{j,k,\alpha} \end{pmatrix} = \begin{pmatrix} \mathbf{u} \\ v \end{pmatrix} \in \mathbb{Z}_q^n. \end{aligned} \quad (17)$$

Here the triple sums range over $j \in [N]$, $k \in [n-1]$, and $0 \leq \alpha < \log q$. The first vector supplies the public sum $\sum_j c_{j,n}$ in Eq. (14). The subtracted columns supply its secret-dependent terms through their phases under $\tilde{\mathbf{t}}$.

Lemma 4.1 (Ciphertext compression). For every $\mathbf{c} \in \mathbb{Z}_q^{nN}$, it holds that:

$$\tilde{\mathbf{t}}^\top \text{Compress}(\mathbf{c}) = \hat{\mathbf{t}}^\top \mathbf{c} - \sum_{j,k,\alpha} \beta_{j,k,\alpha} \varepsilon_{j,k,\alpha},$$

where the added compression error is at most $N^2(n-1) \log q B_{\text{cmp}}$, and the compression running time (after preprocessing) is $O(Nn^2 \log q)$.

Proof.

$$\begin{aligned} \tilde{\mathbf{t}}^\top \text{Compress}(\mathbf{c}) &= v - \langle \tilde{\mathbf{s}}, \mathbf{u} \rangle \\ &= \sum_{j=1}^N c_{j,n} - \sum_{j,k,\alpha} \beta_{j,k,\alpha} (W_{j,k,\alpha} - \langle \tilde{\mathbf{s}}, \mathbf{a}_{j,k,\alpha} \rangle) \quad // \text{ by Eq. (17)} \\ &= \sum_{j=1}^N c_{j,n} - \sum_{j,k,\alpha} \beta_{j,k,\alpha} (2^\alpha s_{j,k} + \varepsilon_{j,k,\alpha}) \quad // \text{ by Eq. (16)} \\ &= \sum_{j=1}^N c_{j,n} - \sum_{j=1}^N \sum_{k=1}^{n-1} s_{j,k} \underbrace{\left(\sum_{\alpha=0}^{\log q - 1} 2^\alpha \beta_{j,k,\alpha} \right)}_{c_{j,k}} - \sum_{j,k,\alpha} \beta_{j,k,\alpha} \varepsilon_{j,k,\alpha} \\ &= \hat{\mathbf{t}}^\top \mathbf{c} - \sum_{j,k,\alpha} \beta_{j,k,\alpha} \varepsilon_{j,k,\alpha} \quad // \text{ by Eq. (14)}, \end{aligned}$$

where using Eqs. (15) and (16) the compression error is bounded by

$$\left| \sum_{j,k,\alpha} \beta_{j,k,\alpha} \varepsilon_{j,k,\alpha} \right| \leq N(n-1) \log q \max_{j,k,\alpha} |\varepsilon_{j,k,\alpha}| \leq N^2(n-1) \log q B_{\text{cmp}},$$

and using Eq. (17), compression requires $O(Nn^2 \log q)$ additions at most. $\square$

4.2 Column Selection and Ciphertext Preparation

Let $\hat{\mathbf{C}} \in \mathbb{Z}_q^{nN \times mN}$ be an MW expansion and let $\hat{\mathbf{G}}_N = I_N \otimes \mathbf{G}$. Its defining relation is $\hat{\mathbf{t}}^\top \hat{\mathbf{C}} = \mu \hat{\mathbf{t}}^\top \hat{\mathbf{G}}_N + \mathbf{e}^\top$. The expansion and its block structure are recalled in Section A.1. Index each nonpadding column by (j, k, α) , where $j \in [N]$ is the user's column group, $k \in [n]$ is the key coordinate, and $0 \leq \alpha < \log q$. Write $\hat{\mathbf{c}}_{j,k,\alpha}$ for column $(j-1)m + (k-1)\log q + \alpha + 1$, and index the entries of $\mathbf{e}$ in the same way.

The column-selection matrix. For every $k \in [n]$ and $0 \leq \alpha < \log q$, form $\mathbf{y}_{k,\alpha} = \sum_{j=1}^N \hat{\mathbf{c}}_{j,k,\alpha}$ if $k < n$, and $\mathbf{y}_{k,\alpha} = \hat{\mathbf{c}}_{1,n,\alpha}$ if $k = n$. Then construct $\mathbf{Y} \in \mathbb{Z}_q^{nN \times m}$ by putting these columns in the same coordinate-and-power order as $\mathbf{G}$, and appending $m - n \log q$ zero columns. For $k < n$, the column has phase $-\mu 2^\alpha \tilde{s}_k + \sum_j e_{j,k,\alpha}$, for $k = n$, it has phase $\mu 2^\alpha + e_{1,n,\alpha}$, and the padding columns have phase zero. These are the desired GSW values, but they are still obtained under the old key $\hat{\mathbf{t}}$.

We write this operation as $\mathbf{Y} = \hat{\mathbf{C}}\mathbf{H}$. Here $\mathbf{H} \in \{0,1\}^{mN \times m}$ simply records which source columns are added. Its rows correspond to columns of $\hat{\mathbf{C}}$, and its columns correspond to columns of $\mathbf{Y}$. Specifically, in the output column (k, α) , place a 1 at each source position (j, k, α) if $k < n$, or only at $(1, n, \alpha)$ if $k = n$; all other entries, including padding, are zero. Note that every column of $\mathbf{H}$ has at most N ones. Right multiplication by $\mathbf{H}$ therefore performs the additions just defined. This matrix is public and depends only on the column indices. In particular,

$$\hat{\mathbf{t}}^\top \hat{\mathbf{G}}_N \mathbf{H} = \tilde{\mathbf{t}}^\top \mathbf{G}. \quad (18)$$

Shortening each column. Apply Compress to every column of $\mathbf{Y}$, including its zero padding columns, which remain zero. By Lemma 4.1, the output has the same phase up to compression error, now under $\tilde{\mathbf{t}}$, and has only n coordinates. Arranging the outputs as columns gives

$$\mathbf{CT} = \text{Compress}^{\text{col}}(\hat{\mathbf{C}}\mathbf{H}) \in \mathbb{Z}_q^{n \times m}. \quad (19)$$

Here the superscript means columnwise application. By Lemma 4.1 and the preceding identities,

$$\begin{aligned} \tilde{\mathbf{t}}^\top \mathbf{CT} &= \hat{\mathbf{t}}^\top \hat{\mathbf{C}}\mathbf{H} + \mathbf{e}_{\text{cmp}}^\top \\ &= \mu \hat{\mathbf{t}}^\top \hat{\mathbf{G}}_N \mathbf{H} + \mathbf{e}^\top \mathbf{H} + \mathbf{e}_{\text{cmp}}^\top = \mu \tilde{\mathbf{t}}^\top \mathbf{G} + \mathbf{e}^\top \mathbf{H} + \mathbf{e}_{\text{cmp}}^\top. \end{aligned} \quad (20)$$

This is the required GSW relation. The column $\mathbf{H}^\top \mathbf{e}$ collects the errors from adding source columns, while $\mathbf{e}_{\text{cmp}}$ collects the compression errors.

Computing the selected columns. A fresh MW expansion consists of N diagonal copies of the same n -by- m matrix and $N - 1$ correction blocks in the input owner's block row, as shown in Fig. 2. Compute these blocks one at a time and add their selected columns to $\mathbf{Y}$. This forms $\hat{\mathbf{C}}\mathbf{H}$ without storing $\hat{\mathbf{C}}$.

4.3 Construction

Here we construct $\text{MKFHE} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{KeyAgg}, \text{CTPrep}, \text{Eval}, \text{Dec})$ for depth-bounded Boolean computations on encrypted bits, with fast evaluation and small ciphertexts. We use MW's fresh encryption and expansion [MW16], followed by the compression above. The parameter settings and LWE assumptions are given after the construction.

Construction 4.2. Let $d = d(\lambda)$ be the supported depth and $N = N(\lambda)$ the number of users. All matrix and vector computations are over $\mathbb{Z}_q$. We use $\mathbf{G} = \mathbf{G}_{n,m}$ from Definition 3.3 and Compress from Eq. (17), with the masks in PP and the compression entries CK contained in PrepKey.

$\text{Setup}(1^\lambda, 1^d, 1^N) \rightarrow \text{PP}$. Do the following:

1. Choose $q, n, m, \chi, B_\chi, \chi_{\text{cmp}}, B_{\text{cmp}}$ according to the parameter settings below, and let $\Delta = (q - 1)/2$.
2. Sample the common MW matrix $\mathbf{B} \leftarrow \mathbb{Z}_q^{(n-1) \times m}$ and set $\text{pp} = (q, n, m, \chi, B_\chi, \mathbf{B})$.
3. For every $j \in [N]$, $k \in [n - 1]$, and $\alpha \in \{0, \dots, \log q - 1\}$, sample $\mathbf{a}_{j,k,\alpha} \leftarrow \mathbb{Z}_q^{n-1}$.

Output $\text{PP} = (\text{pp}, d, N, \chi_{\text{cmp}}, B_{\text{cmp}}, (\mathbf{a}_{j,k,\alpha})_{j,k,\alpha})$.

KeyGen(PP, i) $\rightarrow (\text{PK}_i, \text{SK}_i)$. Do the following:

1. Sample $\mathbf{s}_i \leftarrow \mathbb{Z}_q^{n-1}$ and $\mathbf{e}_i \leftarrow \chi^m$. For every $j \in [N]$, $k \in [n - 1]$, and $0 \leq \alpha < \log q$, sample $\varepsilon_{i,j,k,\alpha} \leftarrow \chi_{\text{cmp}}$.
2. Compute $\mathbf{b}_i = \mathbf{B}^\top \mathbf{s}_i + \mathbf{e}_i$ and form the underlying MW key pair $\text{pk}_i = \mathbf{A}_i = [\mathbf{B}; \mathbf{b}_i^\top] \in \mathbb{Z}_q^{n \times m}$ and $\text{sk}_i = \mathbf{t}_i = (-\mathbf{s}_i^\top, 1)^\top \in \mathbb{Z}_q^n$.
3. For every (j, k, α) in the preceding ranges, compute $w_{i,j,k,\alpha} = \langle \mathbf{s}_i, \mathbf{a}_{j,k,\alpha} \rangle + \delta_{i,j} 2^\alpha s_{i,k} + \varepsilon_{i,j,k,\alpha}$.

Output $\text{PK}_i = (i, \text{pk}_i, (w_{i,j,k,\alpha})_{j,k,\alpha})$ and $\text{SK}_i = (i, \text{sk}_i)$.

Enc($\text{PP}, \text{PK}_i, \mu$) $\rightarrow \text{CT}$. Run the ordinary MW encryption to obtain $(U, \mathbf{C})$, where U is the expansion material. Output $\text{CT} = (i, (U, \mathbf{C}))$.

KeyAgg($\text{PP}, (\text{PK}_i)_{i=1}^N$) $\rightarrow (\text{PrepKey}, \text{EvalKey})$. Parse the N public keys and compute $W_{j,k,\alpha} = \sum_{i=1}^N w_{i,j,k,\alpha}$ for every $j \in [N]$, $k \in [n - 1]$, and $0 \leq \alpha < \log q$. Let $\text{CK} = (W_{j,k,\alpha})_{j,k,\alpha}$. Output $\text{PrepKey} = ((\text{pk}_j)_{j \in [N]}, \text{CK})$ and the empty evaluation key $\text{EvalKey} = \perp$.

CTPrep($\text{PP}, \text{PrepKey}, \text{CT}$) $\rightarrow \text{CT}^{\text{prep}}$. Parse PrepKey and CT , then using U and $(\text{pk}_j)_{j \in [N]}$, run MW's public expansion on the ciphertext to obtain $\hat{\mathbf{C}} \in \mathbb{Z}_q^{nN \times mN}$, as recalled in Section A.1. Compute $\mathbf{Y} = \hat{\mathbf{C}}\mathbf{H} \in \mathbb{Z}_q^{nN \times m}$ using the public column-selection matrix $\mathbf{H}$ from Section 4.2. Output $\text{CT}^{\text{prep}} = \text{Compress}^{\text{col}}(\mathbf{Y}) \in \mathbb{Z}_q^{n \times m}$, applying Compress to each column as in Eq. (19).

Eval($\text{PP}, \text{EvalKey}, C, (\text{CT}_i^{\text{prep}})_{i \in [N]}$) $\rightarrow \text{CT}_C$. Run the ordinary GSW evaluation for the circuit C on the prepared ciphertexts to compute CT_C .

Dec($\text{PP}, (\text{SK}_i)_{i=1}^N, \text{CT}_C$) $\rightarrow \mu$. Parse $\text{sk}_i = (-\mathbf{s}_i^\top, 1)^\top$ from each SK_i , compute $\tilde{\mathbf{s}} = \sum_{i=1}^N \mathbf{s}_i$, and let $\tilde{\mathbf{t}} = (-\tilde{\mathbf{s}}^\top, 1)^\top \in \mathbb{Z}_q^n$. Next compute

$$\mathbf{c}_{\text{out}} = \text{CT}_C \mathbf{G}^{-1}(\Delta \mathbf{e}_n) \in \mathbb{Z}_q^n, \quad (21)$$

and let $z = \tilde{\mathbf{t}}^\top \mathbf{c}_{\text{out}}$. Finally output $\mu = \lceil z \rceil_2$.

For multiple inputs per user, prepare each input separately and run the same **Eval** on the prepared matrices. Public constants use the matrices $\mu \mathbf{G}$.

Parameter settings. Let $d, N \leq \text{poly}(\lambda)$. Take $n = \lceil (\lambda + d + \log(N + 1))^2 \rceil$, and let $\chi = \tilde{\mathcal{D}}_{\lambda, \mathbb{Z}, n}$ be the truncated discrete Gaussian from Section 3.1, with bound $B_\chi = n\sqrt{\lambda}$. Use $\chi_{\text{cmp}} = \chi$ and $B_{\text{cmp}} = B_\chi$. Choose an odd prime q and set $m = n \log q + 2\lambda$, where the extra 2λ columns give the entropy margin for Lemma 3.2. Secrets are uniform in $\mathbb{Z}_q^{n-1}$.

MW's fresh expansion has error at most $B_{\text{MW}} = (m^4 + m)B_\chi$. The analysis below bounds the preparation error by $B_{\text{in}}(N) = NB_{\text{MW}} + N^2(n-1)\log q B_\chi$, and ordinary GSW evaluation increases this bound by a factor of at most $m+1$ per level. Hence choose q large enough that

$$(\log q)(m+1)^d B_{\text{in}}(N) < q/4 - 1/2. \quad (22)$$

Since $B_{\text{in}}(N) \leq 3N^2m^4B_\chi$, this is achieved with $\log q = O((d+1)\log(\lambda+d+\log(N+1))+\log(N+1))$, taking a sufficiently large constant. Thus $q/B_\chi = 2^{\text{poly}(\log \lambda, d, \log N)}$. Security assumes $\text{LWE}_{n-1, q, \chi}$ with $m + N(n-1)\log q$ samples under each secret. The joint public-key and compression-key distribution is covered by the linear-KDM reduction in semantic security. The Gaussian truncation changes this distribution by negligible statistical distance.

Theorem 4.3. Assuming LWE with modulus-to-noise ratio $q/B_\chi = 2^{\text{poly}(\log \lambda, d, \log N)}$, Construction 4.2 is a leveled MKFHE for depth- d circuits satisfying full succinctness and optimal evaluation as defined in Definitions 3.7 and 3.8.

4.4 Analysis

Correctness. Fix honestly generated keys for the N users and a supported depth- d circuit C . Let $\hat{\mathbf{t}} = (\mathbf{t}_1^\top, \dots, \mathbf{t}_N^\top)^\top$ and $\tilde{\mathbf{t}} = (-\sum_i \mathbf{s}_i^\top, 1)^\top$. For an input bit μ encrypted by any user $i \in [N]$, let $\hat{\mathbf{C}}$ be its MW expansion and CT_0 its prepared ciphertext. MW's expansion correctness [MW16] gives $\hat{\mathbf{t}}^\top \hat{\mathbf{C}} = \mu \tilde{\mathbf{t}}^\top \hat{\mathbf{G}}_N + \hat{\mathbf{e}}^\top$, where $\|\hat{\mathbf{e}}\|_\infty \leq B_{\text{MW}}$. Indeed, the owner block has error at most mB_χ , and every other block has error at most $(m^4 + m)B_\chi$. Let $\mathbf{Y} = \hat{\mathbf{C}}\mathbf{H}$. Applying Lemma 4.1 to its columns gives

$$\begin{aligned} \tilde{\mathbf{t}}^\top \text{CT}_0 &= \hat{\mathbf{t}}^\top \mathbf{Y} + \mathbf{e}_{\text{cmp}}^\top && // \text{ by Lemma 4.1} \\ &= \mu \tilde{\mathbf{t}}^\top \hat{\mathbf{G}}_N \mathbf{H} + \hat{\mathbf{e}}^\top \mathbf{H} + \mathbf{e}_{\text{cmp}}^\top && // \text{ by MW expansion} \\ &= \mu \tilde{\mathbf{t}}^\top \mathbf{G} + \hat{\mathbf{e}}^\top \mathbf{H} + \mathbf{e}_{\text{cmp}}^\top && // \text{ by Eq. (18).} \end{aligned}$$

Here $\mathbf{e}_{\text{cmp}} \in \mathbb{Z}^m$ collects the compression errors, with $\|\mathbf{e}_{\text{cmp}}\|_\infty \leq N^2(n-1)\log q B_{\text{cmp}}$. Every column of $\mathbf{H}$ has at most N ones. Thus CT_0 satisfies the GSW relation under $\tilde{\mathbf{t}}$ with error $\mathbf{e}_0 = \mathbf{H}^\top \hat{\mathbf{e}} + \mathbf{e}_{\text{cmp}}$ bounded by

$$\|\mathbf{e}_0\|_\infty \leq N\|\hat{\mathbf{e}}\|_\infty + \|\mathbf{e}_{\text{cmp}}\|_\infty \leq NB_{\text{MW}} + N^2(n-1)\log q B_{\text{cmp}} = B_{\text{in}}(N).$$

This holds for every input, since Lemma 4.1 applies to arbitrary columns, including those depending on the public keys.

All prepared ciphertexts are therefore $B_{\text{in}}(N)$ -noisy GSW ciphertexts under the same key. For two such ciphertexts with bit-valued phase multipliers and error bounds B_1, B_2 , ordinary addition has error at most $B_1 + B_2$, while multiplication has error at most $mB_1 + B_2$, since every column of the gadget decomposition has at most m ones. The multiplication bound determines the error growth in standard GSW Boolean circuit evaluation. Hence, for $\mu = C((\mu_i)_{i \in [N]}) \in \{0, 1\}$, the evaluated ciphertext satisfies $\tilde{\mathbf{t}}^\top \text{CT}_C = \mu \tilde{\mathbf{t}}^\top \mathbf{G} + \mathbf{e}_{\text{out}}^\top$ where $\|\mathbf{e}_{\text{out}}\|_\infty \leq (m+1)^d B_{\text{in}}(N)$. Let $\mathbf{d}_\Delta = \mathbf{G}^{-1}(\Delta \mathbf{e}_n) \in \{0, 1\}^m$, which has at most $\log q$ ones. The phase used by Dec is

$$\begin{aligned} z = \tilde{\mathbf{t}}^\top \mathbf{c}_{\text{out}} &= (\mu \tilde{\mathbf{t}}^\top \mathbf{G} + \mathbf{e}_{\text{out}}^\top) \mathbf{d}_\Delta && // \text{ by Eq. (21)} \\ &= \Delta \mu + \mathbf{e}_{\text{out}}^\top \mathbf{d}_\Delta && // \text{ by } \tilde{t}_n = 1. \end{aligned}$$

Thus, the error $\varepsilon = \mathbf{e}_{\text{out}}^\top \mathbf{d}_\Delta$ satisfies $|\varepsilon| \leq (\log q)(m+1)^d B_{\text{in}}(N) < q/4 - 1/2$ by Eq. (22). Therefore, the scaled rounding $\lceil z \rceil_2$ returns μ .

Compactness and running time. Similar to GSW evaluation, after ciphertext preparation, every addition costs $O(nm)$ and every multiplication costs $O(nm^2)$. During evaluation, all ciphertexts have nm field elements per plaintext bit, which is representable with $nm \log q$ bits. The final decryptable payload $\mathbf{c}_{\text{out}}$ is a vector of n field elements. Since EvalKey is empty and $n, m, \log q$ are polynomial in $\lambda, d, \log N$, these bounds give full succinctness and optimal evaluation.

Key aggregation performs $O(N^2 n \log q)$ additions and outputs PrepKey containing $N(n-1) \log q$ aggregated field elements for compression, in addition to the N source public keys. Although ciphertext preparation is written using $\widehat{\mathbf{C}}$, its fresh MW expansion has only N diagonal copies of $\mathbf{C}$ and $N-1$ correction blocks in the owner's row. Each correction is computed from U and one public key in $O(nm^2 \log q)$ work. MW's linear-combination routine reads only the $\log q$ columns for the constant secret-key coordinate from each of the m^2 helper matrices in U . Their weighted sums form the m correction columns, each of length n . We accumulate the columns selected by $\mathbf{H}$ as these blocks are computed, obtaining $\mathbf{Y} = \widehat{\mathbf{C}}\mathbf{H}$ without materializing the full expanded matrix. Column compression then costs $O(Nn^2 m \log q)$ by Lemma 4.1. Since $m \geq n \log q$, fresh input preparation costs $O(Nnm^2 \log q)$ with random access to U and the preparation key. Sequentially reading all of U adds $\Theta(nm^3)$ work.

Setup publishes $N(n-1)^2 \log q$ compression-mask elements, and each user publishes $N(n-1) \log q$ compression entries in addition to its MW public key. Thus total compression material is quadratic in N , however, aggregation brings the working compression material down to linear.

Semantic security. Semantic security follows the same argument as MW [MW16], except that the public keys additionally contain the compression entries $w_{i,j,k,\alpha} = \langle \mathbf{s}_i, \mathbf{a}_{j,k,\alpha} \rangle + \delta_{i,j} 2^\alpha s_{i,k} + \varepsilon_{i,j,k,\alpha}$.

We prove security of these linear key-dependent messages (KDM) from ordinary LWE using the public-mask translation underlying affine-KDM security [ACPS09]. Fix the challenge user i^* , selected before Setup in Definition 3.6. Given independent LWE challenge samples $(\mathbf{a}'_{j,k,\alpha}, b'_{j,k,\alpha})$ under one unknown secret $\mathbf{s}_{i^*}$, publish

$$\mathbf{a}_{j,k,\alpha} = \mathbf{a}'_{j,k,\alpha} - \delta_{i^*,j} 2^\alpha \mathbf{e}_k, \quad w_{i^*,j,k,\alpha} = b'_{j,k,\alpha},$$

where $\mathbf{e}_k$ is the k -th standard basis vector. When the responses are LWE samples, this gives

$$\begin{aligned} w_{i^*,j,k,\alpha} &= \langle \mathbf{s}_{i^*}, \mathbf{a}'_{j,k,\alpha} \rangle + \varepsilon_{i^*,j,k,\alpha} \\ &= \langle \mathbf{s}_{i^*}, \mathbf{a}_{j,k,\alpha} \rangle + \delta_{i^*,j} 2^\alpha s_{i^*,k} + \varepsilon_{i^*,j,k,\alpha}. \end{aligned} \tag{23}$$

The translations are public and preserve the independent uniform distribution of the masks, including when $j = i^*$. Use m further samples under the same secret for $(\mathbf{B}, \mathbf{b}_{i^*})$. Since $\chi_{\text{cmp}} = \chi$, the resulting joint distribution is the real one. If the challenge responses are uniform, all of $\mathbf{b}_{i^*}$ and the compression entries are instead independent uniform values. Other users' public and secret keys can be generated using their independently sampled secrets and these same public masks. The reduction therefore covers the complete published material, including all slots in Remark 3.9, without a separate circular-security assumption.

We then apply the leftover hash lemma, as in MW, to the GSW encryptions of randomness in U and to the message-carrying matrix $\mathbf{C} = \mathbf{A}_i \mathbf{R} + \mu \mathbf{G}$. This makes the fresh ciphertext statistically

close to a distribution independent of μ , establishing semantic security. Since **KeyAgg**, **CTPrep**, and **Eval** are public computations on these keys and ciphertexts, the same guarantee holds after preparation and evaluation.

4.5 Distributed Decryption

We add single-round distributed decryption in the same way as MW [MW16], by adding smudging noise to each user's contribution to the decryption phase. For this extension, we instantiate Construction 4.2 with a larger modulus q chosen in **Setup**, with $m = n \log q + 2\lambda$ and all error bounds evaluated at this modulus. Let $B_{\text{dec}} = (\log q)(m+1)^d B_{\text{in}}(N)$. Choose q and an integer B_{sm} jointly such that $B_{\text{dec}}/B_{\text{sm}} = \text{negl}(\lambda)$ and $B_{\text{dec}} + NB_{\text{sm}} < q/4 - 1/2$, and include B_{sm} in **PP**. For example, for $\lambda \geq 2$, let $B_{\text{sm}} = \lceil 2^\lambda B_{\text{dec}} \rceil$ and choose $\log q = O(\lambda + (d+1) \log(\lambda + d + \log(N+1)) + \log(N+1))$ with a sufficiently large constant. Thus smudging requires a superpolynomial modulus-to-noise ratio q/B_{sm} . Security assumes **LWE** at this larger modulus.

PartDec(**PP**, $\text{CT}_C, (\text{PK}_j)_{j \in [N]}, i, \text{SK}_i$) $\rightarrow p_i$. Compute $\mathbf{c}_{\text{out}} = \text{CT}_C \mathbf{G}^{-1}(\Delta \mathbf{e}_n)$ as in **Dec**, and write $\mathbf{c}_{\text{out}} = (\mathbf{u}^\top, v)^\top$. Parse $\mathbf{s}_i$ from SK_i , let $\gamma_i = \delta_{i,1}v - \langle \mathbf{s}_i, \mathbf{u} \rangle$, and sample $e_i^{\text{sm}} \leftarrow [-B_{\text{sm}}, B_{\text{sm}}] \cap \mathbb{Z}$ uniformly. Output $p_i = \gamma_i + e_i^{\text{sm}} \pmod{q}$.

FinDec(**PP**, $(p_i)_{i \in [N]}$) $\rightarrow \mu$. Output $\mu = \lceil \sum_{i=1}^N p_i \rceil_2$.

Correctness and simulatability. Let $\mu = C((\mu_i)_{i \in [N]})$. By the correctness argument above, $\sum_i \gamma_i = \tilde{\mathbf{t}}^\top \mathbf{c}_{\text{out}} = \Delta\mu + \varepsilon \pmod{q}$ with $|\varepsilon| \leq B_{\text{dec}}$. Thus $\sum_i p_i = \Delta\mu + \varepsilon + \sum_i e_i^{\text{sm}} \pmod{q}$, whose total error is at most $B_{\text{dec}} + NB_{\text{sm}} < q/4 - 1/2$, so final decryption returns μ . To simulate user i 's partial decryption, compute γ_j for $j \neq i$ using the given secret keys, sample e_i^{sm} as above, and output $p'_i = \Delta\mu - \sum_{j \neq i} \gamma_j + e_i^{\text{sm}} \pmod{q}$. The real share differs only by the shift ε . As in MW, uniform-interval smudging bounds the statistical distance by $B_{\text{dec}}/(2B_{\text{sm}} + 1) = \text{negl}(\lambda)$ for every fixed valid ciphertext and keys. Hence the algorithms satisfy Definition 3.10, with one field element per user per output bit.

5 Fast BFV-Based Multi-Key FHE from Ring-LWE

In this section, we construct fast multi-key FHE from Ring-LWE using BFV encryption and aggregate masked CDKS relinearization. Before giving the construction, we define ciphertext compression and the compact multiplication algorithm. A detailed overview of BFV and CDKS appears in Section A.2. We implement this construction and compare its multiplication time with CDKS, KKLSS, and single-key BFV in Appendix B.

We work over $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, where n is a power of two. Messages belong to R_t , with $t \geq 2$, $t \mid (q-1)$, and $\Delta = (q-1)/t$. We use centered message representatives. User i has an input secret s_i and an independently sampled compression secret σ_i . Define $\tilde{\sigma} = \sum_{i=1}^N \sigma_i$. A prepared ciphertext is a pair $\text{ct} = (c_0, c_1)$ with phase $\text{phase}_{\tilde{\sigma}}(\text{ct}) = c_0 + c_1 \tilde{\sigma} = \Delta\mu + e \pmod{q}$.

5.1 Preparing BFV Ciphertexts by Compression

Compression maps a source column $\mathbf{c} = (c_0, c_1, \dots, c_N)^\top \in R_q^{N+1}$ to a pair $(v, -u)$, preserving its phase $c_0 + \sum_j c_j s_j$ under $\tilde{\sigma}$ up to compression error. Fix a gadget $(h_{\mathbf{c}}, \mathbf{g}_{\mathbf{c}})$ from Definition 3.5, with

length k_c , bound B_c , and $h_c(0) = \mathbf{0}$.

Public compression material. For each $j \in [N]$, setup samples a uniform $\mathbf{a}_j \in R_q^{k_c}$. User i independently publishes

$$\mathbf{w}_{i,j} = \sigma_i \mathbf{a}_j + \delta_{i,j} s_i \mathbf{g}_c + \mathbf{e}_{i,j} \pmod{q}, \quad (24)$$

with fresh small error entries. Key aggregation computes

$$\mathbf{W}_j = \sum_{i=1}^N \mathbf{w}_{i,j} = \tilde{\sigma} \mathbf{a}_j + s_j \mathbf{g}_c + \sum_{i=1}^N \mathbf{e}_{i,j} \pmod{q}. \quad (25)$$

Compression. Define $\text{Compress}(\mathbf{c}) = (v, -u)$ by

$$u = \sum_{j=1}^N \langle h_c(c_j), \mathbf{a}_j \rangle, \quad v = c_0 + \sum_{j=1}^N \langle h_c(c_j), \mathbf{W}_j \rangle \pmod{q}. \quad (26)$$

For $\mathbf{E}_j = \sum_i \mathbf{e}_{i,j}$, gadget reconstruction gives

$$v - u\tilde{\sigma} = c_0 + \sum_j c_j s_j + \sum_j \langle h_c(c_j), \mathbf{E}_j \rangle \pmod{q},$$

where the last sum is the compression error. For a zero-padded fresh input $\mathbf{CT}_j = (c_{j,0}, c_{j,1})$, only its owner's terms are nonzero, since $h_c(0) = \mathbf{0}$. Thus preparation outputs $\mathbf{CT}_j^{\text{prep}} = (v_j, -u_j)$ with

$$u_j = \langle h_c(c_{j,1}), \mathbf{a}_j \rangle \pmod{q}, \quad (27)$$

$$v_j = c_{j,0} + \langle h_c(c_{j,1}), \mathbf{W}_j \rangle \pmod{q}. \quad (28)$$

5.2 Relinearization under the Sum Secret

For $\mathbf{ct} = (c_0, c_1)$ and $\mathbf{ct}' = (c'_0, c'_1)$, define $\text{Tens}(\mathbf{ct}, \mathbf{ct}') = (T_0, T_1, T_2)$ using centered representatives, polynomial products over R , and scaling and rounding before reduction modulo q :

$$\begin{aligned} T_0 &= \lfloor (t/q) c_0 c'_0 \rfloor \pmod{q}, \\ T_1 &= \lfloor (t/q) (c_0 c'_1 + c_1 c'_0) \rfloor \pmod{q}, \\ T_2 &= \lfloor (t/q) c_1 c'_1 \rfloor \pmod{q}. \end{aligned} \quad (29)$$

Its phase under a common secret σ is $T_0 + T_1 \sigma + T_2 \sigma^2$.

Masked keys and aggregation. Fix a gadget $(h, \mathbf{g})$ of length k and bound B_h . For a secret σ , an independent auxiliary secret r , and public masks $\mathbf{a}, \mathbf{u} \in R_q^k$, the relinearization vectors satisfy

$$\begin{aligned} \mathbf{B} &= -\sigma \mathbf{a} + \mathbf{E}_B, \\ \mathbf{D} &= -r \mathbf{a} + \sigma \mathbf{g} + \mathbf{E}_D, \\ \mathbf{V} &= -\sigma \mathbf{u} - r \mathbf{g} + \mathbf{E}_V \pmod{q}, \end{aligned} \quad (30)$$

with error vectors $\mathbf{E}_B, \mathbf{E}_D, \mathbf{E}_V$. Using the same setup masks $\mathbf{a}, \mathbf{u}$, each user i independently samples r_i and publishes

$$\mathbf{b}_i = -\sigma_i \mathbf{a} + \mathbf{e}_{B,i} \pmod{q}, \quad (31)$$

$$\mathbf{d}_i = -r_i \mathbf{a} + \sigma_i \mathbf{g} + \mathbf{e}_{D,i} \pmod{q}, \quad (32)$$

$$\mathbf{v}_i = -\sigma_i \mathbf{u} - r_i \mathbf{g} + \mathbf{e}_{V,i} \pmod{q}. \quad (33)$$

CDKS samples the mask corresponding to $\mathbf{u}$ independently per user.

$$\mathbf{B} = \sum_{i=1}^N \mathbf{b}_i, \quad \mathbf{D} = \sum_{i=1}^N \mathbf{d}_i, \quad \mathbf{V} = \sum_{i=1}^N \mathbf{v}_i. \quad (34)$$

These sums form EvalKey and satisfy Eq. (30) with $(\sigma, r) = (\tilde{\sigma}, \tilde{r})$, where $\tilde{r} = \sum_i r_i$ and $\mathbf{E}_X = \sum_i \mathbf{e}_{X,i}$.

Compact multiplication. The algorithm $\text{FastMult}(\text{PP}, (\mathbf{B}, \mathbf{D}, \mathbf{V}), \text{ct}, \text{ct}')$ reads $h, \mathbf{u}, t, q$ from PP, computes $(T_0, T_1, T_2) = \text{Tens}(\text{ct}, \text{ct}')$, and outputs (c_0^*, c_1^*) defined by

$$\mathbf{z} = h(T_2), \quad x = \langle \mathbf{z}, \mathbf{B} \rangle \pmod{q}, \quad (35)$$

$$c_0^* = T_0 + \langle h(x), \mathbf{V} \rangle \pmod{q}, \quad (36)$$

$$c_1^* = T_1 + \langle \mathbf{z}, \mathbf{D} \rangle + \langle h(x), \mathbf{u} \rangle \pmod{q}. \quad (37)$$

Each decomposition is computed once and reused. The output is a BFV pair under the same secret.

5.3 Construction

Here we construct MKFHE = (Setup, KeyGen, Enc, KeyAgg, CTPrep, Eval, Dec) for arithmetic circuits over R_t with fan-in-two addition and multiplication gates. We use ordinary BFV encryption, the compression of Eq. (26), and the compact multiplication above. The parameter settings are given below.

Construction 5.1. Let $d = d(\lambda)$ be the supported total depth and $N = N(\lambda)$ the number of users. All ciphertext and key computations are over R_q , except the integer products and rounding in Tens.

$\text{Setup}(1^\lambda, 1^d, 1^N) \rightarrow \text{PP}$. Do the following:

1. Choose n, t, q , symmetric secret and error distributions $\chi, \mathcal{D}_e$, and bounds B_χ, B_e according to these parameter settings. Let $\Delta = (q - 1)/t$ and $\text{pp} = (n, t, q, \Delta, \chi, \mathcal{D}_e, B_\chi, B_e)$.
2. Fix gadgets $(h_c, \mathbf{g}_c)$ and $(h, \mathbf{g})$ of lengths k_c, k and digit bounds B_c, B_h , respectively, with $h_c(0) = \mathbf{0}$.
3. Sample $\mathbf{a}_j \leftarrow R_q^{k_c}$ for every $j \in [N]$, and $\mathbf{a}, \mathbf{u} \leftarrow R_q^k$, uniformly and independently.

Output $\text{PP} = (\text{pp}, d, N, h_c, \mathbf{g}_c, B_c, h, \mathbf{g}, B_h, (\mathbf{a}_j)_{j \in [N]}, \mathbf{a}, \mathbf{u})$.

$\text{KeyGen}(\text{PP}, i) \rightarrow (\text{PK}_i, \text{SK}_i)$. Do the following:

1. Sample $s_i, \sigma_i, r_i \leftarrow \chi$, $a_i \leftarrow R_q$, and $e_{\text{pk},i} \leftarrow \mathcal{D}_e$. Sample $\mathbf{e}_{i,j} \leftarrow \mathcal{D}_e^{k_c}$ for every $j \in [N]$, and $\mathbf{e}_{B,i}, \mathbf{e}_{D,i}, \mathbf{e}_{V,i} \leftarrow \mathcal{D}_e^k$.
2. Compute the source BFV public key (b_i, a_i) , where $b_i = -a_i s_i + e_{\text{pk},i}$.
3. Compute $\mathbf{w}_{i,j} = \sigma_i \mathbf{a}_j + \delta_{i,j} s_i \mathbf{g}_c + \mathbf{e}_{i,j}$ for every $j \in [N]$, and compute $\mathbf{b}_i, \mathbf{d}_i, \mathbf{v}_i$ using Eqs. (31) to (33).

Output $\text{PK}_i = (i, (b_i, a_i), (\mathbf{w}_{i,j})_{j \in [N]}, \mathbf{b}_i, \mathbf{d}_i, \mathbf{v}_i)$, and $\text{SK}_i = (i, s_i, \sigma_i)$. The auxiliary secret r_i and sampled errors need not be retained.

$\text{Enc}(\text{PP}, \text{PK}_i, \mu) \rightarrow \text{CT}$. Parse the source public key (b_i, a_i) and let μ be the centered representative of the input in R_t . Sample $z \leftarrow \chi$ and $e_0, e_1 \leftarrow \mathcal{D}_e$. Compute the BFV pair $(c_0, c_1) = (b_i z + e_0 + \Delta\mu, a_i z + e_1) \pmod{q}$, and output $\text{CT} = (i, c_0, c_1)$.

$\text{KeyAgg}(\text{PP}, (\text{PK}_i)_{i \in [N]}) \rightarrow (\text{PrepKey}, \text{EvalKey})$. Compute $\mathbf{W}_j = \sum_{i=1}^N \mathbf{w}_{i,j}$ for every $j \in [N]$ and $(\mathbf{B}, \mathbf{D}, \mathbf{V}) = \sum_{i=1}^N (\mathbf{b}_i, \mathbf{d}_i, \mathbf{v}_i)$. Let $\text{CK} = (\mathbf{W}_j)_{j \in [N]}$. Output $\text{PrepKey} = \text{CK}$ and $\text{EvalKey} = (\mathbf{B}, \mathbf{D}, \mathbf{V})$.

$\text{CTPrep}(\text{PP}, \text{PrepKey}, \text{CT}) \rightarrow \text{CT}^{\text{prep}}$. Parse $\text{CK} = \text{PrepKey}$ and $\text{CT} = (j, c_0, c_1)$. Apply **Compress** to the zero-padded column $(c_0, c_1 \mathbf{e}_j^\top)^\top$, using the masks in PP and the column $\mathbf{W}_j$ in CK . Namely, compute only the owner terms in Eqs. (27) and (28), and output $\text{CT}^{\text{prep}} = (v, -u) \in R_q^2$.

$\text{Eval}(\text{PP}, \text{EvalKey}, C, (\text{CT}_i^{\text{prep}})_{i \in [N]}) \rightarrow \text{CT}_C$. Parse $\text{EvalKey} = (\mathbf{B}, \mathbf{D}, \mathbf{V})$ and evaluate C gate by gate. Represent each public constant by $(\Delta\mu, 0)$ for centered $\mu \in R_t$. Add pairs component-wise at an addition gate, and apply **FastMult**($\text{PP}, \text{EvalKey}, \text{ct}, \text{ct}'$) at a multiplication gate. Output the final pair CT_C .

$\text{Dec}(\text{PP}, (\text{SK}_i)_{i \in [N]}, \text{CT}_C) \rightarrow \mu$. Parse $\text{CT}_C = (c_0, c_1)$ and the compression secrets σ_i from the secret keys. Compute $\tilde{\sigma} = \sum_{i=1}^N \sigma_i$ and output

$$\left\lfloor \frac{t}{q} [c_0 + c_1 \tilde{\sigma}]_q \right\rfloor \pmod{t}. \quad (38)$$

Parameter settings. Choose a power-of-two ring degree n , a plaintext modulus $t \geq 2$, and an odd ciphertext modulus q with $t \mid (q - 1)$. Choose symmetric BFV secret and error distributions $\chi, \mathcal{D}_e$, with coefficient bounds $B_\chi, B_e \geq 1$ that fail only with negligible probability over all samples in a polynomial-size execution. For example, one may use independent ternary secret coefficients and discrete Gaussian error coefficients with a suitable tail bound. The following bounds ensure correctness.

Let $\delta_R = n$, $B_s = NB_\chi$, $B_{\text{ek}} = NB_e$, and $H = \delta_R(1 + B_s)$. The preparation and relinearization errors are bounded by

$$B_{\text{in}}(N) = 2\delta_R B_\chi B_e + B_e + Nk_c \delta_R B_c B_e, \quad (39)$$

$$B_{\text{rlin}}(N) = 2k\delta_R^2 B_h B_s B_{\text{ek}} + k\delta_R B_h B_{\text{ek}}. \quad (40)$$

The first term in B_{in} and the following B_e bound fresh BFV encryption error, while the last term bounds compression. In B_{rlin} , the terms account for the two secret-weighted errors and the remaining unweighted error of the aggregate CDKS key.

For a total depth bound d , let $E_0 = \max\{1, B_{\text{in}}(N)\}$, $\alpha = \delta_R t(2H + 4)$, and $\beta = H^2 + \delta_R t(H + 2) + B_{\text{rin}}(N)$. Choose q so that

$$E_{h+1} = \max\{2E_h + 1, \alpha E_h + \beta\}, \quad (41)$$

$$E_d < \frac{q}{2t} - \frac{1}{2}. \quad (42)$$

The addition term includes the correction for reducing centered messages modulo t , and the multiplication term includes scaling, rounding, and relinearization.

Both gadgets can use coefficient-wise binary decomposition, with $k = k_c = \log q$ and $B_c = 1$ under our ceiling convention. Since $\alpha \geq 2$ and $\beta \geq 1$, the recurrence has the closed form $E_d = \alpha^d E_0 + \beta(\alpha^d - 1)/(\alpha - 1)$. For fixed n, t, N, B_χ, B_e , its dependence on q is only through the gadget lengths. In particular, a sufficiently large constant in

$$\log q = O((d + 1) \log(2 + n + t + N + B_\chi + B_e + d))$$

suffices for the noise inequality, subject to the modulus congruence. Note that this is a correctness feasibility bound, and concrete choices must also satisfy the security assumption at that modulus and ring degree.

Theorem 5.2 (Fast MKFHE from Ring-LWE). Assuming circular Ring-LWE at the stated parameters, Construction 5.1 is a leveled MKFHE over R_t satisfying Definition 3.6, with two ring elements per ciphertext and $O(k)$ ring operations per gate after preparation. With binary gadgets and $n, \log q \leq \text{poly}(\lambda, d, \log N)$, it also satisfies full succinctness and optimal evaluation as defined in Definitions 3.7 and 3.8.

5.4 Analysis

Correctness. Using the bounds in the parameter settings we get

$$\|\tilde{\sigma}\|_\infty, \|\tilde{r}\|_\infty \leq NB_\chi = B_s, \quad \|\mathbf{E}_X\|_\infty \leq NB_e = B_{\text{ek}} \quad (X \in \{B, D, V\}). \quad (43)$$

Consider an input $\text{CT} = (j, c_0, c_1)$ encrypting μ . The BFV encryption error is $e_{\text{enc}} = ze_{\text{pk},j} + e_0 + s_j e_1$, with $\|e_{\text{enc}}\|_\infty \leq 2\delta_R B_\chi B_e + B_e$. Let $\boldsymbol{\eta} = h_c(c_1)$ and $\mathbf{E}_j = \sum_i \mathbf{e}_{i,j}$. The prepared pair satisfies

$$\begin{aligned} v - u\tilde{\sigma} &= c_0 + \langle \boldsymbol{\eta}, \mathbf{W}_j - \tilde{\sigma} \mathbf{a}_j \rangle && // \text{ by Compress} \\ &= c_0 + \langle \boldsymbol{\eta}, s_j \mathbf{g}_c + \mathbf{E}_j \rangle && // \text{ by Eq. (25)} \\ &= c_0 + c_1 s_j + \langle \boldsymbol{\eta}, \mathbf{E}_j \rangle && // \text{ by gadget reconstruction} \\ &= \Delta\mu + e_{\text{enc}} + e_{\text{cmp}} \pmod{q} && // \text{ by BFV correctness,} \end{aligned}$$

where $e_{\text{cmp}} = \langle \boldsymbol{\eta}, \mathbf{E}_j \rangle$ has norm at most $Nk_c \delta_R B_c B_e$. Thus

$$\text{phase}_{\tilde{\sigma}}(\text{CT}^{\text{prep}}) = \Delta\mu + e_{\text{enc}} + e_{\text{cmp}} \pmod{q}, \quad \|e_{\text{enc}} + e_{\text{cmp}}\|_\infty \leq B_{\text{in}}(N). \quad (44)$$

Every input wire therefore has a BFV phase under $\sigma = \tilde{\sigma}$. To track an addition, write $\mu + \mu' = [\mu + \mu']_t + t\tau$. Centered message coefficients give $\|\tau\|_\infty \leq 1$. Since $t\Delta = q - 1$, adding the two pairs

gives phase $\Delta[\mu + \mu']_t + e + e' - \tau \pmod{q}$. For input error bounds E, E' , the output error is at most

$$B_{\text{add}}(E, E') = E + E' + 1. \quad (45)$$

For multiplication, first consider the scaled tensor before relinearization. Let the input errors satisfy $\|e\|_\infty, \|e'\|_\infty \leq E$, where $1 \leq E < q/(2t)$. Use centered ciphertext lifts and work over R , writing

$$A = c_0 + c_1\sigma = \Delta\mu + e + q\kappa, \quad A' = c'_0 + c'_1\sigma = \Delta\mu' + e' + q\kappa'.$$

Let $H = \delta_R(1 + B_s)$ as in the parameter settings. Since $\|A\|_\infty, \|A'\|_\infty \leq qH/2$ and $\|\mu\|_\infty, \|\mu'\|_\infty \leq t/2$, the quotient polynomials satisfy

$$\|\kappa\|_\infty, \|\kappa'\|_\infty \leq H/2 + 1/2 + E/q \leq H + 1.$$

These quotients must be retained when scaling the product. Reduction modulo q before multiplication by t/q would lose their contribution to the error.

Let ρ_0, ρ_1, ρ_2 be the rounding residuals of the three tensor components before modular reduction. Each has norm at most $1/2$, so their weighted sum $\rho = \rho_0 + \sigma\rho_1 + \sigma^2\rho_2$ has norm at most

$$\|\rho\|_\infty \leq \frac{1}{2} + \frac{1}{2}\delta_R B_s + \frac{1}{2}\delta_R^2 B_s^2 \leq H^2/2.$$

Write $\mu\mu' = [\mu\mu']_t + t\tau$, where $\|\tau\|_\infty \leq \delta_R t/4 + 1/2$. Expanding $(t/q)AA' + \rho$ and using $t\Delta = q - 1$ gives

$$T_0 + T_1\sigma + T_2\sigma^2 = \Delta[\mu\mu']_t + e_{\text{tens}} \pmod{q}, \quad (46)$$

with the exact error representative

$$\begin{aligned} e_{\text{tens}} = & -\tau - (\Delta/q)\mu\mu' + (t\Delta/q)(\mu e' + \mu' e) + (t/q)ee' \\ & + \kappa(te' - \mu') + \kappa'(te - \mu) + \rho. \end{aligned} \quad (47)$$

Namely, $(t/q)\Delta^2 = \Delta - \Delta/q$, reducing $\Delta t\tau$ modulo q contributes $-\tau$, and the cross term $t\Delta\mu\kappa'$ contributes $-\mu\kappa'$. The term $tq\kappa\kappa'$ vanishes modulo q .

The first two terms in Eq. (47) have combined norm at most $\delta_R t/2 + 1/2$. The message-error terms contribute at most $\delta_R tE$, and $(t/q)ee'$ contributes at most $\delta_R E/2$ because $E < q/(2t)$. The two quotient terms contribute at most $2\delta_R(H + 1)(tE + t/2)$, and rounding contributes at most $H^2/2$. For $t \geq 2$, $H \geq 1$, and $E \geq 1$, their sum is bounded by

$$B_{\text{tens}}(B_s, E) = \delta_R t(2H + 4)E + H^2 + \delta_R t(H + 2) \geq \|e_{\text{tens}}\|_\infty. \quad (48)$$

Relinearization now replaces $T_2\sigma^2$ by a phase linear in σ . Let $r = \tilde{r}$, $\mathbf{z} = h(T_2)$, and $x = \langle \mathbf{z}, \mathbf{B} \rangle \pmod{q}$. Summing the local key equations gives

$$\mathbf{B} = -\sigma\mathbf{a} + \mathbf{E}_B, \quad (49)$$

$$\mathbf{D} = -r\mathbf{a} + \sigma\mathbf{g} + \mathbf{E}_D, \quad (50)$$

$$\mathbf{V} = -\sigma\mathbf{u} - r\mathbf{g} + \mathbf{E}_V \pmod{q}. \quad (51)$$

In particular, $x = -\sigma\langle \mathbf{z}, \mathbf{a} \rangle + \langle \mathbf{z}, \mathbf{E}_B \rangle \pmod{q}$. For the pair ct^* output by **FastMult**, we obtain

$$\begin{aligned} \text{phase}_\sigma(\text{ct}^*) &= T_0 + T_1\sigma + \sigma\langle \mathbf{z}, \mathbf{D} \rangle + \langle h(x), \mathbf{V} + \sigma\mathbf{u} \rangle \\ &= T_0 + T_1\sigma - \sigma r\langle \mathbf{z}, \mathbf{a} \rangle + \sigma^2 T_2 + \sigma\langle \mathbf{z}, \mathbf{E}_D \rangle - rx + \langle h(x), \mathbf{E}_V \rangle \\ &= T_0 + T_1\sigma + \sigma^2 T_2 + \sigma\langle \mathbf{z}, \mathbf{E}_D \rangle - r\langle \mathbf{z}, \mathbf{E}_B \rangle + \langle h(x), \mathbf{E}_V \rangle \pmod{q}. \end{aligned}$$

The second equality substitutes the aggregate vectors and uses ordinary gadget reconstruction. The last equality substitutes x , canceling the two σr terms. Thus

$$\text{phase}_\sigma(\text{ct}^*) = T_0 + T_1\sigma + T_2\sigma^2 + e_{\text{rlin}} \pmod{q}, \quad (52)$$

where

$$e_{\text{rlin}} = \sigma\langle \mathbf{z}, \mathbf{E}_D \rangle - r\langle \mathbf{z}, \mathbf{E}_B \rangle + \langle h(x), \mathbf{E}_V \rangle. \quad (53)$$

Each of the first two terms has norm at most $k\delta_R^2 B_h B_s B_{\text{ek}}$, and the last has norm at most $k\delta_R B_h B_{\text{ek}}$. This proves the bound $B_{\text{rlin}}(N)$ in Eq. (40). In particular, the error in $\mathbf{B}$ remains present through x . Neither decomposition requires independence from the keys, since its digit bound holds for every input. Combining scaling and relinearization, the product pair encrypts $[\mu\mu']_t$ with error at most

$$B_{\text{mult}}(N, E) = B_{\text{tens}}(NB_\chi, E) + B_{\text{rlin}}(N) = \alpha E + \beta. \quad (54)$$

For unequal input bounds, use $E = \max\{E_1, E_2, 1\}$. Prepared ciphertexts have error at most E_0 , and each addition or multiplication maps error bounds at level h to at most E_{h+1} in Eq. (41). The parameter condition keeps all these bounds below $q/(2t) - 1/2$, within the range used in the tensor calculation. Consequently, if $\mu = C((\mu_i)_{i \in [N]}) \in R_t$, the output satisfies

$$c_0 + c_1\tilde{\sigma} = \Delta\mu + e_{\text{out}} \pmod{q}, \quad \|e_{\text{out}}\|_\infty \leq E_d < \frac{q}{2t} - \frac{1}{2}. \quad (55)$$

A wire-specific bound may instead start from $B_{\text{in}}(N)$ for inputs, and apply Eqs. (45) and (54) at the corresponding gates.

Finally, consider one coefficient $\bar{\mu}$ of the centered output message. For some integer ζ , the corresponding centered phase coefficient is $\Delta\bar{\mu} + e - q\zeta$. The decoder scales it to

$$\frac{t}{q}(\Delta\bar{\mu} + e - q\zeta) = \bar{\mu} - t\zeta + \frac{te - \bar{\mu}}{q}.$$

Since $|\bar{\mu}| \leq t/2$, the last term has magnitude less than $1/2$ whenever

$$|e| < \text{rad}(q, t) := \frac{q}{2t} - \frac{1}{2}. \quad (56)$$

Rounding therefore returns $\bar{\mu} - t\zeta$, and reduction modulo t returns $\bar{\mu}$. Applying this argument coefficient-wise proves correctness. A union bound accounts for failures of the sampled-secret and error bounds.

Compactness and running time. After preparation, each ciphertext has two elements of R_q , representable with $2n \log q$ bits per output polynomial. Addition uses two ring additions. Multiplication uses two ordinary decompositions and four length- k inner products, hence $4k$ products in R_q , together with the fixed-size BFV tensor calculation. The tensor can use three integer-polynomial products by computing $c_0c'_0$, $c_1c'_1$, and $(c_0 + c_1)(c'_0 + c'_1)$, then subtracting the first two from the third before scaling and rounding. All sums and products in this calculation use integer lifts before reduction modulo q . Thus each gate uses $O(k)$ ring operations at fixed global parameters, without enumerating users. The evaluation key `EvalKey` contains $3k$ ring elements, representable with $3kn \log q$ bits. With binary gadgets, $k = \log q$, so $n, \log q \leq \text{poly}(\lambda, d, \log N)$ gives full succinctness and optimal evaluation.

Key aggregation takes $O(N^2k_c + Nk)$ ring additions and outputs Nk_c ring elements in `PrepKey` and $3k$ ring elements in `EvalKey`. Preparing a fresh input then uses one decomposition and two length- k_c inner products, for $O(k_c)$ ring operations. If inputs use only u distinct owners, aggregating only those compression columns takes $O(Nuk_c + Nk)$ additions. Every aggregated column still sums all N users' contributions. Circuit gates use only `EvalKey`.

Setup publishes $Nk_c + 2k$ mask elements. Each user publishes $Nk_c + 3k$ auxiliary ring elements and a two-element source public key, so total public material is quadratic in N , however, after aggregation, it reduces to linear in N .

Semantic security. As in CDKS, we assume circular security of Ring-LWE for the published key material. Here this means that the challenge user's compression entries and masked CDKS vectors are jointly indistinguishable from independent uniform values, given the public masks, source BFV public keys, and the other users' honestly generated public and secret keys. The assumption covers every published compression entry, including those in the dynamic setting of Remark 3.9. After this replacement, the auxiliary material is independent of the challenge user's source secret s_{i^*} . Ordinary BFV semantic security under Ring-LWE therefore hides the fresh ciphertext's message, establishing the semantic-security property of Definition 3.6. Since `KeyAgg`, `CTPrep`, and `Eval` are public computations, this guarantee also holds after preparation and evaluation.

5.5 Distributed Decryption

We add single-round distributed decryption by smudging each user's contribution to the BFV phase, as in CDKS. For this extension, use secret and error distributions supported within the bounds B_χ, B_e , and choose a larger modulus q during `Setup`. Recompute the gadget lengths and E_d at this modulus, and include an integer B_{sm} in `PP` such that

$$\frac{nE_d}{B_{\text{sm}}} = \text{negl}(\lambda), \quad E_d + NB_{\text{sm}} < \frac{q}{2t} - \frac{1}{2}.$$

For example, take $B_{\text{sm}} = \lceil n2^\lambda E_d \rceil$ for $\lambda \geq 2$ and increase q accordingly. This requires a superpolynomial ratio q/E_d . We assume circular Ring-LWE at the enlarged parameters.

`PartDec`(`PP`, `CTC`, $(\text{PK}_j)_{j \in [N]}$, i , SK_i) $\rightarrow p_i$. Parse `CTC` = (c_0, c_1) and σ_i from SK_i . Let $\gamma_i = \delta_{i,1}c_0 + c_1\sigma_i$. Sample a polynomial $e_i^{\text{sm}} \in R$ with independent uniform coefficients in $[-B_{\text{sm}}, B_{\text{sm}}] \cap \mathbb{Z}$, and output $p_i = \gamma_i + e_i^{\text{sm}} \pmod{q}$.

`FinDec`(`PP`, $(p_i)_{i \in [N]}$) $\rightarrow \mu$. Output $\mu = \lfloor (t/q) [\sum_i p_i]_q \rfloor \pmod{t}$, with coefficient-wise rounding.

Correctness and simulatability. By Eq. (55), $\sum_i \gamma_i = \Delta\mu + e_{\text{out}} \pmod{q}$ with $\|e_{\text{out}}\|_\infty \leq E_d$. The added noise has norm at most NB_{sm} , so the parameter condition ensures correct decoding. To simulate user i 's share from the centered output μ and the other users' secret keys, compute their γ_j and output $p'_i = \Delta\mu - \sum_{j \neq i} \gamma_j + e_i^{\text{sm}} \pmod{q}$. The real and simulated distributions differ by the shift e_{out} . Applying the uniform-interval smudging bound coefficient-wise gives statistical distance at most $nE_d/(2B_{\text{sm}} + 1) = \text{negl}(\lambda)$. Thus the algorithms satisfy Definition 3.10, with one ring element per user per output polynomial.

References

- [ACG⁺24] Abtin Afshar, Jiaqi Cheng, Rishab Goyal, Aayush Yadav, and Saikumar Yadugiri. Encrypted ram delegation: Applications to rate-1 extractable arguments, homomorphic nizks, mpc, and more. *Cryptology ePrint Archive*, 2024.
- [ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.
- [AG26] Abtin Afshar and Rishab Goyal. Fully-succinct multi-key FHE and rate-1 simulatable threshold decryption from LWE. In *CRYPTO*, 2026. Cryptology ePrint Archive, Paper 2026/1683.
- [AGY26] Abtin Afshar, Rishab Goyal, and Saikumar Yadugiri. Silent distributed cryptography for DNFs and threshold policies from lattices. Cryptology ePrint Archive, Paper 2026/1547, 2026.
- [AJJM20] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fully-homomorphic encryption in the plain model. Cryptology ePrint Archive, Paper 2020/180, 2020.
- [AJLA⁺12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, 2012. Cryptology ePrint Archive, Paper 2011/613.
- [AKÖ23] Yavuz Akin, Jakub Klemsa, and Melek Önen. A practical TFHE-based multi-key homomorphic encryption with linear complexity and low noise growth. Cryptology ePrint Archive, Paper 2023/065, 2023.
- [AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Succinct oblivious tensor evaluation and applications: Adaptively-secure laconic function evaluation and trapdoor hashing for all circuits. In *STOC*, 2025. Cryptology ePrint Archive, Paper 2025/336.
- [BCC⁺24] Jean-Philippe Bossuat, Rosario Cammarota, Ilaria Chillotti, Benjamin R. Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, Xi-anhui Lu, Carsten Maple, Alberto Pedrouzo-Ulloa, Rachel Player, Yuriy Polyakov, Luis Antonio Ruiz Lopez, Yongsoo Song, and Donggeon Yhee. Security guidelines for implementing homomorphic encryption. Cryptology ePrint Archive, Paper 2024/463, 2024.
- [BGG⁺18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, pages 565–596, 2018.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.

- [BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *TCC*, pages 645–677, 2017. Cryptology ePrint Archive, Paper 2017/386.
- [BLP⁺13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.
- [BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In *CRYPTO*, 2016.
- [BPTG14] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. *Cryptology ePrint Archive*, 2014.
- [Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual cryptology conference*, pages 868–886. Springer, 2012.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, 2011.
- [CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-key homomorphic encryption from TFHE. In *ASIACRYPT*, pages 446–472, 2019. Cryptology ePrint Archive, Paper 2019/116.
- [CDH⁺25] Geoffroy Couteau, Lalita Devadas, Aditya Hegde, Abhishek Jain, and Sacha Servan-Schreiber. Multi-key homomorphic secret sharing. Cryptology ePrint Archive, Paper 2025/094, 2025.
- [CDKS19] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *ACM CCS*, 2019.
- [CM15] Michael Clear and Ciarán McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *CRYPTO*, pages 630–656, 2015. Cryptology ePrint Archive, Paper 2014/798.
- [CO17] Wutichai Chongchitmate and Rafail Ostrovsky. Circuit-private multi-key FHE. Cryptology ePrint Archive, Paper 2017/010, 2017.
- [CZW17] Long Chen, Zhenfeng Zhang, and Xueqing Wang. Batched multi-hop multi-key FHE from ring-LWE with compact ciphertext extension. In *TCC*, pages 597–627, 2017. Cryptology ePrint Archive, Paper 2017/923.
- [DGJ⁺20] Yfke Dulek, Alex B Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 729–758. Springer, 2020.
- [DGMR21] Nico Döttling, Vipul Goyal, Giulio Malavolta, and Justin Raizes. Interaction-preserving compilers for secure computation. Cryptology ePrint Archive, Paper 2021/1503, 2021.

- [DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO*, pages 93–122, 2016. Cryptology ePrint Archive, Paper 2016/272.
- [DMR22] Vanesa Daza, Paz Morillo, and Sergi Rovira. Leveled multikey FHE with constant-size ciphertexts from RLWE. Cryptology ePrint Archive, Paper 2022/447, 2022.
- [FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012.
- [GBDL⁺16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [Goy18] Rishab Goyal. Quantum multi-key homomorphic encryption for polynomial-sized circuits. *Cryptology ePrint Archive*, 2018.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013. Cryptology ePrint Archive, Paper 2013/340.
- [Jia22] Bingbing Jiang. Multi-key FHE without ciphertext-expansion in two-server model. *Frontiers of Computer Science*, 16(1), 2022.
- [JVC18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. {GAZELLE}: A low latency framework for secure neural network inference. In *27th USENIX security symposium (USENIX security 18)*, pages 1651–1669, 2018.
- [KKL⁺23] Taechan Kim, Hyesun Kwak, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Asymptotically faster multi-key homomorphic encryption from homomorphic gadget decomposition. In *ACM CCS*, 2023. Cryptology ePrint Archive, Paper 2022/347.
- [KLSW21] Hyesun Kwak, Dongwon Lee, Yongsoo Song, and Sameer Wagh. A general framework of homomorphic encryption for multiple parties with non-interactive key-aggregation. Cryptology ePrint Archive, Paper 2021/1412, 2021.
- [KMS24] Hyesun Kwak, Seonhong Min, and Yongsoo Song. Towards practical multi-key TFHE: Parallelizable, key-compatible, quasi-linear complexity. In *PKC*, 2024. Cryptology ePrint Archive, Paper 2022/1460.
- [Lat21] Lattigo v2.3.0. <https://github.com/ldsec/lattigo/tree/v2.3.0>, October 2021. EPFL-LDS.
- [LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, 2012.

- [LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010.
- [LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for Ring-LWE cryptography. In *EUROCRYPT*, pages 35–54, 2013. Full version: Cryptology ePrint Archive, Paper 2013/293.
- [MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.
- [MPS26] Seonhong Min, Jeongeun Park, and Yongsoo Song. Multi-key fully homomorphic encryption with non-interactive setup in the plain model. Cryptology ePrint Archive, Paper 2026/322, 2026.
- [MTPBH21] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Proceedings on Privacy Enhancing Technologies*, 2021. Cryptology ePrint Archive, Paper 2020/304.
- [MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, 2016. Cryptology ePrint Archive, Paper 2015/345.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 333–342, 2009.
- [PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In *TCC*, pages 217–238, 2016. Cryptology ePrint Archive, Paper 2016/196.
- [RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180. Academic Press, 1978.
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.
- [WFNL15] David J Wu, Tony Feng, Michael Naehrig, and Kristin Lauter. Privately evaluating decision trees and random forests. *Cryptology ePrint Archive*, 2015.
- [XMH⁺26] Binwu Xiang, Seonhong Min, Intak Hwang, Zhiwei Wang, Haoqi He, Yuanju Wei, Kang Yang, Jiang Zhang, Yi Deng, and Yu Yu. HERDS: Multi-key fully homomorphic encryption with sublinear bootstrapping. In *EUROCRYPT*, 2026. Cryptology ePrint Archive, Paper 2025/1804.
- [ZCC⁺21] Tanping Zhou, Long Chen, Xiaoliang Che, Wenchao Liu, Zhenfeng Zhang, and Xiaoyuan Yang. Multi-key fully homomorphic encryption scheme with compact ciphertexts. Cryptology ePrint Archive, Paper 2021/1131, 2021.
- [ZHW25] Liang Zhou, Ruwei Huang, and Bingbing Wang. Enhancing multi-key fully homomorphic encryption with efficient key switching and batched multi-hop computations. *Applied Sciences*, 15(10):5771, 2025.

A Background on (Multi-Key) FHE

This appendix gives a more detailed account of the prior schemes used by our constructions, in the notation of this paper.

A.1 MKFHE over Fields: GSW and MW

We recall single-key GSW [GSW13] and the MW multi-key expansion [MW16] in the column notation used in Section 4. We work over $\mathbb{Z}_q$ for an odd prime q , let $\Delta = (q - 1)/2$, and use the binary gadget $\mathbf{G} = \mathbf{G}_{n,m}$ from Definition 3.3.

The GSW ciphertext. A single-key GSW ciphertext is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ encrypting one bit μ under $\mathbf{t} = (-\mathbf{s}^\top, 1)^\top$, where $\mathbf{s} \in \mathbb{Z}_q^{n-1}$. Its defining relation is

$$\mathbf{t}^\top \mathbf{C} = \mu \mathbf{t}^\top \mathbf{G} + \mathbf{e}^\top, \quad (57)$$

where $\mathbf{e} \in \mathbb{Z}^m$ is a small error column and $\mathbf{G} = \mathbf{G}_{n,m}$. We now use this relation to view the ciphertext in blocks. The gadget $\mathbf{G}$ has one group of $\log q$ columns for each key coordinate, followed by padding. Partition $\mathbf{C}$ in the same way,

$$\mathbf{C} = [\mathbf{C}^{(1)} \mid \dots \mid \mathbf{C}^{(n)} \mid \mathbf{C}^{(\text{pad})}].$$

Here $\mathbf{C}^{(k)} \in \mathbb{Z}_q^{n \times \log q}$ is the block corresponding to key coordinate $k \in [n]$. Within this block, index the columns by $\alpha = 0, \dots, \log q - 1$, in the order of the powers of two in $\mathbf{g}$. Thus

$$\mathbf{C}^{(k)} = [\mathbf{c}_{k,0} \mid \dots \mid \mathbf{c}_{k,\log q - 1}], \quad \mathbf{G}^{(k)} = [\mathbf{e}_k \mid 2\mathbf{e}_k \mid \dots \mid 2^{\log q - 1} \mathbf{e}_k].$$

The pair (k, α) denotes the column of $\mathbf{C}$ aligned with $2^\alpha \mathbf{e}_k$ in $\mathbf{G}$. Its ordinary column number is $(k - 1) \log q + \alpha + 1$. The remaining $m - n \log q$ columns form $\mathbf{C}^{(\text{pad})}$ and align with the zero columns of $\mathbf{G}$. Figure 1 shows this correspondence and the phase row obtained by multiplying by $\mathbf{t}^\top$.

Reading the columns. The *phase* of a column $\mathbf{c}$ is the inner product $\mathbf{t}^\top \mathbf{c}$. Reading Eq. (57) at position (k, α) gives

$$\mathbf{t}^\top \mathbf{c}_{k,\alpha} = \mu 2^\alpha t_k + e_{k,\alpha} = \begin{cases} -\mu 2^\alpha s_k + e_{k,\alpha}, & k < n, \\ \mu 2^\alpha + e_{n,\alpha}, & k = n. \end{cases}$$

All columns encode the same bit μ , multiplied by different key coordinates and powers of two. The padding columns have only error in their phases; the columns themselves need not be zero. Throughout this subsection, a column *encodes* a value when its phase is close to that value modulo q . This does not imply that a secret-dependent value can itself be recovered by rounding.

For message recovery, we use the last coordinate $t_n = 1$. The column $\mathbf{c} = \mathbf{C}\mathbf{G}^{-1}(\Delta \mathbf{e}_n)$ has phase close to $\Delta \mu$. Rounding distinguishes a phase close to 0 from one close to $q/2$. This one vector suffices to decrypt, while the full collection of coordinate blocks enables GSW evaluation.

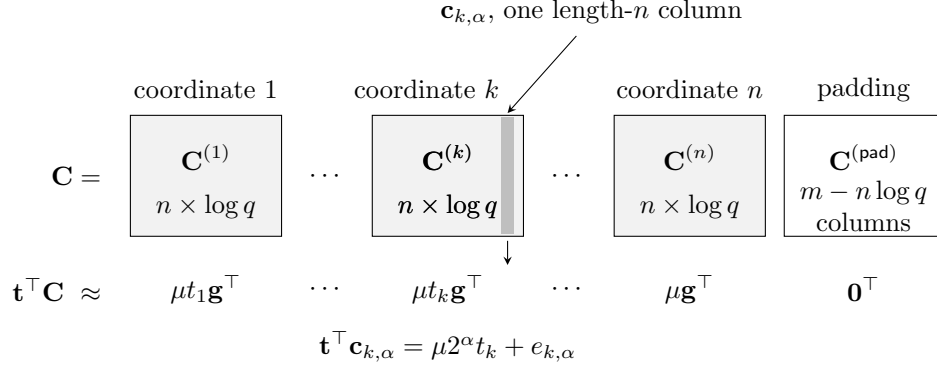


Figure 1: The GSW matrix, its coordinate blocks, and its phase row. Block $\mathbf{C}^{(k)}$ corresponds to key coordinate k . Its selected column $\mathbf{c}_{k,\alpha}$ aligns with $2^\alpha \mathbf{e}_k$ in $\mathbf{G}$, so its phase is $\mu 2^\alpha t_k$ plus error. The $\log q$ columns within each block are ordered by $\alpha = 0, \dots, \log q - 1$, and all blocks encrypt the same bit μ . The last key coordinate is $t_n = 1$. Here $\approx$ omits only the additive error row. Padding has phase close to zero, even when its ciphertext entries are nonzero.

Encryption and key-less evaluation. Let $\mathbf{B} \in \mathbb{Z}_q^{(n-1) \times m}$ be uniform and sample $\mathbf{e}_{\text{pk}} \leftarrow \chi^m$. Set $\mathbf{b} = \mathbf{B}^\top \mathbf{s} + \mathbf{e}_{\text{pk}}$. With public key $\mathbf{A} = [\mathbf{B}; \mathbf{b}^\top]$, a fresh GSW matrix has the form $\mathbf{C} = \mathbf{A}\mathbf{R} + \mu\mathbf{G}$, where $\mathbf{R} \in \{0, 1\}^{m \times m}$ is fresh randomness. Its phase row is $\mathbf{t}^\top \mathbf{C} = \mu \mathbf{t}^\top \mathbf{G} + \mathbf{e}_{\text{pk}}^\top \mathbf{R}$. For any public column $\mathbf{x}$, gadget reconstruction gives

$$\mathbf{t}^\top (\mathbf{C}\mathbf{G}^{-1}(\mathbf{x})) = \mu \mathbf{t}^\top \mathbf{x} + \mathbf{e}^\top \mathbf{G}^{-1}(\mathbf{x}).$$

The matrix therefore multiplies a column's phase by its encrypted bit, up to error. The basic GSW operations are addition $\mathbf{C}_1 + \mathbf{C}_2$ and multiplication $\mathbf{C}_1 \mathbf{G}^{-1}(\mathbf{C}_2)$, as recalled by MW [MW16, Section 5.1]. Addition adds the phase rows, while multiplication applies the preceding identity to every column of $\mathbf{C}_2$. Both use only ciphertext matrices and the public gadget, giving *key-less evaluation*.

Independent keys. Each user, indexed by i , generates its own secret $\mathbf{s}_i$ and key $\mathbf{t}_i = (-\mathbf{s}_i^\top, 1)^\top$. Its public key is $\text{pk}_i = \mathbf{A}_i = [\mathbf{B}; \mathbf{b}_i^\top]$, using the same public $\mathbf{B}$, and encryption uses only this user's key. MW lets a user encrypt before the other users are known.

The MW block matrix. Fix an input ciphertext and let $i \in [N]$ denote its owner. Its fresh MW representation contains a single-key GSW matrix $\mathbf{C}$ and auxiliary encryptions of its randomness. Expansion arranges N^2 blocks in N block rows and N column groups, one of each per user. We use t for the block-row index and j for the column-group index. Each block has n rows and m columns. For every other user $j \neq i$, the auxiliary material and that user's public key produce a correction block $\mathbf{X}_j \in \mathbb{Z}_q^{n \times m}$. MW places $\mathbf{C}$ on every diagonal block and places $\mathbf{X}_j$ in block (i, j) . All other blocks are zero. Thus the expanded ciphertext has the form

$$\widehat{\mathbf{C}} = (\widehat{\mathbf{C}}_{t,j})_{t,j \in [N]} \in \mathbb{Z}_q^{nN \times mN}, \quad \widehat{\mathbf{C}}_{t,j} = \begin{cases} \mathbf{C}, & t = j, \\ \mathbf{X}_j, & t = i \neq j, \\ 0_{n \times m}, & \text{otherwise.} \end{cases}$$

Figure 2 shows the general block structure and one selected column. We retain the complete fresh ciphertext, including the auxiliary material, for this public expansion.

$$\begin{array}{c}
 \text{group } j \\
 \begin{array}{c} t=i \\ t=j \end{array} \left[\begin{array}{cccccc} \mathbf{C} & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \vdots & & \vdots & & \vdots \\ \mathbf{X}_1 & \cdots & \mathbf{C} & \cdots & \mathbf{X}_j & \cdots & \mathbf{X}_N \\ \vdots & & 0 & \ddots & 0 & & \vdots \\ 0 & \cdots & 0 & \cdots & \mathbf{C} & \cdots & 0 \\ \vdots & & \vdots & & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 0 & \mathbf{C} \end{array} \right] \xrightarrow{(j,k,\alpha)} \left[\begin{array}{c} \mathbf{0}_n \\ \vdots \\ \mathbf{x}_{j,k,\alpha} \\ \vdots \\ \mathbf{c}_{k,\alpha} \\ \vdots \\ \mathbf{0}_n \end{array} \right]
 \end{array}$$

Figure 2: Structure of the MW expanded ciphertext $\hat{\mathbf{C}}$ for N users and input owner i , following [MW16]. Each block is $n \times m$. The highlighted column (j, k, α) is shown on the right, with its n -entry blocks in the same row order. Here $\mathbf{x}_{j,k,\alpha}$ is column (k, α) of $\mathbf{X}_j$. The schematic shows $i \neq j$, and for $j = i$, only $\mathbf{c}_{k,\alpha}$ in block row i remains.

The purpose of $\mathbf{X}_j$ is to correct the phase of the diagonal copy of $\mathbf{C}$ when it is paired with user j 's key. Namely, MW ensures $\mathbf{t}_i^\top \mathbf{X}_j + \mathbf{t}_j^\top \mathbf{C} = \mu \mathbf{t}_j^\top \mathbf{G} + \mathbf{e}_j^\top$ for $j \neq i$. For $j = i$, the original GSW relation already holds. To read all N column groups together, concatenate the local decryption keys and place one copy of $\mathbf{G}$ on each diagonal block,

$$\hat{\mathbf{t}} = \begin{pmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_N \end{pmatrix} \in \mathbb{Z}_q^{nN}, \quad \hat{\mathbf{G}}_N = \begin{pmatrix} \mathbf{G} & & 0 \\ & \ddots & \\ 0 & & \mathbf{G} \end{pmatrix} = I_N \otimes \mathbf{G} \in \mathbb{Z}_q^{nN \times mN}.$$

The associated decomposition applies $\mathbf{G}^{-1}$ separately to each length- n block. This is the expanded gadget used by MW. Multiplying the expanded ciphertext by $\hat{\mathbf{t}}^\top$ now gives

$$\hat{\mathbf{t}}^\top \hat{\mathbf{C}} = \mu \hat{\mathbf{t}}^\top \hat{\mathbf{G}}_N + \mathbf{e}^\top, \tag{58}$$

where $\mathbf{e} = (\mathbf{e}_1^\top, \dots, \mathbf{e}_N^\top)^\top$. MW implements the regular GSW evaluation on these expanded ciphertexts. Its N block rows and N column groups explain the nmN^2 entries and the quadratic dependence on the number of parties.

Locating a column and reading its phase. Write $\hat{\mathbf{C}}_j = [\hat{\mathbf{C}}_{1,j}^\top \mid \cdots \mid \hat{\mathbf{C}}_{N,j}^\top]^\top \in \mathbb{Z}_q^{nN \times m}$ for column group j . Within it, we keep the same coordinate-and-power order as in GSW,

$$\hat{\mathbf{C}}_j = \underbrace{[\hat{\mathbf{c}}_{j,1,0} \cdots \hat{\mathbf{c}}_{j,1,\log q-1}]}_{k=1} \mid \cdots \mid \underbrace{[\hat{\mathbf{c}}_{j,n,0} \cdots \hat{\mathbf{c}}_{j,n,\log q-1}]}_{k=n} \mid \hat{\mathbf{c}}_{j,\text{pad}}.$$

Thus $\hat{\mathbf{c}}_{j,k,\alpha}$ is one entire length- nN column. Its indices denote, in order, the user's column group, the key coordinate, and the power of two. It is column $(j-1)m + (k-1)\log q + \alpha + 1$ of $\hat{\mathbf{C}}$. Reading

Eq. (58) at this position gives

$$\widehat{\mathbf{t}}^\top \widehat{\mathbf{c}}_{j,k,\alpha} = \begin{cases} -\mu 2^\alpha s_{j,k} + e_{j,k,\alpha}, & k < n, \\ \mu 2^\alpha + e_{j,n,\alpha}, & k = n. \end{cases} \quad (59)$$

The index j specifies which secret coordinate the phase involves. All columns use the full decryption key $\widehat{\mathbf{t}}$, pairing each of their n -entry blocks with the corresponding $\mathbf{t}_t$.

The constant-coordinate columns, with $k = n$, contain the information needed for message recovery. Namely, writing $\gamma_\alpha = (g^{-1}(\Delta))_\alpha$, then

$$\mathbf{c} = \sum_{\alpha=0}^{\log q-1} \gamma_\alpha \widehat{\mathbf{c}}_{1,n,\alpha} \implies \widehat{\mathbf{t}}^\top \mathbf{c} = \Delta\mu + \sum_{\alpha=0}^{\log q-1} \gamma_\alpha e_{1,n,\alpha}.$$

This is an nN -coordinate vector sufficient for decryption. The other columns are needed for evaluation rather than for recovering this one message.

A.2 MKFHE over Rings: BFV and CDKS

We recall BFV encryption [FV12] and the CDKS multi-key representation [CDKS19], using the phase and sign conventions of Section 5.

The BFV ciphertext. We work over $R_q = \mathbb{Z}_q[X]/(X^n+1)$, where n is a power of two. Messages belong to R_t , for a plaintext modulus $t \geq 2$. We use the BFV scaling convention $\Delta = (q-1)/t$, requiring $t \mid (q-1)$, and centered message representatives. A ciphertext $\mathbf{ct} = (c_0, c_1) \in R_q^2$ under secret s has *phase*

$$\text{phase}_s(\mathbf{ct}) = c_0 + c_1 s = \Delta\mu + e \pmod{q}. \quad (60)$$

The phase is the noisy encoding from which decryption recovers μ . Namely, compute

$$\text{BFV.Dec}_s(\mathbf{ct}) = \left\lfloor \frac{t}{q} [c_0 + c_1 s]_q \right\rfloor \pmod{t}. \quad (61)$$

This succeeds when $\|e\|_\infty < q/(2t) - 1/2$, as proved in Eq. (56).

For completeness, ordinary BFV encryption [FV12] uses a public key $(b, a) = (-as + e_{\text{pk}}, a)$. With fresh small z, e_0, e_1 , it outputs

$$(c_0, c_1) = (bz + e_0 + \Delta\mu, az + e_1) \pmod{q}, \quad (62)$$

whose phase is $\Delta\mu + ze_{\text{pk}} + e_0 + se_1$.

Single-key evaluation. Addition adds the two components. Multiplication first multiplies the phases and restores the plaintext scale by scaling and rounding. This produces three coefficients with phase $T_0 + T_1 s + T_2 s^2$. BFV relinearization uses a public key encoding multiples of s^2 under s to return to two components [FV12]. Multiplication therefore requires public material for the decryption secret.

BFV multiplication before relinearization. Let $\text{ct} = (c_0, c_1)$ and $\text{ct}' = (c'_0, c'_1)$ be ciphertexts under the same secret σ . Multiplying their phases gives

$$(c_0 + c_1\sigma)(c'_0 + c'_1\sigma) = c_0c'_0 + (c_0c'_1 + c_1c'_0)\sigma + c_1c'_1\sigma^2.$$

This has three coefficients, corresponding to $1, \sigma, \sigma^2$. Since each input encodes its message multiplied by Δ , their product has plaintext scale Δ^2 . BFV restores the scale by multiplying each coefficient by t/q and rounding. The exact scaled tensor **Tens** is defined in Eq. (29). Writing its output as (T_0, T_1, T_2) , the first two terms already have the form of a BFV phase. Relinearization replaces $T_2\sigma^2$ by a pair (d_0, d_1) satisfying $d_0 + d_1\sigma \approx T_2\sigma^2$, so $(T_0 + d_0, T_1 + d_1)$ is a BFV pair encrypting the product message.

The CDKS expansion. CDKS uses one coordinate for each user's secret [CDKS19]. Write its ciphertext and concatenated secret as

$$\mathbf{c} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{pmatrix}, \quad \hat{\mathbf{s}} = \begin{pmatrix} 1 \\ s_1 \\ \vdots \\ s_N \end{pmatrix}, \quad \hat{\mathbf{s}}^\top \mathbf{c} = c_0 + \sum_{i=1}^N c_i s_i = \Delta\mu + e \pmod{q}.$$

The first coordinate contributes directly to the phase, and coordinate i multiplies s_i . A fresh BFV pair (c_0, c_1) owned by user j expands to $(c_0, c_1 \mathbf{e}_j^\top)^\top$, where $\mathbf{e}_j$ is the j -th unit vector of length N . Its phase remains $c_0 + c_1 s_j$. This zero padding aligns ciphertexts under the same concatenated key, without changing their messages or errors.

CDKS evaluates on this expanded representation, relinearizing the products $s_i s_j$ created by a multiplication gate. Its masked representation supplies the relinearization mechanism used in Section 5.2. We recall the single-secret specialization below.

The masked CDKS representation for one secret. The masked CDKS representation constructs this pair [CDKS19]. Let $(h, \mathbf{g})$ be an ordinary gadget from Definition 3.5, with length k and digit bound B_h . Decompose the known coefficient as $\mathbf{z} = h(T_2)$, so that $\langle \mathbf{z}, \mathbf{g} \rangle = T_2 \pmod{q}$. Introduce an independent small auxiliary secret r and public masks $\mathbf{a}, \mathbf{u} \in R_q^k$. The required vectors are

$$\begin{aligned} \mathbf{B} &= -\sigma \mathbf{a} + \mathbf{E}_B, \\ \mathbf{D} &= -r \mathbf{a} + \sigma \mathbf{g} + \mathbf{E}_D, \\ \mathbf{V} &= -\sigma \mathbf{u} - r \mathbf{g} + \mathbf{E}_V \pmod{q}. \end{aligned} \tag{63}$$

The pair $(\mathbf{D}, \mathbf{a})$ encodes multiples of σ under r , and $(\mathbf{V}, \mathbf{u})$ encodes multiples of $-r$ under σ . Each vector contains only degree-one secret terms, σ or r , multiplied by public coefficients. These linked vectors allow the evaluator to create the quadratic secret term and cancel its auxiliary mask without learning either secret. In CDKS, this mechanism handles a coefficient associated with a pair of user secrets. Here both secret positions refer to σ , so the result belongs to one BFV pair.

To see the cancellation, omit errors temporarily and let $x = \langle \mathbf{z}, \mathbf{B} \rangle = -\sigma \langle \mathbf{z}, \mathbf{a} \rangle$. The pair

$$(\langle h(x), \mathbf{V} \rangle, \langle \mathbf{z}, \mathbf{D} \rangle + \langle h(x), \mathbf{u} \rangle)$$

has phase

$$\begin{aligned} \sigma \langle \mathbf{z}, \mathbf{D} \rangle + \langle h(x), \mathbf{V} + \sigma \mathbf{u} \rangle &= -\sigma r \langle \mathbf{z}, \mathbf{a} \rangle + \sigma^2 \langle \mathbf{z}, \mathbf{g} \rangle - rx \\ &= \sigma^2 T_2. \end{aligned}$$

Here $\mathbf{D}$ supplies the desired $\sigma^2 T_2$ term together with a mask. The public value x and the vector $\mathbf{V}$ supply the opposite mask. Adding T_0 to the first component and T_1 to the second gives the full product phase. Both decompositions use only the ordinary reconstruction identity.

In the multi-key CDKS construction, the mask corresponding to $\mathbf{u}$ is sampled independently by each user. The common-mask choice and aggregation used in our construction are given in Section 5.2.

B Implementation and Evaluation

We implement Construction 5.1 in Go 1.26.4 using Lattigo v2.3.0. We refer to our implementation as Fast CDKS and compare it with our implementation of CDKS [CDKS19], the existing KKLSS [KKL⁺23], and Lattigo’s single-key BFV [FV12, Lat21]. We measure ciphertext sizes and one multiplication of two distinct ciphertexts, each involving the same N users.

Implementation and parameters. We use $n = 2^{14}$, $t = 65537$, and a 348-bit ciphertext modulus q consisting of six 58-bit primes, each congruent to 1 modulo $2nt$. Secrets are uniform ternary. Gaussian errors have standard deviation 3.2 and truncation bound $B_e = 48$. Relinearization uses the CDKS residue gadget with $k = 6$ centered digits and $B_h < 2^{57}$. Its gadget entries are CRT projectors, so decomposition only centers and copies residues. Compression splits each centered residue into two signed base-2²⁹ pieces, giving $k_c = 12$ and $B_c = 2^{29} - 1$.

Both CDKS evaluators retain the construction’s centered tensor and exact rounding. An auxiliary 420-bit residue basis satisfies $P_{\text{aux}} > 2ntq + 2$ and carries no encrypted key material. Guarded CRT conversion processes eight coefficients together and falls back to exact integer arithmetic near correction boundaries. We preallocate gate buffers, reuse decompositions, and accumulate products in the transform domain. Our relinearization uses two decompositions, four length-six inner products, and three inverse transforms of a full polynomial. The preceding tensor requires additional transforms and basis conversions. Neither step loops over users.

CDKS shares these arithmetic kernels and implements the symmetric-pair and diagonal-key optimizations [CDKS19]. It combines $c_i c'_j + c_j c'_i$ before rounding for $i < j$, and uses an ordinary switching key for each s_i^2 term. Thus our workload has $N(N - 1)/2$ off-diagonal terms and N diagonal terms. This comparator is our implementation of CDKS in Go, rather than the authors’ unpublished SEAL artifact.

Lattigo BFV uses the same q and one 60-bit switching prime, giving a 408-bit secret-bearing modulus. Its Gaussian sampler has standard deviation 3.2 and truncation bound 19. The published ordinary-RLWE calibration [BCC⁺24] gives a 430-bit ceiling for $n = 2^{14}$, uniform ternary secrets, error standard deviation 3.19, and a classical 128-bit target. The 348-bit CDKS and 408-bit BFV key moduli lie below this ceiling, which does not estimate security under the circular assumption.

KKLSS retains its original PN14QP439 parameters, with the same n, t and six ciphertext limbs, a 319-bit ciphertext modulus, and a 439-bit key modulus. The limb count agrees, but the modulus sizes and arithmetic differ.

Workload and measurement. For each $N \in \{2, 4, 8, 16, 32\}$, we form two independent batches of N fresh encryptions, one per user, with uniformly sampled slots modulo t . Summing each batch gives two distinct operands, each with all N user coordinates present. Fast CDKS prepares the fresh inputs before summing them. Single-key BFV independently encrypts the same two sum plaintexts under one key. All operands are ready before the multiplication timer starts.

We measure ciphertext payloads by counting the stored 64-bit residue coefficients of each operand and the relinearized output. Their sizes agree for every row and construction.

We run on an Apple M3 with eight CPU cores and 24 GB of memory, using macOS 15.7.3 and one execution thread on AC power. We report medians of 15 repetitions after three warmups. The four gate measurements are interleaved with a rotating order. We collect garbage before each execution and disable ordinary collection during timing, with a 14 GiB memory limit. CDKS and BFV reuse output buffers, while KKLSS retains its allocating interface. Key generation, encryption, preparation, and decryption are excluded from gate times.

N	Ours		CDKS		KKLSS		BFV	
	$ \text{CT} $ (MB)	T_{Mult} (ms)	$ \text{CT} $ (MB)	T_{Mult} (ms)	$ \text{CT} $ (MB)	T_{Mult} (ms)	$ \text{CT} $ (MB)	T_{Mult} (ms)
2	1.6	29	2.4	54	2.4	89	1.6	21
4	1.6	30	3.9	168	3.9	174	1.6	22
8	1.6	30	7.1	564	7.1	336	1.6	21
16	1.6	30	13.4	2102	13.4	665	1.6	21
32	1.6	30	26.0	8180	26.0	1353	1.6	21

Table 2: Ciphertext size $|\text{CT}|$ and median time for one multiplication T_{Mult} , including relinearization, of two distinct ciphertexts, each involving the same N users. Single-key BFV evaluates the same plaintext product under one key. All rows use $n = 2^{14}$ and $t = 65537$. CDKS and BFV use a 348-bit q , while KKLSS retains its 319-bit q .

Results and preparation. Table 2 reports the ciphertext sizes and gate times. Our median multiplication time ranges from 29.48 to 30.19 ms across the five user counts. At $N = 32$, our multiplication is $275.13\times$ faster than CDKS and $45.50\times$ faster than KKLSS, and takes $1.39\times$ the time of single-key BFV, using unrounded medians. Public preparation aggregates already-generated public keys, prepares the $2N$ fresh input ciphertexts, and sums them into the two operands. The median times are 53.47, 110.38, 233.32, 513.41, and 1240.63 ms for $N = 2, 4, 8, 16, 32$, respectively. This work is performed before the measured gate.

Correctness and costs. We verify every output slot after every execution. For each summed CDKS input, the phase error is at most $NE_0 + N$, where E_0 is the fresh prepared-input bound. Applying the multiplication recurrence in Eq. (41) once, and conservatively doubling it for CDKS, keeps both bounds below the decoding radius. Observed errors satisfy these bounds. Lattigo BFV

has a separate correctness calculation, while KKLSS is checked empirically. The artifact contains exact prime lists, certificates, raw samples, and tests against integer arithmetic references.

Our ciphertexts and single-key BFV ciphertexts each contain two ring elements (1.57 MB). CDKS and KKLSS ciphertexts each contain $N + 1$ ring elements, growing from 2.36 MB at $N = 2$ to 25.95 MB at $N = 32$. At $N = 32$, our ciphertexts are 16.5 times smaller than both multi-key baselines. At $N = 32$, all user public keys contain 10.17 GB and the aggregate keys contain 316.15 MB, excluding metadata and working space. These costs accompany the removal of party-dependent gate work. We do not instantiate statistically simulatable distributed decryption or its larger parameters.

C Removing User Indices

We remove the index from **KeyGen** while encrypting each input only once. As in our main constructions, exactly N users participate in evaluation. They have no assigned positions during key generation or encryption. At key aggregation, their public keys are supplied in canonical order, fixing positions $1, \dots, N$ for the subsequent public algorithms. Ciphertexts are supplied for preparation together with their owners' positions, and decryption uses the secret keys in the same order. Preparation evaluates source decryption [Gen09] using the selected encryptions of the source secret.

Construction C.1. Let indMKFHE be an indexed MKFHE as in Section 3.2, supporting multiple inputs per user and continued evaluation under the same N users, with correctness for the composed circuit. Let PKE be semantically secure for the same message space, with supported decryption circuits $D_{\text{ct}^*}(s) = \text{PKE.Dec}(s, \text{ct}^*)$. We encode source secrets into underlying plaintexts, include decoding in D_{ct^*} , and apply **Enc** and **CTPrep** componentwise to bundles.

We construct MKFHE as follows, using the ordering above and writing one application input per user for simplicity.

Setup($1^\lambda, 1^d, 1^N$) $\rightarrow$ **PP**. Let D bound the depth of source decryption followed by depth- d evaluation. Sample $\text{pp} \leftarrow \text{indMKFHE.Setup}(1^\lambda, 1^D, 1^N)$ and output $\text{PP} = (\text{pp}, d, N)$.

KeyGen(**PP**) $\rightarrow$ (**PK**, **SK**). Sample $(\text{pk}^*, s^*) \leftarrow \text{PKE.KeyGen}(1^\lambda)$. Independently for every $\ell \in [N]$, sample

$$\begin{aligned} (\text{pk}^{(\ell)}, \text{sk}^{(\ell)}) &\leftarrow \text{indMKFHE.KeyGen}(\text{pp}, \ell), \\ \text{bk}^{(\ell)} &\leftarrow \text{indMKFHE.Enc}(\text{pp}, \text{pk}^{(\ell)}, s^*). \end{aligned}$$

Output $\text{PK} = (\text{pk}^*, (\text{pk}^{(\ell)}, \text{bk}^{(\ell)})_{\ell \in [N]})$ and $\text{SK} = (\text{sk}^{(\ell)})_{\ell \in [N]}$.

Enc(**PP**, **PK**, μ) $\rightarrow$ **CT**. Output $\text{CT} \leftarrow \text{PKE.Enc}(\text{pk}^*, \mu)$.

KeyAgg(**PP**, $(\text{PK}_i)_{i \in [N]}$) $\rightarrow$ (**PrepKey**, **EvalKey**). For the canonically ordered public keys, compute

$$\begin{aligned} (\text{prep}, \text{eval}) &= \text{indMKFHE.KeyAgg}(\text{pp}, (\text{pk}_i^{(i)})_{i \in [N]}), \\ \widehat{\text{bk}}_i &= \text{indMKFHE.CTPrep}(\text{pp}, \text{prep}, \text{bk}_i^{(i)}) \quad (i \in [N]). \end{aligned}$$

Output $\text{PrepKey} = ((\widehat{\text{bk}}_i)_{i \in [N]}, \text{eval})$ and $\text{EvalKey} = \text{eval}$.

$\text{CTPrep}(\text{PP}, \text{PrepKey}, (i, \text{CT})) \rightarrow \text{CT}^{\text{prep}}$. For a ciphertext associated with the i th public key, let $\text{ct}^* = \text{CT}$ and output

$$\text{CT}^{\text{prep}} = \text{indMKFHE.Eval}(\text{pp}, \text{eval}, D_{\text{ct}^*}, \widehat{\text{bk}}_i).$$

$\text{Eval}(\text{PP}, \text{EvalKey}, C, (\text{CT}_i^{\text{prep}})_{i \in [N]}) \rightarrow \text{CT}_C$. Return $\text{indMKFHE.Eval}(\text{pp}, \text{EvalKey}, C, (\text{CT}_i^{\text{prep}})_{i \in [N]})$.

$\text{Dec}(\text{PP}, (\text{SK}_i)_{i \in [N]}, \text{CT}_C) \rightarrow \mu$. Output $\text{indMKFHE.Dec}(\text{pp}, (\text{sk}_i^{(i)})_{i \in [N]}, \text{CT}_C)$.

Theorem C.2 (Removing MKFHE Indexing). Assuming indexed leveled MKFHE with fully succinct ciphertext and optimal evaluation, Construction C.1 is an unindexed MKFHE satisfying full succinctness and optimal evaluation as defined in Definitions 3.7 and 3.8.

Corollary C.3 (MKFHE Without Indexing). Assuming LWE or circular Ring-LWE, Construction C.1 is a leveled MKFHE satisfying full succinctness and optimal evaluation as defined in Definitions 3.7 and 3.8.

Analysis. *Correctness.* For an input $\text{ct}_i^* \leftarrow \text{PKE.Enc}(\text{pk}_i^*, \mu_i)$, follow the source secret and input through three steps.

1. **Encrypt the source secret.** Key generation publishes $\text{bk}_i^{(\ell)}$ for every candidate position ℓ . Each bundle encrypts s_i^* under $\text{pk}_i^{(\ell)}$, with its original expansion material.
2. **Prepare the selected bundle.** The common ordering selects $\text{pk}_i^{(i)}$ and $\text{bk}_i^{(i)}$. Key aggregation prepares this bundle under the same aggregate keys used for evaluation,

$$\underbrace{s_i^*}_{s_i^* \text{ under candidate } i} \xrightarrow{\text{indMKFHE.CTPrep}} \underbrace{\widehat{\text{bk}}_i}_{s_i^* \text{ under the aggregate keys}}.$$

3. **Key switching the input ciphertext.** Input preparation evaluates $D_{\text{ct}_i^*}$ on $\widehat{\text{bk}}_i$. The resulting ciphertext satisfies

$$\text{indMKFHE.Dec}(\text{pp}, (\text{sk}_j^{(j)})_{j \in [N]}, \text{CT}_i^{\text{prep}}) = D_{\text{ct}_i^*}(s_i^*) = \mu_i.$$

It is already in the aggregate evaluation domain and needs no further underlying preparation. Continued evaluation computes

$$F(s_1^*, \dots, s_N^*) = C(D_{\text{ct}_1^*}(s_1^*), \dots, D_{\text{ct}_N^*}(s_N^*)) = C(\mu_1, \dots, \mu_N).$$

Correctness follows for suitable underlying parameters, except with negligible probability from the underlying schemes. Our constructions support continued evaluation by the gate-wise correctness arguments in Sections 4.4 and 5.4.

Semantic security. Replace all encryptions of the challenge user's s^* by encryptions of zero, componentwise, using semantic security of indMKFHE with its complete public keys. The source secret is independent of the indexed keys and known in this reduction. Then semantic security of PKE hides the challenge ciphertext. All remaining operations are public postprocessing, so the transformation adds no circular-security assumption.

Costs. We use the conventions of Table 1. Each user stores N underlying secret keys. Over fields, a dimension- n LWE source PKE has $O(n)$ encoded secret bits and $O(n)$ -gate decryption,

including reduction and rounding. Each user publishes N bundles of $O(n)$ fresh encryptions, so its public key grows from $O(Nn+m)$ to $O(N^2n+Nm^2)$. Preparing the N selected bundles increases aggregation from $O(N^2n)$ to $O(N^2n^2m^2)$ and the preparation key from $O(Nm)$ to $O(Nn^2m)$. Fresh ciphertext size falls from $O(nm^3)$ to $O(n)$, while evaluating source decryption changes per-input preparation from $O(Nnm^2)$ to $O(n^2m^2)$. Over rings, only the public-key entry changes, from $O(N)$ to $O(N^2)$, provided a compatible source PKE has $O(1)$ -element public keys, ciphertexts, and secret encodings, and complete supported decryption in $O(1)$ ring gates. This requirement includes decoding and coefficient-wise rounding. A one-polynomial secret alone does not establish it, and we do not instantiate this requirement here. Ciphertext sizes after preparation, the evaluation key, and gate costs are inherited at the chosen parameters. Full succinctness and optimal evaluation are preserved when source decryption depth is polynomial in $\lambda, d, \log N$.

Short secret keys. Expand a private PRG seed κ into the random tapes for all N indexed KeyGen calls, using a fixed partition. Then $\text{SK} = \kappa$ suffices to regenerate the selected key, and a PRG hybrid replaces these tapes by independent uniform tapes. Independently, generate (pk^*, s^*) from a short PRG-expanded seed ρ^* , and encrypt ρ^* in each $\text{bk}^{(\ell)}$. Include secret reconstruction in source decryption, using suitable parameters for the supported composed circuit. This can shorten the public encrypted-secret bundles, while the N indexed public keys remain.