Fully-Succinct Multi-Key FHE & Rate-1 Simulatable Threshold Decryption from LWE*

Abtin Afshar
UW-Madison†

Rishab Goyal
UW-Madison‡

Abstract

We construct the first multi-key fully homomorphic encryption (MKFHE) scheme where the ciphertext size, public key size, and secret key size remain independent of the number of users, N. Our construction is leveled and relies on the standard Learning with Errors (LWE) assumption. All prior MKFHE schemes incur at least linear growth in ciphertext size with the number of users ($|\mathsf{ct}| \propto N$), a limitation that has persisted across more than a decade of research. Our results provide the first evidence that MKFHE with constant ciphertext size is achievable under standard assumptions and pave the way for many interesting applications.

We also describe a single-round distributed decryption protocol for multi-key ciphertexts in our fully-succinct MKFHE scheme. More remarkably, we show that our MKFHE scheme simultaneously satisfies the following properties: (1) the size of each user's partial decryption share is identically equal to the plaintext length (i.e., partial decryption shares are truly rate-1), and (2) an honest user's partial decryption can be simulated. To the best of our knowledge prior to this work, we did not have any MKFHE with one-round distributed decryption from standard assumptions that simultaneously satisfied both these properties. We show that our MKFHE is significantly useful in designing various forms of multi-party computation (MPC) protocols with asymptotically optimal communication complexity.

*© IACR 2026. This article is a minor revision of the version published by Springer-Verlag available at https://doi.org/10.1007/978-3-032-35374-0_19. The proceedings version of this work (CRYPTO 2026) contains a bug in our construction of rate-1 simulatable threshold decryption. Nevertheless, we fix it in this version using correlation intractable hash functions, at the cost of a trusted setup (see Section 5).

†Email: abtin@cs.wisc.edu.

‡Email: rishab@cs.wisc.edu. Support for this research was provided by a gift from the Stellar Development Foundation.

Contents

1 Introduction **1**
- 1.1 Our results . . . 1
- 1.2 Applications . . . 2

2 Technical Overview **5**
- 2.1 Fully Succinct MKFHE . . . 5
- 2.2 Rate-1 Simulatable Threshold Decryption . . . 8
- 2.3 Threshold TDH . . . 13
- 2.4 Related Works . . . 14

3 Preliminaries **15**
- 3.1 Lattice Preliminaries . . . 16
- 3.2 Multi-Key Fully Homomorphic Encryption . . . 17
  - 3.2.1 MKFHE With Single-Round Distributed Decryption . . . 19
- 3.3 Succinct Non-Interactive Oblivious Tensor Evaluation . . . 20
- 3.4 Correlation Intractable Hash Functions . . . 22

4 Fully-Succinct Multi-Key Fully Homomorphic Encryption **23**
- 4.1 Construction . . . 24
- 4.2 Analysis . . . 26
- 4.3 Fully-Succinct MKFHE With Multi-Hop Evaluation . . . 29
- 4.4 Achieving Circuit Privacy . . . 30
- 4.5 Rate-1 Fully-Succinct MKFHE . . . 31

5 Fully-Succinct MKFHE With Rate-1 Simulatable Distributed Decryption **32**
- 5.1 Construction . . . 33
- 5.2 Analysis . . . 35

6 Threshold Trapdoor Hashing **45**
- 6.1 Definition . . . 45
- 6.2 Construction . . . 46
- 6.3 Analysis . . . 47

A Fully-Succinct MKFHE With Circuit Privacy **54**
- A.1 Construction . . . 54
- A.2 Analysis . . . 55

1 Introduction

Fully Homomorphic Encryption (FHE), first envisioned by Rivest, Adleman, and Dertouzos in 1978 [RAD+78], enables computation on encrypted data without decryption or access to raw data. Since Gentry's breakthrough in 2009, FHE has seen rapid development with improved efficiency and expanded functionality.

A natural and highly useful extension of FHE is multi-key FHE (MKFHE) [LATV12, CM15, MW16], which supports homomorphic evaluation on ciphertexts encrypted under different keys. In particular, given a set of N ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_N$, each encrypted under a distinct public key $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ (respectively), MKFHE lets an evaluator perform arbitrary homomorphic computation on the set of N ciphertexts (to compute an evaluated ciphertext $\widetilde{\mathsf{ct}}$) without any secret key. MKFHE has found broad applications, including on-the-fly and round-optimal multiparty computation protocols [AJLA+12a, LATV12, MW16], privacy-preserving distributed machine learning [GBDL+16, JVC18], multi-client private decision tree classification [BPTG14, WFNL15], multiparty quantum computation [CGS02, BOCG+06, BFK09, Goy18, DGJ+20], and thresholdizing cryptographic primitives [BGG+18].

Despite its vast utility, all existing MKFHE constructions [LATV12, CM15, MW16, BP16, PS16, DHRW16, CO17, BHP17, CZW17, CDKS19, DGMR22, KKL+23] suffer from a critical drawback: *the size of ciphertexts after homomorphic evaluation grows polynomially with the number of users,* N. This $\mathsf{poly}(N)$-barrier is, both, theoretically dissatisfying and practically discouraging. From a theoretical viewpoint, a polynomial dependence on N is quite far from an ideal aspiration of ciphertext size being independent of N. On a practical note, various implementations [HS14, HS21, CGGI20] of (single-key) FHE already indicate space complexity as a major bottleneck across various applications, thus reducing the size overhead is important for making MKFHE practical in the future. Given there are no known implausibility or hardness results that indicate overcoming this barrier is infeasible, we study the following question.

Can we design MKFHE where ciphertext size does not scale with N, the number of parties?

We refer to an MKFHE where the ciphertext size scales only poly-logarithmically with N as a "fully-succinct" MKFHE.

What do we know? To date, we do not have any MKFHE scheme that overcomes this $\mathsf{poly}(N)$-barrier. Answering the above has been discussed as an important open problem in the area [MW16, Wic15] for the past decade. To the best of our knowledge, we do not have a construction for fully-succinct MKFHE even if we rely on heavy cryptographic machinery such as indistinguishability obfuscation [BGI+01].

1.1 Our results

Our first main result is the construction of the first *fully-succinct* MKFHE scheme under the hardness of the standard Learning with Errors (LWE) assumption [Reg09]. As is common in the literature, our MKFHE scheme based on standard LWE is a leveled scheme; under the circular security of LWE, we can improve it to be an unleveled scheme. In addition to succinct evaluated ciphertexts, our construction achieves full succinctness in all other MKFHE parameters. These succinctness guarantees concern the final representation sizes of the scheme, while the homomorphic evaluation procedure still inherits the N-dependent run time and space complexity of the

underlying MKFHE evaluation. Reducing this overhead remains an important open problem. As an additional refinement, we can realize fully-optimal ciphertexts, where the evaluated ciphertext length asymptotically matches the plaintext length.

Our second main result concerns single-round distributed decryption for MKFHE. We address the problem of simultaneously achieving *optimal partial decryption size* and *simulatability of partial decryptions*. Prior LWE-based constructions achieved these properties separately, but not simultaneously. The subtle point is that simulatability is a per-instance guarantee: it must hold for every fixed choice of keys, circuit, and evaluated ciphertext, with probability only over the randomness of partial decryption and simulation. In a rounding-based decryption procedure with nonzero correctness error, as in [DHRW16], the real partial decryption is tied to the actual noisy decryption value, whereas the simulator is given only the target plaintext. Thus, when the noisy value crosses the rounding boundary, the real share may be consistent with an incorrectly decoded value, while the simulated share is generated to be consistent with the target plaintext. This prevents simulation for such fixed ciphertexts, even if they arise only with negligible probability in honest executions. We elaborate on this point in the technical overview.

We fill this gap by constructing an MKFHE scheme with single-round distributed decryption that achieves *rate-1 simulatable threshold decryption*: each user's partial decryption share has length identically equal to the plaintext length, and an honest user's partial decryption can be simulated. This improvement comes at the cost of a trusted setup and higher decryption complexity, which grows with the number of parties. We leave removing the trusted set up, reducing this complexity, or finding alternative routes to smaller simulatable partial decryptions, as an interesting open problem. Moreover, this contribution is orthogonal to the fully-succinct ciphertext result above, and the two can be combined generically to obtain MKFHE with fully-succinct parameters and rate-1 simulatable threshold decryption.

While our construction is not optimized for practicality due to heavy bootstrapping machinery and non-black-box techniques, it establishes new theoretical boundaries for MKFHE and opens the door to future efficiency improvements. Our results provide the first evidence that MKFHE with constant ciphertext size is achievable and enable many interesting applications. We summarize our results and compare them with prior LWE-based MKFHE schemes in Table 1.

1.2 Applications

Our MKFHE scheme leads to significant improvement to a number of known applications as well as enables new results in rate-1 threshold cryptography. These include on-the-fly MPC and two-round MPC with optimal communication complexity, interaction-preserving compilers for general MPC, threshold trapdoor hash functions, threshold lossy trapdoor functions, and more.

I. On-the-fly MPC. The phrase "on-the-fly" MPC was coined by Lopez-Alt, Tromer, and Vaikuntanathan [LATV12] as a special type of server-aided MPC [KMR11], where a computationally-powerful but untrusted cloud server is available to run the bulk of the computation for computationally-weak clients. They proposed an elegant template using MKFHE to design on-the-fly MPC. Their idea was to let each client encrypt its own data, then have the cloud server do (multi-key) homomorphic evaluation, and finally have each client threshold decrypt the evaluated ciphertext. A distinct advantage of their template, over traditional server-aided MPC protocols [KMR11, HLP11, KMR12, BB16, MOR16], was that the clients' work does not grow with the size of the computation. However, since prior MKFHE constructions suffered from the $\mathsf{poly}(N)$ overhead, the resulting

Scheme	$\lvert\mathsf{pk}\rvert + \lvert\mathsf{sk}\rvert$	$\lvert\widetilde{\mathsf{ct}}\rvert$	ThrDec	$\lvert p_i\rvert$	Sim
[LATV12]	$\mathsf{P}(\lambda)$	$\ell \cdot \mathsf{P}(\lambda, 2^N)$	No	-	-
[CM15, MW16, PS16]	$\mathsf{P}(\lambda)$	$\ell N^2 \cdot \mathsf{P}(\lambda)$	Yes	$\ell \cdot \mathsf{P}(\lambda)$	Yes
[BP16]	$\mathsf{P}(\lambda)$	$\ell N \cdot \mathsf{P}(\lambda)$	Yes	$\ell \cdot \mathsf{P}(\lambda)$	Yes
[DHRW16]	$\mathsf{P}(\lambda)$	$\ell N \cdot \mathsf{P}(\lambda)$	Yes	ℓ	No
[DGMR22]	$\mathsf{P}(\lambda, \ell)$	$\ell + N \cdot \mathsf{P}(\lambda)$	No	-	-
Construction 4.7	$\mathsf{P}(\lambda)$	$\ell + \mathsf{P}(\lambda)$	Yes	$\ell \cdot \mathsf{P}(\lambda)$	No
Construction 5.1	$\mathsf{P}(\lambda)$	$\ell \cdot \mathsf{P}(\lambda)$	Yes	ℓ	Yes

Table 1: Table of comparison for MKFHE parameters. We write $\mathsf{P}(\cdot)$ for $\mathsf{poly}(\cdot)$. In the table, ThrDec indicates threshold-decryption support, $|p_i|$ denotes the size of partial decryptions, and Sim indicates whether the partial decryption values can be simulated. All listed schemes rely on plain LWE (not Ring-LWE). As MKFHE parameters typically grow with the evaluation depth d, we omit this dependence from the table. Finally, we note that [DHRW16] also construct an MKFHE scheme with rate-1 simulatable threshold decryption from $i\mathcal{O}$, but the ciphertexts are not fully-succinct.

protocols do not yield a compact common encrypted result when the cloud is required to publish a single ciphertext for all clients.

Consider the following motivating scenario, *distributed malware detection*, taken nearly verbatim from the original on-the-fly MPC paper [LATV12]. Suppose when a malware is detected by an organization, its security analysts want to know which other organizations have observed similar malware. It is undesirable for them to publicize all malware they have detected. Using on-the-fly MPC, each organization can build an ever-growing dataset of malware it has detected, stored encrypted on a cloud service: whenever it detects a new sample, the company simply encrypts it with its own public key and sends it to the cloud. The cloud service, on a daily basis, gathers the encrypted datasets of its current subscribers, evaluates a comparison function which scans these datasets for similar malware samples using some executable-code similarity metrics, and sends the *short encrypted results* to those subscribers, for (potentially interactive but cheap) decryption.

Observe that if the size of an evaluated ciphertext in MKFHE scales with the number of encrypting parties, then the above protocol does not really give a "short common encrypted result" in a broadcast-channel implementation. Suppose all academic institutions in the United States decide to use CrowdStrike or Palo Alto Networks for malware sharing, then the state of the art on-the-fly MPC system would incur at least a $1000\times$ overhead (assuming 1000 institutions participate and MKFHE ciphertext scales linearly with N). Using our MKFHE, we can design truly-compact on-the-fly MPC, in this broadcast setting, thereby removing this overhead. We note that in a point-to-point implementation, prior schemes can already avoid sending the full evaluated ciphertext to each client by sending only the corresponding ciphertext block.

II. Optimal two-round MPC. One of the main challenges in MPC research is to optimize the efficiency of MPC protocols. It is well established that, for general computation, there is a simple lower bound of 2 rounds in MPC [HLP11]. Starting with the beautiful work of Mukherjee and Wichs [MW16], it has been well established that two-round MPC is achievable under "standard" cryptographic assumptions. Despite an incredible amount of research over the last decade in designing two-round MPC, it is still a challenging task to design a two-round MPC protocol from

standard assumptions where the total number of bits sent by each party is *asymptotically* equal to their input length plus the output length. That is, suppose $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_N} \to \{0,1\}^k$ is an N-party MPC function that a group of N parties want to evaluate. A natural question to ask is: *can we design a two-round MPC protocol for any such computation F, where the total number of bits being broadcast by the i-th party is equal to $\ell_i + k + \mathsf{poly}(\lambda)$?*

By plugging in our MKFHE scheme with rate-1 simulatable threshold decryption in the Mukherjee-Wichs [MW16] template, we can design a semi-malicious two-round MPC protocol for general computation with a trusted setup, where the i-th party broadcasts $\ell_i + k + \mathsf{poly}(\lambda)$ number of bits. Moreover, by applying the recent rate-preserving semi-malicious-to-malicious MPC compiler by Afshar et al. [ACG+24], this can be generically boosted to a maliciously-secure MPC protocol.

III. Interaction-Preserving MPC Compiler. Consider the following question — what is the cost of security in MPC? In particular, given an insecure MPC protocol where parties exchange L bits in T rounds, is it possible to design a "secure" protocol where parties exchange $L + \mathsf{poly}(\lambda)$ bits in T rounds?

The above question was first put forth by Döttling et al. [DGMR22], who designed a generic compiler using MKFHE to boost any insecure MPC protocol to a secure protocol where they needed an additional round of interaction, but their secure protocol required exchanging *almost* the same number of bits between parties.[1] Inspecting their compiler and final result carefully, we observe that the *additional* number of bits that each party must exchange in each round in their secure protocol grows as $\mathsf{poly}(N, \lambda)$, where N is the number of parties. By plugging in our MKFHE scheme, we can reduce the number of additional bits to just $\mathsf{poly}(\lambda)$, thereby making it truly optimal. We leave the problem of reducing the additional round of interaction with similar communication overhead under standard assumptions as an interesting open problem.

IV. Threshold Trapdoor Hash Functions. Trapdoor hash functions (TDHs) [DGI+19] are specialized hash functions with additional trapdoor-function-like properties. Specifically, a hash function $H : \{0,1\}^\ell \to \{0,1\}^\lambda$ is a TDH if, for any boolean function $f : \{0,1\}^\ell \to \{0,1\}$, one can sample an encoding key ek (that hides f) along with a corresponding trapdoor td with the following property. For any input $x \in \{0,1\}^\ell$, given $H(x)$ and a hint value $\mathsf{Enc}(\mathsf{ek}, x)$, a decoder can efficiently recover $f(x)$ given trapdoor td. A central question in the study of TDH is how small can the hint value, $\mathsf{Enc}(\mathsf{ek}, x)$, be? This is captured as the rate of the TDH. Prior works [DGI+19, BKM20, GHO20, BJSS25] have demonstrated numerous applications of TDHs to rate-1 oblivious transfer, private information retrieval, lossy trapdoor functions, and more.

In this work, we generalize this concept to threshold trapdoor hash functions (TTDHs), where multiple parties can asynchronously sample their individual encoding key ek_i along with a trapdoor td_i. And, we desire the TDH-like property of efficient recovery of $f(x)$ to hold as long as all trapdoors, $\{\mathsf{td}_i\}_i$, are available to the decoder. Essentially, we want to distribute the TDH trapdoor to multiple parties. We show that using our MKFHE, we can design a threshold TDH with a special feature – *silent setup* (i.e., parties holding trapdoors do not need to interact to set up). This directly leads to new threshold constructions of many primitives such as: threshold rate-1-batch OLE/OT, threshold lossy trapdoor functions, threshold PIR. We point out that in this work we only focus on N-out-of-N thresholds, and leave the problem of general threshold as an interesting research direction.

[1] Döttling et al. [DGMR22] also gave an improved compiler where they did not need any additional rounds, but assumed strong (non-falsifiable) assumptions such as extractable witness encryption [GGSW13, GKP+13].

2 Technical Overview

A multi-key fully homomorphic encryption (MKFHE) scheme extends the standard FHE functionality to the setting of multiple independently generated keys. In such a scheme, each user i generates a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ such that for any subset of N users[2], given ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mu_i)$, an evaluator can homomorphically compute an evaluated ciphertext $\widetilde{\mathsf{ct}} \leftarrow \mathsf{Eval}(C, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]})$, and decryption under the joint secret keys recovers the circuit value $C(\mu_1, \ldots, \mu_N) = \mathsf{Dec}((\mathsf{sk}_i)_{i\in[N]}, \widetilde{\mathsf{ct}})$.

A single-round distributed decryption for MKFHE further allows each user i to independently compute a partial decryption p_i of the evaluated ciphertext as $p_i \leftarrow \mathsf{PartDec}(\widetilde{\mathsf{ct}}, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i)$ and the final message is recovered from the collection of partial decryptions via $\mu \leftarrow \mathsf{FinDec}(p_1, \ldots, p_N)$.

The rest of this overview is divided into four parts. In Section 2.1 we present our construction of fully succinct MKFHE. In Section 2.2 we present a threshold-decryption wrapper for any perfectly correct MKFHE scheme with a sufficiently shallow decryption circuit (as in our fully-succinct construction). Finally, in Section 2.3 we discuss threshold TDHs as an application of our fully-succinct MKFHE, and in Section 2.4 we discuss related works.

2.1 Fully Succinct MKFHE

All prior constructions of MKFHE (e.g., [CM15, MW16]) follow a block-based paradigm. Namely, when the ciphertexts ct_k under different keys $(\mathsf{pk}_k, \mathsf{sk}_k)$ are evaluated together, first, each ciphertext ct_k is expanded to N blocks, one for each user, and then the evaluation happens in the expanded dimension. Concretely, an expanded ciphertext $\overline{\mathsf{ct}}_k$ has the form $\overline{\mathsf{ct}}_k = (\overline{\mathsf{ct}}_k^{(1)} \| \ldots \| \overline{\mathsf{ct}}_k^{(N)})$ that looks like a ciphertext under an expanded key, namely, each block $\overline{\mathsf{ct}}_k^{(i)}$ contains the terms necessary for decryption under sk_i. This expansion is enabled by appending additional linear combinations of FHE ciphertext components to the MKFHE ciphertext. After such an expansion, the evaluation will be similar to an FHE evaluation.

Suppose circuit C has ℓ bits of output. Then at the end of evaluation, every user i has a dedicated block $\widetilde{\mathsf{ct}}_j^{(i)}$ for each $j \in [\ell]$ of the evaluated ciphertext

$$\widetilde{\mathsf{ct}} = \left(\left(\widetilde{\mathsf{ct}}_1^{(1)} \| \ldots \| \widetilde{\mathsf{ct}}_1^{(N)} \right), \ldots, \left(\widetilde{\mathsf{ct}}_\ell^{(1)} \| \ldots \| \widetilde{\mathsf{ct}}_\ell^{(N)} \right) \right)$$

aligned with its secret key sk_i which ensures that decryption for each output bit $j \in [\ell]$ can be performed locally. This feature is present in all constructions and it is hard to get around this. This is the ciphertext expansion barrier due to which at least linear and sometimes quadratic dependence in N appears in the evaluated ciphertext[3].

Our approach. We revisit MKFHE through the lens of secure two-party computation (2PC). Think of decryption as a protocol between an *evaluator*, holding the evaluated ciphertext $\widetilde{\mathsf{ct}}$, and the *decrypter side* (instantiated by the key-generation and decryption algorithms), holding the secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$. If this protocol could simultaneously (i) preserve the privacy of the decrypters' keys and (ii) be fully *succinct* on the evaluator's side, then one might compress the evaluated ciphertext without sacrificing security or correctness.

[2] There may be a larger universe of users of size N', but any subset of N users should be able to perform evaluation and decryption. For simplicity, we relabel the participating subset as $[N]$. Throughout the overview, we treat $\log N$ as $\mathsf{poly}(\lambda)$.

[3] In some schemes, further cross-user expansion terms are added for correctness, leading to quadratic dependence.

A direct implementation of the above idea fails to respect the asynchronous and reusable nature of MKFHE. Concretely, consider a system with a large (and possibly evolving) universe of users $[N']$. In MKFHE, *any* subset $S = \{i_1, \ldots, i_N\} \subseteq [N']$ (chosen at evaluation time) may evaluate its ciphertexts to obtain an output ciphertext $\widetilde{\mathsf{ct}}_S$. Therefore, any first message produced by the decrypter side at key-generation time must be fixed *before* S (and hence $\widetilde{\mathsf{ct}}_S$) is known, yet still support decryption and be reusable for every possible S. This effectively forces the first message to incorporate, or depend on, all secret keys $(\mathsf{sk}_1, \ldots, \mathsf{sk}_{N'})$ in advance. Hence, our starting point seems infeasible even for a static N', and becomes only more problematic in a dynamic system where users may join after the first message is generated.

However, the very reason that limits the prior works, i.e., the ciphertext and decryption structure, turns out to be an advantage to us. Namely, the standard block-based MKFHE already decomposes decryption: for each output bit $j \in [\ell]$, user i only needs to use sk_i to decrypt their designated slot $\widetilde{\mathsf{ct}}_j^{(i)}$. Thus, decryption is conceptually a collection of N tiny 2PCs in parallel, and one per user. This avoids the above issues since each user i only needs to interact with their own slots $(\widetilde{\mathsf{ct}}_1^{(i)}, \ldots, \widetilde{\mathsf{ct}}_\ell^{(i)})$.

First attempt (still linear in N). We start by instantiating each tiny 2PC with a succinct protocol. On the decrypter side, each user i prepares at key generation a hiding encoding $\hat{\mathsf{sk}}_i$ and a private state st_i. The evaluator given encoded secret keys $(\hat{\mathsf{sk}}_i)_{i\in[N]}$, computes tuples $(\mathsf{d}_i, z_i)_{i\in[N]}$ (one per tiny 2PC), where d_i is a succinct hash of user i's slots $\widetilde{\mathsf{ct}}^{(i)} = (\widetilde{\mathsf{ct}}_1^{(i)} \| \cdots \| \widetilde{\mathsf{ct}}_\ell^{(i)})$, and z_i is an additive (per-user) decryption share of $\widetilde{\mathsf{ct}}^{(i)}$.[4] Next, it combines $(z_1, \ldots, z_N)$ to compute an aggregated decryption share z of $\widetilde{\mathsf{ct}}$, and sends $(z, \mathsf{d}_1, \ldots, \mathsf{d}_N)$ to the decrypter side. Then, the private state st_i is used at decryption time to locally compute $z_i' \leftarrow \mathsf{Recover}(\mathsf{d}_i, \mathsf{st}_i)$, a complementary decryption share of $\widetilde{\mathsf{ct}}^{(i)}$. Finally, the decrypter puts together $(z, z_1', \ldots, z_N')$ to recover the whole message. This removes the dependency on ℓN^2 but still requires N digests.

How to remove the linear dependency on N? a single, reusable digest. We split the evaluator's task into two pieces: (a) a reusable succinct digest d computed as $(\mathsf{st}, \mathsf{d}) \leftarrow \mathsf{Hash}(\widetilde{\mathsf{ct}})$ that depends only on $\widetilde{\mathsf{ct}}$ (and not on any user), and (b) a public partial value z_i derivable from st and $\hat{\mathsf{sk}}_i$. By doing so, we have a single digest d, and all we are left with is instantiating the above protocol with a fully succinct 2PC, with appropriate functionality.

Instantiation via OTE (inner products suffice). Our next observation, which is crucial to our construction, is that the decryption of each block in the prior MKFHE work is only an almost *linear* procedure. More specifically, an MKFHE decryption is simply an inner product of the expanded secret keys with the evaluated ciphertext, namely:

$$C_j(\mu_1, \ldots, \mu_N) \;=\; \left\lceil \langle (\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \widetilde{\mathsf{ct}}_j \rangle \right\rfloor_2 \;=\; \left\lfloor \sum_{i=1}^{N} \left\langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_j^{(i)} \right\rangle \right\rfloor_2 .$$

[4] To avoid confusion, we reserve the term *partial decryption* for the standard distributed decryption algorithm $\mathsf{PartDec}$. In our fully succinct constructions, we instead use *decryption shares* to refer to values (e.g., z and z') that additively combine to reveal the plaintext.

where C_j is the j^{th} output bit of C. Therefore, each user's decryption algorithm is just an inner product $\langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_j^{(i)} \rangle$ for each $j \in [\ell]$.

Thanks to both the block-by-block and the near-linear decryption procedure of prior MKFHE constructions, we are able to combine the non-interactive succinct oblivious tensor evaluation (OTE) scheme recently introduced and constructed in [AMR25] with a standard MKFHE framework to obtain our results. Next, we describe the OTE scheme and highlight how it aligns with the decryption structure of a standard MKFHE.

An OTE scheme allows two parties with inputs $\mathbf{y} \in \mathbb{Z}_q^{\ell_\mathbf{y}}$ and $\mathbf{x} \in \mathbb{Z}_q^{\ell_\mathbf{x}}$ to non-interactively compute an additive secret sharing of

$$\mathbf{y} \otimes \mathbf{x} = [y_1\mathbf{x}, \ldots, y_{\ell_\mathbf{y}}\mathbf{x}] = [y_1x_1, \ldots, y_1x_{\ell_\mathbf{x}}, \ldots, y_{\ell_\mathbf{y}}x_1, \ldots, y_{\ell_\mathbf{y}}x_{\ell_\mathbf{x}}].$$

More formally, in an OTE the hasher computes $(\mathsf{d}, \mathsf{st}_\mathbf{y}) \leftarrow \mathsf{Hash}(\mathbf{y})$ where d is independent of $|\mathbf{y}|$, and the encoder computes $(\hat{\mathbf{x}}, \mathsf{st}_\mathbf{x}) \leftarrow \mathsf{Encode}(\mathbf{x})$, where $\hat{\mathbf{x}}$ hides $\mathbf{x}$, s.t.

$$\mathbf{v} \leftarrow \mathsf{HashEval}(\hat{\mathbf{x}}, \mathsf{st}_\mathbf{y}), \qquad \mathbf{w} \leftarrow \mathsf{EncodeEval}(\mathsf{d}, \mathsf{st}_\mathbf{x}), \qquad \mathbf{v} + \mathbf{w} = \mathbf{y} \otimes \mathbf{x}.$$

A tensor product crucially enumerates all degree-2 cross terms between $\mathbf{y} \in \mathbb{Z}_q^{\ell_\mathbf{y}}$ and $\mathbf{x} \in \mathbb{Z}_q^{\ell_\mathbf{x}}$: for every $i \in [\ell_\mathbf{y}]$ and $j \in [\ell_\mathbf{x}]$, $(\mathbf{y} \otimes \mathbf{x})[(i-1)\ell_\mathbf{x} + j] = y_i x_j$. Let OTE output shares $\mathbf{v}, \mathbf{w}$ with $\mathbf{v} + \mathbf{w} = \mathbf{y} \otimes \mathbf{x}$, and define $v_{i,j} := \mathbf{v}[(i-1)\ell_\mathbf{x} + j]$ and $w_{i,j} := \mathbf{w}[(i-1)\ell_\mathbf{x} + j]$; then $v_{i,j} + w_{i,j} = y_i x_j$. In other words, OTE produces additive secret sharings of all degree-2 cross terms between $\mathbf{y}$ and $\mathbf{x}$. Consequently, for any publicly computable function f that depends on a subset of these cross terms and is additively homomorphic (i.e., $f(z_1 + z_2) = f(z_1) + f(z_2)$), we can recover:

$$f\big((y_i x_j)_{i,j}\big) = f\big((v_{i,j})_{i,j}\big) + f\big((w_{i,j})_{i,j}\big).$$

Now let $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{Z}_q^n$ and $\mathbf{y} = \big(y_{j,k}^{(i)}\big)_{k \in [n],\, i \in [N],\, j \in [\ell]} \in \mathbb{Z}_q^{nN\ell}$. Running OTE on each pair $(\mathbf{y}, \mathbf{x}_i)$ yields $(\mathbf{v}_i, \mathbf{w}_i)$ sharing $\mathbf{y} \otimes \mathbf{x}_i$. In our application we are specifically interested in *aligned products* $\big(y_{j,k}^{(i)} x_{i,k}\big)_{k,i,j}$, and f simply sums nN distinct such terms per output index j, so f is additively homomorphic and can be recovered from the shares. Since $\mathbf{y}$ is the same for all i, the hasher-side message (the digest d) is computed once and reused across all N instances.

Our Construction. We instantiate the above template by setting $\mathbf{y} = \widetilde{\mathsf{ct}}$ and $\mathbf{x}_i = \mathsf{sk}_i$. For each $i \in [N]$, OTE yields $(\mathbf{v}_i, \mathbf{w}_i)$, which jointly provide additive shares of cross terms between $\widetilde{\mathsf{ct}}$ and sk_i; in our application we only use a subset of *aligned products* $\{\widetilde{\mathsf{ct}}_j^{(i)}[k] \cdot \mathsf{sk}_i[k]\}_{k \in [n],\, j \in [\ell]}$. From $\mathbf{v}_i$ and $\mathbf{w}_i$ we can therefore obtain $\big(\langle \widetilde{\mathsf{ct}}_j^{(i)}, \mathsf{sk}_i \rangle\big)_{j \in [\ell]}$. Thus, a single succinct digest of $\widetilde{\mathsf{ct}}$, together with hiding encodings of $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$, suffices to derive all required products $\{\widetilde{\mathsf{ct}}_j^{(i)}[k] \cdot \mathsf{sk}_i[k]\}_{k,i,j}$, and thereby reconstruct $\big(\sum_{i=1}^N \langle \widetilde{\mathsf{ct}}_j^{(i)}, \mathsf{sk}_i \rangle\big)_{j \in [\ell]}$. This matches exactly the block-wise inner-product decryption structure of standard MKFHE, and is why OTE fits seamlessly into our construction. We present our construction of fully-succinct MKFHE as follows:

- any user i encodes its secret key to get $(\hat{\mathsf{sk}}_i, \mathsf{st}_{\mathsf{sk},i})$,
- the evaluator hashes the evaluated ciphertext $\widetilde{\mathsf{ct}}$ to get $(\mathsf{d}, \mathsf{st}_{\widetilde{\mathsf{ct}}})$, then computes $\mathbf{v}^{(i)} \leftarrow \mathsf{HashEval}(\hat{\mathsf{sk}}_i, \mathsf{st}_{\widetilde{\mathsf{ct}}})$. Let $v_{j,k}^{(i)}$ be the slot of $\mathbf{v}^{(i)}$ that corresponds to the j^{th} output bit and

k^{th} index of the inner product computation $\langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_j^{(i)} \rangle$. In other words, $v_{j,k}^{(i)}$ is the element in the tensor evaluation corresponding to the multiplication of $\mathsf{sk}_i[k]$ and $\widetilde{\mathsf{ct}}_j^{(i)}[k]$. Then the evaluator computes $z_j = \sum_{i=1}^{N} \sum_{k=1}^{n} v_{j,k}^{(i)}$ for $j \in [\ell]$ (where n is the secret key dimension, i.e., $\mathsf{sk}_i \in \mathbb{Z}_q^n$) and outputs $(z = (z_1, \ldots, z_\ell), \mathsf{d})$ as the new evaluated ciphertext,

- the decrypter similarly computes $\mathbf{w}^{(i)} \leftarrow \mathsf{EncodeEval}(\mathsf{d}, \mathsf{st}_{\mathsf{sk},i})$ for $i \in [N]$, analogously defines $w_{j,k}^{(i)}$, and then computes $z'_j = \sum_{i=1}^{N} \sum_{k=1}^{n} w_{j,k}^{(i)}$ for $j \in [\ell]$, and outputs $\left(\left\lceil z_j + z'_j \right\rfloor_2 \right)_{j \in [\ell]}$.

Plugging OTE into our template as above leads to a fully-succinct digest d whose size is independent of N (and ℓ). Moreover, using similar rounding ideas as [DHRW16], we can further compress the aggregated decryption share z to be exactly ℓ bits (instead of $\ell \cdot \log q$ bits), which leads to a rate-1 ciphertext size. Furthermore, we show how to make minor modifications to our construction, and carefully use the additive secret-sharing of the output between z and z', in addition to smudging techniques to get circuit privacy for free. While the transformation to a circuit private MKFHE is only presented for this construction, it can be applied to all our constructions in this paper. This transformation is presented in Appendix A and could be of independent interest.

Note that our construction admits simple single-round distributed decryption, but its partial decryptions are not rate-1, and since achieving rate-1 and simulatability simultaneously is also our goal, we defer achieving the simulatability property to the next section where we construct rate-1 threshold decryption.[5]

2.2 Rate-1 Simulatable Threshold Decryption

In this section, for simplicity we assume that N is a power of 2 and we focus on evaluation circuits with a single-bit output $\tilde{\mu} \in \{0, 1\}$. We write $\widetilde{\mathsf{ct}}_i$ for the i^{th} party's block of the evaluated ciphertext $\widetilde{\mathsf{ct}}$. Recall that party i computes its (distributed) partial decryption share as $p_i \leftarrow \mathsf{PartDec}(\widetilde{\mathsf{ct}}, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i)$. We say that partial decryption is *simulatable* if there exists an efficient simulator $p'_i \leftarrow \mathcal{S}\big(\tilde{\mu}, \widetilde{\mathsf{ct}}, C, i, (\mathsf{sk}_k)_{k \in [N] \setminus \{i\}}\big)$ such that $p_i \overset{stat}{\approx} p'_i$. Importantly, this statistical indistinguishability must hold for each fixed instance (parameters, keys, circuit, evaluated ciphertext, and index), with randomness only from the partial decryption and the simulator algorithms.

Now suppose the underlying MKFHE scheme is not perfectly correct. Then there is a nonzero probability over the choice of the instance that an evaluated ciphertext $\widetilde{\mathsf{ct}}^*$ decrypts to the wrong value, i.e., $\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_N), \widetilde{\mathsf{ct}}^*) \neq \tilde{\mu}$. For such a ciphertext, party i's partial decryption share p_i is tied to the actual noisy decryption computation on $\widetilde{\mathsf{ct}}^*$; in particular, to whether the corresponding noisy sum landed on the correct or incorrect side of the final rounding step. However, the simulator for p_i is only given $\tilde{\mu}$ and the other parties' secret keys $(\mathsf{sk}_k)_{k \in [N] \setminus \{i\}}$; it does not get to observe which "side" $\widetilde{\mathsf{ct}}^*$ landed on. As a result, in the presence of correctness errors, the information

[5] In our construction if we replace the underlying MKFHE with the lattice-based Spooky encryption of [DHRW16] we get a fully succinct evaluated ciphertext and rate-1 threshold decryption. Moreover, by making some minor modifications to their construction (similar to our techniques in Section 2.2), we can apply the rounding lemma for one more round, which allows for applying our transformation from fully-succinct to rate-1 evaluated ciphertext. This approach achieves optimal evaluated ciphertext and partial decryptions. However, as we explain in Section 2.2 since the Spooky encryption of [DHRW16] does not satisfy perfect correctness, we cannot simulate the partial decryptions.

provided to the simulator can be insufficient to reproduce the distribution of p_i without either weakening the simulation guarantee or the correctness guarantee. This is the same obstruction that prevents Spooky encryption [DHRW16] from directly yielding rate-1 simulatable threshold decryption. Namely, as noted in Section 6.2 of [DHRW16], imperfect correctness in their LWE-based construction leads to an unsimulatability issue, and the problem of simultaneously achieving rate-1 and simulatability was left as an open problem in their work.

Simulatability vs. rate. Our starting point is the MKFHE scheme of [MW16], which supports simulatable partial decryption shares. Recall that the decryption in their scheme computes $C(\mu_1, \ldots, \mu_N) = \left\lceil \sum_{i=1}^{N} \langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_i \rangle \right\rfloor_2$; hence, by letting $p_i = \langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_i \rangle$, the final decryption can correctly recover the output as $\tilde{\mu} = \left\lceil \sum_{i=1}^{N} p_i \right\rfloor_2$. For simulatability, notice that $p'_i = q/2 \cdot \tilde{\mu} - \sum_{k \in [N] \setminus \{i\}} \langle \mathsf{sk}_k, \widetilde{\mathsf{ct}}_k \rangle$ differs from p_i only by the decryption noise term; therefore, one can smudge the real and simulated shares so that their distributions become statistically indistinguishable.

However, in this baseline, each share p_i is an element of $\mathbb{Z}_q$ and therefore has size $\Theta(\log q) = \mathsf{poly}(\lambda, d, \log N)$, which is far from the desired rate-1 threshold decryption. A natural idea is to compress shares using the rounding lemma of [DHRW16]. For two random additive shares y_1, y_2 of $q/2 \cdot \mu + e$, one can recover μ as $\lceil y_1 \rfloor_2 + \lceil y_2 \rfloor_2$ with overwhelming probability. Generalizing this lemma to N different random shares requires rounding to $O(\log N)$ most-significant bits of each y_i to recover μ with high probability. Therefore, one can improve the rate in [MW16] by letting $y_i = \langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_i \rangle$, and computing partial decryptions as $p_i = \lceil y_i \rfloor_{q'}$ for some modulus $q' = O(N)$. Then final decryption recovers the output as $\tilde{\mu} = \left\lceil \sum_{i \in [N]} p_i \right\rfloor_2$. This reduces the per-party communication to $O(\log N)$ bits. Our goal is to reduce this to be exactly one bit.

Structured rounding. For *unstructured* random additive shares, rounding-based compression appears to face information-theoretic limitations. This suggests exploiting additional structure in the shares. As a toy example, suppose that for every i the share y_i is already close to a binary multiple of $q/2$, i.e., $y_i = q/2 \cdot \lceil y_i \rfloor_2 + e'_i$ for some small e'_i. Then the sum $y = \sum_i y_i$ equals $\frac{q}{2} \cdot (\sum_i \lceil y_i \rfloor_2) + e$ with $e = \sum_i e'_i$, and if e remains sufficiently small, one can recover the output as $\tilde{\mu} = \sum_{i \in [N]} \lceil y_i \rfloor_2$.

Next, we extend the above argument from individual shares to groups of shares. Let $S_1, \ldots, S_K$ be a partition of $[N]$, and for any $S \subseteq [N]$, write $y_S := \sum_{i \in S} y_i$. Suppose that for every $k \in [K]$, $y_{S_k} = \frac{q}{2} \cdot \lceil y_{S_k} \rfloor_2 + e'_k$ for some small e'_k. Then, assuming the total error remains sufficiently small, one can recover the output as $\tilde{\mu} = \sum_{k \in [K]} \lceil y_{S_k} \rfloor_2$. We refer to the above property as a *roundable structure*.

Our approach. We combine the above roundable-structure viewpoint with the two-share rounding lemma of [DHRW16] to obtain the following structure-preserving splitting principle. Suppose we have a partition $S_1^{(t-1)}, \ldots, S_K^{(t-1)}$ of $[N]$ such that

$$\tilde{\mu} = \sum_{k \in [K]} \left\lceil y_{S_k^{(t-1)}} \right\rfloor_2, \qquad \text{and} \qquad y_{S_k^{(t-1)}} = \frac{q}{2} \cdot \left\lceil y_{S_k^{(t-1)}} \right\rfloor_2 + e'_k \quad \text{for each } k \in [K].$$

Then we can split each set $S_k^{(t-1)}$ into two subsets $S_{2k-1}^{(t)}$ and $S_{2k}^{(t)}$ (e.g., by splitting it into two halves) so that $\tilde{\mu} = \sum_{k\in[2K]} \left\lceil y_{S_k^{(t)}} \right\rfloor_2$. Intuitively, this follows by viewing $y_{S_{2k-1}^{(t)}}$ and $y_{S_{2k}^{(t)}}$ as two additive shares of $y_{S_k^{(t-1)}}$ and applying the rounding lemma as $\left\lceil y_{S_k^{(t-1)}} \right\rfloor_2 = \left\lceil y_{S_{2k-1}^{(t)}} \right\rfloor_2 + \left\lceil y_{S_{2k}^{(t)}} \right\rfloor_2$.

One might hope to apply the above splitting process iteratively in a binary-tree fashion defined over the partitions $\{S_k^{(t)}\}_{t\in[0,\log N],\, k\in[2^t]}$. Let $S_1^{(0)} = [N]$ be the root, and for each level $t \in [\log N]$ split every node $S_k^{(t-1)}$ into two children $S_{2k-1}^{(t)}$ and $S_{2k}^{(t)}$ (by halving $S_k^{(t-1)}$) until reaching the leaves where we have $S_k^{(\log N)} = \{k\}$. The goal is to define, for every node $S_k^{(t)}$, a corresponding value $y_{S_k^{(t)}}$ so that starting from the level-0 shares $y_i^{(0)} = \langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_i \rangle$, at level $t \in [0, \log N]$ we obtain $\tilde{\mu} = \sum_{k\in[2^t]} \left\lceil y_{S_k^{(t)}} \right\rfloor_2$. Crucially, for the template to apply recursively, the roundable structure must be preserved at every node $S_k^{(t)}$, i.e., each $y_{S_k^{(t)}}$ must remain close to a binary multiple of $q/2$.

Instantiating this template is a challenging task for three reasons. First, in distributed decryption the parties do not have access to the other parties' shares y_i, so they cannot explicitly compute $y_{S_k^{(t)}}$. Second, this process must be essentially deterministic (or publicly reproducible) so that all parties compute consistent partial decryptions. Third, a direct use of the rounding lemma is inherently "one-shot": after producing rounded values, it does not by itself provide the additional structure needed to continue splitting in subsequent levels. Next, we describe a structure-preserving splitting process that resolves the third issue, and then explain how standard techniques address the first two as well.

Instantiating our template. A useful way to view the roundable structure in our setting is through the ciphertext blocks themselves. Consider a subset $S \subseteq [N]$, and the sum $y_S = \sum_{i\in S} y_i$ where $y_i = \langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_i \rangle$. Then y_S behaves in the desired "$\frac{q}{2} \cdot (\cdot) +$ small error" form precisely when the block tuple $\widetilde{\mathsf{ct}}_S := (\widetilde{\mathsf{ct}}_i)_{i\in S}$ is a *complete ciphertext* for the user set S; namely, when $\widetilde{\mathsf{ct}}_S$ could have arisen as the output of a valid encryption/evaluation process run for exactly the parties in S (as opposed to being an arbitrary subcollection of blocks taken from a larger ciphertext).

Moreover, a splitting step $y_S \rightsquigarrow (y_{S_1}, y_{S_2})$ preserves the roundable structure if it produces new block tuples $\widetilde{\mathsf{ct}}'_{S_1}$ and $\widetilde{\mathsf{ct}}'_{S_2}$ that are themselves complete ciphertexts, and such that the split shares again have the canonical form $y'_i = \langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}'_i \rangle$ for each $i \in S$. In that case, the same reasoning applies recursively to S_1 and S_2.

Using the above ideas, an ideal splitting step for a node S would look as follows. Assume that $y_S = \sum_{i\in S} y_i = \frac{q}{2} \cdot \mu + e$ for some bit μ and small error e, and split S into two children S_1, S_2. One would like to define $\mu_k := \lceil y_{S_k} \rfloor_2$ for $k \in \{1, 2\}$ and then produce two *complete ciphertexts* ct'_{S_1} and ct'_{S_2} encrypting μ_1 and μ_2 under the corresponding public keys. Setting $y'_i = \langle \mathsf{sk}_i, \mathsf{ct}'_i \rangle$ would then preserve the roundable structure in each child and allow the recursion to continue.

Of course, this ideal procedure is not directly implementable: it requires knowing the hidden sums y_{S_1} and y_{S_2}, and it would be randomized. We fix both issues using bootstrapping [Gen09]. At key generation, each party i samples a fresh bootstrapping key pair $(\mathsf{pk}'_i, \mathsf{sk}'_i)$ and publishes an encryption $\mathsf{ct}_{\mathsf{sk},i} \leftarrow \mathsf{Enc}(\mathsf{pk}'_i, \mathsf{sk}_i)$. Now for any node S_k and its block tuple $\widetilde{\mathsf{ct}}_{S_k}$, let C_{S_k} be the

decryption circuit for ciphertext $\widetilde{\mathsf{ct}}_{S_k}$ corresponding to set S_k of keys, namely,

$$C_{S_k}((\mathsf{sk}_i)_{i\in S_k}) \coloneqq \mathsf{Dec}\left((\mathsf{sk}_i)_{i\in S_k}, \widetilde{\mathsf{ct}}_{S_k}\right) = \left\lceil \sum_{i\in S_k} \langle \mathsf{sk}_i, \widetilde{\mathsf{ct}}_i\rangle \right\rfloor_2 = \left\lceil \sum_{i\in S_k} y_i \right\rfloor_2 = \lceil y_{S_k} \rfloor_2 = \mu_k$$

where $\widetilde{\mathsf{ct}}_{S_k}$ is hardwired in C_{S_k}. Then we compute a fresh complete ciphertext for S_k as $\widetilde{\mathsf{ct}}'_{S_k} \leftarrow \mathsf{Eval}\big(C_{S_k}, (\mathsf{pk}'_i, \mathsf{ct}_{\mathsf{sk},i})_{i\in S_k}\big)$, parse $\widetilde{\mathsf{ct}}'_{S_k} = (\widetilde{\mathsf{ct}}'_i)_{i\in S_k}$ and set $y'_i \coloneqq \langle \mathsf{sk}'_i, \widetilde{\mathsf{ct}}'_i\rangle$ for $i \in S_k$. Since Eval is public and deterministic, this yields a structure-preserving splitting step.

Our construction. Given any perfectly correct MKFHE with sufficiently shallow decryption, we implement the above template via $\log N$ levels of bootstrapping. Let $y_i^{(t)} = \left\langle \mathsf{sk}_i^{(t)}, \widetilde{\mathsf{ct}}_i^{(t)} \right\rangle$ for $t \in [0, \log N]$ and $i \in [N]$ and $\mathsf{sk}_S^{(t)} = (\mathsf{sk}_i^{(t)})_{i\in S}$; our construction works as follows:

1. **Key generation.** Each party i in addition to the underlying MKFHE keys $(\mathsf{pk}_i, \mathsf{sk}_i)$, samples a fresh key pair $(\mathsf{pk}_i^{(t)}, \mathsf{sk}_i^{(t)})$ for every $t \in [0, \lambda]$, and publishes

$$\mathsf{ct}_{\mathsf{sk},i}^{(0)} \leftarrow \mathsf{Enc}(\mathsf{pk}_i^{(0)}, \mathsf{sk}_i), \qquad \mathsf{ct}_{\mathsf{sk},i}^{(t)} \leftarrow \mathsf{Enc}(\mathsf{pk}_i^{(t)}, \mathsf{sk}_i^{(t-1)}) \ \text{ for } t \in [\lambda].$$

2. **Partial decryption by party i^*.** Given $\widetilde{\mathsf{ct}}$ and $\mathsf{SK}_{i^*} = (\mathsf{sk}_{i^*}, (\mathsf{sk}_{i^*}^{(0)}, \ldots, \mathsf{sk}_{i^*}^{(\lambda)}))$, first bootstrap once to obtain an MW16-style block ciphertext $\widetilde{\mathsf{ct}}_{[N]}^{(0)} = (\widetilde{\mathsf{ct}}_1^{(0)}, \ldots, \widetilde{\mathsf{ct}}_N^{(0)})$ of the same output bit. Then, for all $t \in [\log N]$, let $S_1^{(t)}, \ldots, S_{2^t}^{(t)}$ be the canonical partition of $[N]$ into blocks of size $N/2^t$; then find k_t s.t. $i^* \in S_{k_t}^{(t)}$, and compute

$$\widetilde{\mathsf{ct}}_{S_{k_t}^{(t)}}^{(t)} \leftarrow \mathsf{Eval}\left(C_{k_t}^{(t)}, \ (\mathsf{pk}_i^{(t)}, \mathsf{ct}_{\mathsf{sk},i}^{(t)})_{i\in S_{k_t}^{(t)}}\right) \quad \text{where} \quad C_{k_t}^{(t)}(\mathsf{sk}_{S_{k_t}^{(t)}}^{(t-1)}) \coloneqq \mathsf{Dec}\left(\mathsf{sk}_{S_{k_t}^{(t)}}^{(t-1)}, \widetilde{\mathsf{ct}}_{S_{k_t}^{(t)}}^{(t-1)}\right)$$

s.t. $\widetilde{\mathsf{ct}}_{S_{k_t}^{(t)}}^{(t-1)}$ is hardwired in $C_{k_t}^{(t)}$; and note that $\mathsf{Dec}\left(\mathsf{sk}_{S_{k_t}^{(t)}}^{(t-1)}, \widetilde{\mathsf{ct}}_{S_{k_t}^{(t)}}^{(t-1)}\right) = \left\lceil \sum_{i\in S_{k_t}^{(t)}} y_i^{(t-1)} \right\rfloor_2$. In the end, the partial decryption algorithm outputs $p_{i^*} = \mathsf{Dec}\left(\mathsf{sk}_{i^*}^{(\log N)}, \widetilde{\mathsf{ct}}_{i^*}^{(\log N)}\right) = \left\lceil y_{i^*}^{(\log N)} \right\rfloor_2$.

3. **Final decryption.** Output $\tilde{\mu} = \sum_{i\in[N]} p_i$.

We emphasize that our wrapper is generic; however, in the main body we instantiate it on top of our fully succinct MKFHE.

Avoiding the rounding boundaries. If we implement the above template directly using the rounding lemma, then the two shares at each split are noisy, and for some fixed instances the rounded values are incorrect. This is not acceptable for simulatability, which must hold for every fixed instance. The standard way to handle this would be to smudge the shares. However, smudging is a randomized process, and since each decrypter computes the splitting tree on its own, different decrypters would obtain different trees, and their partial decryptions would not match.[6] We avoid

[6]This is the approach that was presented in the proceedings version [AG26] which led to a correctness error. We replace the smudging approach with the CIH-based approach in this revision.

this issue by shifting the two shares with a fixed public value (added to one share and subtracted from the other) that keeps them away from the rounding boundaries. Then every split is exact and the whole process is deterministic, so all decrypters compute the same tree, which gives both correctness and simulatability.

Choosing the shift via correlation intractability. To choose such a shift, we fix a set of possible shifts such that, for any value of the shares, at most one of them, which we call the bad shift, puts the shares close to a rounding boundary. The bad shift depends on the shares, which are hidden, so the decrypters cannot avoid it directly. Instead, the shift is chosen by a correlation intractable hash function (CIH) [CGH98, CCH+19, PS19], whose hash key can be generated for a circuit f so that the hash of any input x is never $f(x)$. To compute the bad shift from public values without revealing the shares, each party also publishes FHE encryptions of its secret keys, so that anyone can homomorphically compute an FHE encryption of a share, and the hash key is generated for the circuit that decrypts this ciphertext and outputs its bad shift. Generating this hash key requires the FHE secret key, which decrypts the encryptions of all the parties' secret keys, so a trusted setup generates it and then erases the FHE secret key. Finally, to guarantee this does not compromise the security, we use the CIH dual-mode keys, and switch to an indistinguishable key which is independent of the FHE secret key.

For simulatability, recall the binary tree structure, where each node $S_k^{(t)}$ is associated with the (noisy) value $y^{(t)}_{S_k^{(t)}} := \sum_{i \in S_k^{(t)}} \langle \mathsf{sk}_i^{(t)}, \widetilde{\mathsf{ct}}_i^{(t)} \rangle$. As in the correctness analysis, (a) the root value $y^{(0)}_{[N]}$ equals $\frac{q}{2} \cdot \tilde{\mu}$ plus a small error, and (b) for every internal node, the parent value is close to the sum of its two children values (up to a small error term). Now, fix a leaf index i^* and consider the path from the root $S_1^{(0)} = [N]$ to the leaf $S_{i^*}^{(\log N)} = \{i^*\}$. By repeatedly using (b), one can express the leaf value $y^{(\log N)}_{i^*}$ as the root value minus the sum of the values of all off-path sibling nodes, up to an accumulated small error. Therefore, a simulator that is given $\tilde{\mu}$ and the secret keys of all other parties can compute the off-path contributions, and since every split is exact, the simulated partial decryption equals the real one.

Instantiating via our fully-succinct MKFHE. Our fully-succinct MKFHE is obtained by combining (i) an underlying perfectly correct MKFHE (e.g., [MW16]) with (ii) an OTE scheme. Thus, the only nontrivial correctness issue is the OTE component: the construction of [AMR25] satisfies only a weaker notion of α-correctness. In the main body (see Section 3.3) we show how to obtain the perfect-correctness guarantee needed for our wrapper by slightly adjusting the correctness formulation and proving that the OTE of [AMR25] already meets it under standard lattice techniques.[7] Furthermore, the decryption circuit in our fully-succinct construction consists of linear operations in the secret-key hence can be implemented using a shallow circuit (see Section 4.3 for more details).

[7] The rounding techniques needed to obtain rate-1 evaluated ciphertexts appear to preclude perfect correctness. Accordingly, we instantiate our template using our fully succinct, but not rate-1, MKFHE construction.

2.3 Threshold TDH

A classical trapdoor hash (TDH) [DGI+19] for predicates $f : \{0,1\}^\ell \to \{0,1\}$ consists of the following algorithms

$$\mathsf{Setup}(1^\lambda, 1^\ell) \to \mathsf{hk}, \qquad \mathsf{Gen}(\mathsf{hk}, f) \to (\mathsf{ek}, \mathsf{td}), \qquad \mathsf{Enc}(\mathsf{ek}, x; r) \to e,$$
$$\mathsf{Hash}(\mathsf{hk}, x; r) \to \mathsf{d}, \qquad \mathsf{Decode}(\mathsf{td}, \mathsf{d}) \to (e_0, e_1).$$

Informally speaking correctness requires $e = e_{f(x)}$ and $e \neq e_{1-f(x)}$, function privacy (resp. input privacy) requires that ek (resp. d) hides f (resp. x), and compactness requires that $|\mathsf{d}|$ is short. A key point in TDHs is that Hash and Enc use the same randomness r. In this work, we consider TDHs as an application of fully-succinct MKFHE. However, given the asynchronous nature of MKFHE, it is not clear how to construct TDHs from MKFHEs where hashing and encoding are separate algorithms. Therefore, we consider a modification of TDHs where these two algorithms are merged together, i.e., there is a single hashing and encoding procedure as $(\mathsf{d}, e) \leftarrow \mathsf{Hash\&Enc}(\mathsf{hk}, \mathsf{ek}, x)$.[8] While this choice stems from technical reasons, we notice that it is already enough for many applications such as rate-1 oblivious transfer. More specifically, since hashing and encoding share the same randomness, in a lot of applications, they are coupled together and used by the same entity.

Threshold TDH. We lift TDH to an N-out-of-N threshold setting. Setup and generation are the same, while the hashing-and-encoding and decoding algorithms are thresholdized as follows:

$$(\mathsf{d}, e) \leftarrow \mathsf{Hash\&Enc}\big(\mathsf{hk}, (\mathsf{ek}_i)_{i\in[N]}, x\big), \qquad (e_0, e_1) \leftarrow \mathsf{Decode}\big((\mathsf{td}_i)_{i\in[N]}, \mathsf{d}\big).$$

Intuitively, the trapdoor is distributed among the parties, and only the joint decoder (with all (td_i)) can evaluate (e_0, e_1) from the digest d, and recover $f(x)$ using the hint e. More specifically, the threshold input hiding requires that as long as at least one of the trapdoors is hidden from the adversary, then the input x (and so $f(x)$) is also hidden. Our construction of TTDH from MKFHE is as follows:

- the generation algorithm samples an MKFHE key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, encrypts the function f to ct_i, and outputs $(\mathsf{ek}_i = (\mathsf{pk}_i, \mathsf{ct}_i), \mathsf{td}_i = \mathsf{sk}_i)$,
- the hashing and encoding algorithm on input $(\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}$ samples random e and e' and homomorphically evaluates a circuit

$$C_{x,e,e'}(f_1, \ldots, f_N) = \begin{cases} \bot, & \text{if } f_i \neq f_j \text{ for some } i, j \in [N] \\ (e, e'), & \text{if } f_1(x) = 0 \\ (e', e), & \text{if } f_1(x) = 1 \end{cases}$$

  to get a ciphertext $\widetilde{\mathsf{ct}}$, and outputs $(\mathsf{d} = \widetilde{\mathsf{ct}}, e)$,
- the decoding algorithm given $(\mathsf{td}_i)_{i\in[N]} = (\mathsf{sk}_i)_{i\in[N]}$ and $\mathsf{d} = \widetilde{\mathsf{ct}}$, performs the MKFHE decryption to get (e_0, e_1).

[8] We leave the problem of threshold TDHs with separate hashing and encoding algorithms as an interesting open problem.

Correctness and function hiding of the above construction follow directly from the correctness and semantic security of MKFHE. The threshold input hiding follows from the semantic security of MKFHE at any uncompromised index j together with circuit privacy of MKFHE. Finally compactness follows from full succinctness of our MKFHE and since there is no constraint on the length of (e, e') in our scheme, we achieve rate-1 (where $|e| = 1$).

2.4 Related Works

Boyle et al. [BJSS25] introduced *Simultaneous-Message and Succinct Secure Computation (SMS)* protocols for two-party succinct computation in a simultaneous-message model, in a typically asymmetric regime where one party holds a *large* input and the other holds a *small* input. Informally, each party $\sigma \in \{A, B\}$ publishes a *succinct public encoding* pe_σ of its input (while keeping local state st_σ private), and then locally derives an additive share of the function value using st_σ and the other party's encoding $\mathsf{pe}_{\bar{\sigma}}$. They give lattice-based instantiations of SMS by combining lattice-encoding techniques (e.g., [BGG+14, GVW15]) with vector oblivious linear evaluation (VOLE)-type ingredients (e.g., [ARS24]), leveraging ideas from the laconic function evaluation literature. One important application of SMS is a template for constructing *rate-1 FHE*. Although this is presented in the single-key setting, it motivates asking whether analogous SMS-style succinctness ideas can be brought to the multi-key setting.

In a concurrent line of work, Abram, Malavolta, and Roy [AMR25] construct *succinct oblivious tensor evaluation (OTE)* and show that it yields significantly improved succinctness for VOLE-type correlations as a special case, via linear post-processing. Concretely, OTE can serve as the succinctness primitive inside SMS-style templates whenever the target computation admits a structured linear view.

Multi-party SMS. A natural route to importing SMS-style ideas into MKFHE is to formulate an appropriate *multi-party* analogue of SMS. A particularly relevant asymmetric regime is when many parties hold *small* private inputs $(y_i)_i$, while a distinguished evaluator holds a *large* input X; the goal is that the parties can reconstruct a succinct secret sharing of $f(X, (y_i)_i)$. Even at a high level, there are several inequivalent ways to formalize such a notion, depending on requirements such as privacy and succinctness for each party, whether the participating subset is fixed or adaptively chosen, and whether users' contributions must be reusable and posted *asynchronously* long before X is known.

One useful formalization (motivated by FHE ideas and MKFHE notation) considers the following algorithms $\mathsf{ClientEncode}(\mathsf{crs}, y_i, i) \to (\hat{y}_i, \mathsf{sk}_i)$, $\mathsf{ServerEncode}(\mathsf{crs}, (\hat{y}_i)_i, X) \to \hat{X}$, and $\mathsf{Decode}(\mathsf{crs}, (\mathsf{sk}_i)_i, \hat{X}) \to f(X, (y_i)_i)$, where crs is affiliated with f. In this formalization, $\mathsf{ServerEncode}$ is allowed to use the clients' published $\hat{y}_i$ values. If instead one insists on a more simultaneous-message flavor in which $\mathsf{ServerEncode}$ does *not* receive $(\hat{y}_i)_i$, while still requiring strong input privacy for the server-side output $\hat{X}$, then the resulting notion appears very close to obfuscation: for example, letting f be a universal circuit and taking X to describe the circuit to be evaluated would make $\hat{X}$ function as an obfuscation of X.

This illustrates that the exact choice of privacy/succinctness requirements, together with reusability and asynchrony constraints, can substantially affect what a "multi-party SMS" notion can plausibly mean. Our intuition is that we can design a slightly weaker version of multi-party SMS, which is a sufficient and close notion to MKFHE. Consider a relaxed, asynchronous multi-party SMS flavor: users with *small private inputs* publish input-dependent information ahead of time; later, an

evaluator (given the *large* input, which may be public) produces a succinct digest of the joint computation; and the participating users can then reconstruct a succinct secret sharing of the function output using their private inputs. While we do not abstract this framework out as part of our paper, our fully-succinct MKFHE result can be viewed as an *implicit* realization of such a relaxed notion which can later be used in other applications of multi-party SMS. We leave formalizing this notion and exploring further applications as an interesting direction for future work.

3 Preliminaries

Notations. We denote $[N] = \{1, \ldots, N\}$ and $[M, N] = \{M, \ldots, N\}$. In this work we consider all the vectors as row vectors by default, and for vectors $\mathbf{v}_i = (v_{i,1}, \ldots, v_{i,n})$ for $i \in [t]$, by $(\mathbf{v}_1, \ldots, \mathbf{v}_t)$ we denote $(v_{1,1}, \ldots, v_{1,n}, \ldots, v_{t,1}, \ldots, v_{t,n})$. For a vector $\mathbf{v}$, we let $\|\mathbf{v}\|$ denote its ℓ_2 norm and $\|\mathbf{v}\|_\infty$ denote its infinity norm. Similarly, for matrices $\|\cdot\|$ and $\|\cdot\|_\infty$ denote their ℓ_2 and infinity norms (respectively). We use $\lceil x \rfloor$ to denote the rounding of a real number x to the nearest integer, and for integers $p, q \geq 2$, we define $\lceil \cdot \rfloor_q : \mathbb{Z}_p \to \mathbb{Z}_q$ as $\lceil x \rfloor_q = \lceil (q/p)x \rfloor \mod q$. Throughout this paper, when the modulus q of the computation is clear from the context, we drop the $\mod q$ for simplicity, e.g., for two values $x, y \in \mathbb{Z}_q$, by $x + y$ we mean $x + y \mod q$, and for two values $x, y \in \mathbb{Z}_p$, by $\lceil x \rfloor_q + \lceil y \rfloor_q$ we mean $\lceil x \rfloor_q + \lceil y \rfloor_q \mod q$.

The statistical distance for two distributions X, Y over a finite domain Ω is defined as $\Delta(X, Y) := \frac{1}{2} \sum_{\omega \in \Omega} \big| X(\omega) - Y(\omega) \big|$. If X, Y are distribution ensembles parameterized by the security parameter λ and $\Delta(X, Y) \leq \mathsf{negl}(\lambda)$, we say X and Y are statistically indistinguishable and we denote it by $X \overset{stat}{\approx} Y$. Similarly, by $X \overset{comp}{\approx} Y$ we denote computational indistinguishability of X and Y. We write $x \leftarrow \mathcal{D}$ (resp. $x \leftarrow S$) to denote that x is sampled at random according to distribution $\mathcal{D}$ (resp. uniformly at random from set S).

Lemma 3.1 (Rounding of noisy secret shares [DHRW16])**.** For some even modulus $q \in \mathbb{N}$, and $z \in \{0, 1\}$, let $(z_0, z_1) \in \mathbb{Z}_q^2$ be (jointly) random subject to $z_0 + z_1 = q/2 \cdot z + e \mod q$, where e satisfies $q/4 \geq |e|$. Then, $\Pr_{z_0, z_1}[\lceil z_0 \rfloor_2 + \lceil z_1 \rfloor_2 = z \mod 2] > 1 - 2(|e| + 1)/q$.

Looking ahead, in our construction we use a sub-exponential modulus-to-noise ratio, i.e., let $q/(2|e|) \geq \lambda^{\omega(1)}$, using which in Lemma 3.1 we get that:

$$\Pr_{z_0, z_1}[\lceil z_0 \rfloor_2 + \lceil z_1 \rfloor_2 = z \mod 2] > 1 - 2(|e| + 1)/q \geq 1 - \lambda^{-\omega(1)}.$$

Many prior works (e.g., [BJSS25, BKS19]) have used similar generalizations of the rounding lemma.

The following lemma which says that adding large noise "smudges out" any small values is taken verbatim from [MW16] (see e.g., [AJLA+12b] for proof).

Lemma 3.2 (Smudging Lemma)**.** Let $B_1 = B_1(\lambda)$, and $B_2 = B_2(\lambda)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \leftarrow [-B_2, B_2]$ be chosen uniformly at random. Then the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$ as long as $B_1/B_2 = \mathsf{negl}(\lambda)$.

Definition 3.3 (Pseudorandom function (PRF) [GGM86, Gol01])**.** Let λ be the security parameter and let $\mathcal{F} = \{F_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}_{k \in \{0,1\}^\lambda}$ be an efficiently computable family of functions. We say that $\mathcal{F}$ is a pseudorandom function family if for every PPT distinguisher $\mathcal{D}$,

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{F}}(\mathcal{D}, \lambda) := \left| \Pr\left[k \leftarrow \{0,1\}^\lambda : \ \mathcal{D}^{F_k}(1^\lambda) = 1 \right] - \Pr\left[\ \mathcal{D}^{R}(1^\lambda) = 1 \right] \right| \leq \mathsf{negl}(\lambda)$$

where $R : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ is a random function, and by $\mathcal{D}^H$ we denote $\mathcal{D}$ has oracle access to function H.

Remark 3.4. PRFs allow rerandomization even when the PRF key K is public. Namely, given noisy additive shares z_0, z_1, the shifted shares $z_0 + F_K(0)$ and $z_1 - F_K(0)$ form two jointly almost random noisy additive shares, which can be plugged into lemma Lemma 3.1 to argue:

$$\Pr_K\Big[\ \lceil z_0 + F_K(0)\rfloor_2 + \lceil z_1 - F_K(0)\rfloor_2 = \lceil z_0 + z_1\rfloor_2\Big] \geq 1 - \mathsf{negl}(\lambda).$$

This technique is used in many prior works (e.g., Section 5 of [BJSS25]).

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 a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we let $\Lambda_q^{\perp}(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{A}\cdot\mathbf{x}^\top = \mathbf{0} \mod q\}$. For $\mathbf{u} \in \mathbb{Z}_q^n$, we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the coset $\{\mathbf{x} \in \mathbb{Z}^m \ : \ \mathbf{A}\cdot\mathbf{x}^\top = \mathbf{u}^\top \mod q\}$.

Discrete Gaussians. Let σ be any positive real number. The Gaussian distribution $\mathcal{D}_\sigma$ with parameter σ is defined by the probability distribution function $\rho_\sigma(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/\sigma^2)$. For any set $\mathcal{L} \subset \mathcal{R}^m$, define $\rho_\sigma(\mathcal{L}) = \sum_{\mathbf{x}\in\mathcal{L}} \rho_\sigma(\mathbf{x})$. The discrete Gaussian distribution $\mathcal{D}_{\mathcal{L},\sigma}$ over $\mathcal{L}$ with parameter σ is defined by the probability distribution function $\rho_{\mathcal{L},\sigma}(\mathbf{x}) = \rho_\sigma(\mathbf{x})/\rho_\sigma(\mathcal{L})$ for all $\mathbf{x} \in \mathcal{L}$.

Truncated Discrete Gaussians. The truncated discrete Gaussian distribution over $\mathbb{Z}$ with parameter σ, denoted by $\widetilde{\mathcal{D}}_{m,\mathbb{Z},\sigma}$, is same as the discrete Gaussian distribution $\mathcal{D}_\sigma$ except it outputs 0 whenever the ℓ_∞ norm exceeds $\sqrt{m}\cdot\sigma$. Note that, by definition, $\widetilde{\mathcal{D}}_{m,\mathbb{Z},\sigma}$ is $\sqrt{m}\cdot\sigma$-bounded. Also, note that $\widetilde{\mathcal{D}}_{m,\mathbb{Z},\sigma} \approx_s \mathcal{D}_{m,\mathbb{Z},\sigma}$.

Learning with Errors (LWE). The Learning with Errors (LWE) problem was introduced by Regev [Reg09]. The LWE problem has four parameters: the dimension of the lattice n, the number of samples m, the modulus q and the error distribution $\chi = \chi(n)$.

Assumption 3.5 (Learning with Errors)**.** Let n, m and q be positive integers and χ a noise distribution over $\mathbb{Z}_q$. The Learning with Errors assumption (n, m, q, χ)-LWE, parameterized by n, m, q, χ, states that the following distributions are computationally indistinguishable:

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

Under a quantum reduction, Regev [Reg09] showed that for certain noise distributions, LWE is as hard as worst case lattice problems such as the decisional approximate shortest vector problem (GapSVP) and approximate shortest independent vectors problem (SIVP). Later works [Pei09, BLP+13] showed classical reductions from LWE to GapSVP_γ.

These works show that for B_χ-bounded discretized Gaussian error distributions χ, solving (n, m, q, χ)-LWE is as hard as approximating GapSVP_γ and SIVP_γ to a factor of $\tilde{O}(n \cdot q/B_\chi)$. Given the current state of art in lattice algorithms, GapSVP_γ and SIVP_γ are believed to be hard for $\gamma = \tilde{O}(2^{n^\epsilon})$ (for fixed $\epsilon \in (0, 1/2)$), and therefore (n, m, q, χ)-LWE is believed to be hard for B_χ-bounded discretized Gaussian error distributions χ with $B_\chi = 2^{-n^\epsilon} \cdot q \cdot \mathsf{poly}(n)$.

3.2 Multi-Key Fully Homomorphic Encryption

Next we recall the definition of (leveled) multi-key fully-homomorphic encryption (MKFHE) with a single-hop of evaluation[9].

Syntax. A multi-key (leveled) fully-homomorphic encryption scheme for message space $\mathcal{M} = \{0,1\}^*$ consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, 1^d) \to \mathsf{pp}$**.** The probabilistic setup algorithm takes as input a security parameter 1^λ, and a maximum depth d in unary, and outputs the public parameters pp. We will consider public parameters pp as an implicit input to all algorithms.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$**.** The probabilistic key generation algorithm takes as input the public parameters pp and outputs the public and secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, \mu) \to \mathsf{ct}$**.** The probabilistic encryption algorithm takes as input the public key pk, and a message $\mu \in \mathcal{M}$, and outputs a ciphertext ct.

$\mathsf{Eval}(C, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}) \to \widetilde{\mathsf{ct}}$**.** The evaluation algorithm takes as input the description of the circuit C of depth at most d, and a set of public keys and ciphertexts $(\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}$, and outputs the evaluated ciphertext $\widetilde{\mathsf{ct}}$.

$\mathsf{Dec}((\mathsf{sk}_i)_{i\in[N]}, \widetilde{\mathsf{ct}}) \to \tilde{\mu}$**.** The decryption algorithm takes as input a set of secret keys $(\mathsf{sk}_i)_{i\in[N]}$, and an evaluated ciphertext $\widetilde{\mathsf{ct}}$, and outputs a message $\tilde{\mu}$.

Definition 3.6 (MKFHE)**.** A multi-key fully homomorphic encryption $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ for $\mu \in \mathcal{M}$ is required to satisfy the following properties:

Correctness. For any polynomial functions $N = N(\lambda)$ and $d = d(\lambda)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for any $\lambda \in \mathbb{N}$, any $\mu_i \in \mathcal{M}$ for $i \in [N]$, any N-ary circuit C of maximum depth d, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, all $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for $i \in [N]$, all $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mu_i)$ for $i \in [N]$, and $\widetilde{\mathsf{ct}} = \mathsf{Eval}(C, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]})$:

$$\Pr[\mathsf{Dec}((\mathsf{sk}_i)_{i\in[N]}, \widetilde{\mathsf{ct}}) = C((\mu_i)_{i\in[N]})] \geq 1 - \mathsf{negl}(\lambda).$$

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

$$\Pr\left[\begin{array}{l} b \leftarrow \mathcal{A}(\mathsf{ct}_b) \end{array} : \begin{array}{l} 1^d \leftarrow \mathcal{A}, b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^d), \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), \\ (\mu_0, \mu_1) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}), \\ |\mu_0| = |\mu_1|, \\ \mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, \mu_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

[9] It is known that any single-hop MKFHE can be generically bootstrapped to support multi-hop evaluation via Gentry's bootstrapping [Gen09]. We will discuss this in more detail in Section 4.3.

Compactness. For any $\lambda \in \mathbb{N}$, any polynomial functions $N = N(\lambda)$ and $d = d(\lambda)$, any $\mu_i \in \mathcal{M}$ for $i \in [N]$, and any N-ary circuit C of maximum depth d, there exists a fixed polynomial $\mathsf{poly}(\cdot,\cdot,\cdot,\cdot)$ such that for any $\widetilde{\mathsf{ct}}$ in the support of the evaluation (as defined in the correctness experiment), we have:

$$|\widetilde{\mathsf{ct}}| = \mathsf{poly}(\lambda, d, N, |C((\mu_i)_{i\in[N]})|).$$

Definition 3.7 (Fully Succinct MKFHE)**.** An MKFHE scheme is said to be fully succinct if the evaluated ciphertext size $|\widetilde{\mathsf{ct}}|$ in the compactness property is bounded by $\mathsf{poly}(\lambda, d, \log N, |C(\mu_1, \ldots, \mu_N)|)$.

Definition 3.8 (Rate-1 MKFHE)**.** An MKFHE scheme is said to be rate-1 if the evaluated ciphertext size $|\widetilde{\mathsf{ct}}|$ in the compactness property is bounded by $|C(\mu_1, \ldots, \mu_N)| + \mathsf{poly}(\lambda, d, N)$.

Definition 3.9 (Circuit Privacy.)**.** An MKFHE scheme is said to be circuit private if for any polynomial $N = N(\lambda)$ and any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\Pr\left[\, b = b' \quad : \quad \begin{array}{l} 1^d \leftarrow \mathcal{A}, b \leftarrow \{0,1\}, \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^d), \\ \forall i \in [N], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), \\ ((\mu_i)_{i\in[N]}, C_0, C_1) \leftarrow \mathcal{A}(\mathsf{pp}, (\mathsf{pk}_i, \mathsf{sk}_i)_{i\in[N]}), \\ C_0((\mu_i)_{i\in[N]}) = C_1((\mu_i)_{i\in[N]}), \\ \forall i \in [N], \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mu_i), \\ \widetilde{\mathsf{ct}}_b \leftarrow \mathsf{Eval}(\mathsf{pp}, C_b, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}), \\ b' \leftarrow \mathcal{A}(\widetilde{\mathsf{ct}}_b, (\mathsf{ct}_i)_{i\in[N]}) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Additional Properties. In this work we rely on multi-key FHE schemes that satisfy some additional properties as defined below:

Definition 3.10 (MKFHE With Split and Near-Linear Decryption)**.** For a multi-key fully-homomorphic encryption scheme MKFHE, we say MKFHE has split and near-linear decryption if it additionally satisfies the following properties:

1. For any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, the public parameter pp contains an even modulus q.
2. For any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, the secret key sk is a vector in $\mathbb{Z}_q^n$ for some $n = \mathsf{poly}(\lambda, d)$.
3. The encryption algorithm $\mathsf{Enc}(\mathsf{pk}, \mu)$ outputs a sequence of matrices $\mathsf{ct} \in (\mathbb{Z}_q^{n\times m})^{(m^2+1)}$ for some $m = n \log q + \omega(\log \lambda)$.
4. The evaluation algorithm $\mathsf{Eval}(C, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]})$ for a binary circuit C outputs a vector $\widetilde{\mathsf{ct}} \in \mathbb{Z}_q^{Nn}$ such that:
$$\langle \widetilde{\mathsf{ct}}, \overline{\mathsf{sk}} \rangle = q/2 \cdot \mathsf{Dec}(\overline{\mathsf{sk}}, \widetilde{\mathsf{ct}}) + e \mod q$$
where $\overline{\mathsf{sk}} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$, $e \in \mathbb{Z}_q$ and $|e| \leq (mN+1)^d \cdot B_\chi \cdot \mathsf{poly}(\lambda)$, where B_χ is the bound on the Gaussian error distribution χ.
5. The evaluation algorithm $\mathsf{Eval}(C, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]})$ for a circuit C with ℓ bits of output, outputs a vector $\widetilde{\mathsf{ct}} \in \mathbb{Z}_q^{\ell n N}$ where $\widetilde{\mathsf{ct}} = (\widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_\ell)$ and $\widetilde{\mathsf{ct}}_j \in \mathbb{Z}_q^{Nn}$ for $j \in [\ell]$ such that $\widetilde{\mathsf{ct}}_j = \mathsf{Eval}(C_j, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]})$, where C_j is a binary circuit that computes the j^{th} output bit of C. Namely, it holds that $\widetilde{\mathsf{ct}} = (\mathsf{Eval}(C_1, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}), \ldots, \mathsf{Eval}(C_\ell, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}))$.

Theorem 3.11 (MKFHE With Split and Near-Linear Decryption [MW16])**.** Assuming the LWE assumption with modulus-to-noise ratio $q/B_\chi = 2^{\mathsf{poly}(\log\lambda, d, \log N)}$, there exists a multi-key fully homomorphic encryption scheme with split and near-linear decryption.

3.2.1 MKFHE With Single-Round Distributed Decryption

Next, we recall the definition of (leveled) multi-key fully homomorphic encryption equipped with single-round distributed decryption, following [MW16].[10]

Syntax. A multi-key (leveled) fully-homomorphic encryption with single-round distributed decryption scheme is a multi-key (leveled) fully-homomorphic encryption scheme consisting of the following additional polynomial time algorithms.

$\mathsf{PartDec}(\widetilde{\mathsf{ct}}, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i) \to p_i$**.** The partial decryption algorithm takes as input an evaluated ciphertext $\widetilde{\mathsf{ct}}$, a set of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$, an index i, and a secret key sk_i, and outputs a partial decryption p_i.

$\mathsf{FinDec}(p_1, \ldots, p_N) \to \mu$**.** The final decryption algorithm takes as input a set of partial decryptions $(p_1, \ldots, p_N)$, and outputs a message μ.

Definition 3.12 (MKFHE With Single-Round Distributed Decryption)**.** A multi-key (leveled) fully-homomorphic encryption with single-round distributed decryption scheme in addition to Definition 3.6 is required to additionally satisfy the following properties:

Correctness of Partial Decryption. For any polynomial functions $N = N(\lambda)$ and $d = d(\lambda)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for any $\lambda \in \mathbb{N}$, any $\mu_i \in \mathcal{M}$ for $i \in [N]$, any N-ary circuit C of maximum depth d, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, all $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for $i \in [N]$, all $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mu_i)$ for $i \in [N]$, $\widetilde{\mathsf{ct}} = \mathsf{Eval}(C, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]})$, and all $p_i \leftarrow \mathsf{PartDec}(\widetilde{\mathsf{ct}}, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i, \mathsf{sk}_i)$ for $i \in [N]$:

$$\Pr[\mathsf{FinDec}(p_1, \ldots, p_N) = C((\mu_i)_{i\in[N]})] \geq 1 - \mathsf{negl}(\lambda).$$

Simulatability of Partial Decryption. There exists a PPT simulator $p_i' \leftarrow \mathcal{S}(\tilde{\mu}, \widetilde{\mathsf{ct}}, C, i, (\mathsf{sk}_j)_{j\in[N]\setminus\{i\}})$ that takes the output value $\tilde{\mu} = C(\mu_1, \ldots, \mu_N)$, the circuit C, an index $i \in [N]$, and all but the i^{th} secret keys $(\mathsf{sk}_j)_{j\in[N]\setminus\{i\}}$, and outputs a simulated partial decryption p_i' such that, for any polynomial functions $N = N(\lambda)$ and $d = d(\lambda)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for any $\lambda \in \mathbb{N}$, any $\mu_i \in \mathcal{M}$ for $i \in [N]$, any N-ary circuit C of maximum depth d, any $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, all $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for $i \in [N]$, all $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mu_i)$ for $i \in [N]$, and $\widetilde{\mathsf{ct}} = \mathsf{Eval}(C, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]})$, it holds that $p_i \overset{stat}{\approx} p_i'$ (i.e., $\Delta(p_i, p_i') \leq \mathsf{negl}(\lambda)$) where $p_i \leftarrow \mathsf{MKFHE.PartDec}(\widetilde{\mathsf{ct}}, (\mathsf{pk}_1, ..., \mathsf{pk}_N), i, \mathsf{sk}_i)$.

Note that the randomness (in the distribution of p_i and p_i') is only over the random coins of the simulator and the partial decryption procedure and all other values are assumed to be fixed (and known).

[10] In [MW16] this property is termed *threshold decryption*. To avoid overloading the term "threshold," we use *single-round distributed decryption* throughout this paper.

3.3 Succinct Non-Interactive Oblivious Tensor Evaluation

Here we recall the definition of a succinct non-interactive oblivious tensor evaluation from [AMR25].

Syntax. A succinct non-interactive oblivious tensor evaluation (OTE) scheme consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$**.** The probabilistic setup algorithm takes as input a security parameter 1^λ, and outputs a CRS crs.

$\mathsf{Hash}(\mathsf{crs}, \mathbf{y}) \to (\mathsf{d}, \mathsf{st}_\mathbf{y})$**.** The potentially probabilistic hashing algorithm takes as input the CRS crs, and a vector $\mathbf{y}$, and outputs a digest d and a hasher's private state $\mathsf{st}_\mathbf{y}$. In this work we rely on a deterministic hashing algorithm.

$\mathsf{Encode}(\mathsf{crs}, \mathbf{x}) \to (\hat{\mathbf{x}}, \mathsf{st}_\mathbf{x})$**.** The probabilistic encoding algorithm takes as input the CRS crs, and a vector $\mathbf{x}$, and outputs an encoding $\hat{\mathbf{x}}$ and an encoder's private state $\mathsf{st}_\mathbf{x}$.

$\mathsf{HashEval}(\mathsf{crs}, \hat{\mathbf{x}}, \mathsf{st}_\mathbf{y}) \to \mathbf{v}$**.** The hasher's evaluation algorithm takes as input the CRS crs, an encoding $\hat{\mathbf{x}}$, and the hasher's private state $\mathsf{st}_\mathbf{y}$ and outputs a vector $\mathbf{v}$.

$\mathsf{EncodeEval}(\mathsf{crs}, \mathsf{d}, \mathsf{st}_\mathbf{x}) \to \mathbf{w}$**.** The encoder's evaluation algorithm takes as input the CRS crs, a digest d, and the encoder's private state $\mathsf{st}_\mathbf{x}$ and outputs a vector $\mathbf{w}$.

Definition 3.13 (Succinct NI-OTE)**.** A succinct non-interactive oblivious tensor evaluation scheme $\mathsf{OTE} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{HashEval}, \mathsf{EncodeEval})$ for any polynomial is required to satisfy the following properties:

Correctness. There exists a negligible function $\mathsf{negl}(\lambda)$ such that for any $\lambda \in \mathbb{N}$, any $\ell_\mathbf{x}, \ell_\mathbf{y} = \mathsf{poly}(\lambda)$, any $\mathbf{y} \in \mathbb{Z}_q^{\ell_\mathbf{y}}$, and any $\mathbf{x} \in \mathbb{Z}_q^{\ell_\mathbf{x}}$, it holds that:

$$\Pr\left[\begin{array}{l|ll} & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ & (\mathsf{d}, \mathsf{st}_\mathbf{y}) \leftarrow \mathsf{Hash}(\mathsf{crs}, \mathbf{y}) \\ \mathbf{v} + \mathbf{w} = \mathbf{y} \otimes \mathbf{x} & : & (\hat{\mathbf{x}}, \mathsf{st}_\mathbf{x}) \leftarrow \mathsf{Encode}(\mathsf{crs}, \mathbf{x}) \\ & \mathbf{v} \leftarrow \mathsf{HashEval}(\mathsf{crs}, \hat{\mathbf{x}}, \mathsf{st}_\mathbf{y}) \\ & \mathbf{w} \leftarrow \mathsf{EncodeEval}(\mathsf{crs}, \mathsf{d}, \mathsf{st}_\mathbf{x}) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

where $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_q^{\ell_\mathbf{y} \cdot \ell_\mathbf{x}}$.

Encoder Privacy. There is a stateful PPT simulator $\mathcal{S}$ such that for any stateful PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds that:

$$\Pr\left[\begin{array}{l|ll} & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), b \leftarrow \{0,1\} \\ \mathcal{A}(\hat{\mathbf{x}}_b) = b & : & \mathbf{x} \leftarrow \mathcal{A}(\mathsf{crs}) \\ & (\hat{\mathbf{x}}_0, \mathsf{st}_\mathbf{x}) \leftarrow \mathsf{Encode}(\mathsf{crs}, \mathbf{x}) \\ & \hat{\mathbf{x}}_1 \leftarrow \mathcal{S}(\mathsf{crs}, \ell_\mathbf{x}) \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda).$$

Succinctness. There exists a fixed polynomial $\mathsf{poly}(\cdot,\cdot,\cdot,\cdot)$ s.t. for all $\lambda \in \mathbb{N}$, all crs in the support of $\mathsf{Setup}(1^\lambda)$, all $\ell_{\mathbf{x}}, \ell_{\mathbf{y}} = \mathsf{poly}(\lambda)$, all inputs $\mathbf{x} \in \mathbb{Z}_q^{\ell_{\mathbf{x}}}$ and $\mathbf{y} \in \mathbb{Z}_q^{\ell_{\mathbf{y}}}$, all encoding $\hat{\mathbf{x}}$ and private state $\mathsf{st}_{\mathbf{x}}$ output by $\mathsf{Encode}(\mathsf{crs}, \mathbf{x})$ and all digest d output by $\mathsf{Hash}(\mathsf{crs}, \mathbf{y})$ it holds that $|\mathsf{crs}|, |\hat{\mathbf{x}}|, |\mathsf{st}_{\mathbf{x}}|, |\mathsf{d}| \leq \mathsf{poly}(\lambda, \log q, \ell_{\mathbf{x}}, \log \ell_{\mathbf{y}})$.[11]

Definition 3.14 (Perfectly Correct Succinct NI-OTE)**.** A succinct non-interactive oblivious tensor evaluation scheme $\mathsf{OTE} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{HashEval}, \mathsf{EncodeEval})$ is said to be perfectly correct, if $p > \mathsf{poly}(\lambda, q, \ell_{\mathbf{y}})$ is a multiple of q, and it satisfies encoder privacy and succinctness as in Definition 3.13, and instead of the correctness in Definition 3.13, it satisfies the following property:
Perfect Correctness. For any $\lambda \in \mathbb{N}$, any $\ell_{\mathbf{x}}, \ell_{\mathbf{y}} = \mathsf{poly}(\lambda)$, any $\mathbf{y} \in \mathbb{Z}_q^{\ell_{\mathbf{y}}}$, and any $\mathbf{x} \in \mathbb{Z}_q^{\ell_{\mathbf{x}}}$, it holds that:

$$\Pr\left[\begin{array}{l|ll} & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ & (\mathsf{d}, \mathsf{st}_{\mathbf{y}}) \leftarrow \mathsf{Hash}(\mathsf{crs}, \mathbf{y}) \\ \lceil \mathbf{v} + \mathbf{w} \rfloor_q = \mathbf{y} \otimes \mathbf{x} & : & (\hat{\mathbf{x}}, \mathsf{st}_{\mathbf{x}}) \leftarrow \mathsf{Encode}(\mathsf{crs}, \mathbf{x}) \\ & \mathbf{v} \leftarrow \mathsf{HashEval}(\mathsf{crs}, \hat{\mathbf{x}}, \mathsf{st}_{\mathbf{y}}) \\ & \mathbf{w} \leftarrow \mathsf{EncodeEval}(\mathsf{crs}, \mathsf{d}, \mathsf{st}_{\mathbf{x}}) \end{array}\right] = 1$$

where $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p^{\ell_{\mathbf{x}} \cdot \ell_{\mathbf{y}}}$.

Remark 3.15 (From α-Correctness to Correctness and Perfect Correctness.)**.** [AMR25] defines OTE with a weaker notion of α-correctness which allows for correctness error up to some $\alpha \cdot \|\mathbf{y}\|_\infty$ in the equality of $\mathbf{v} + \mathbf{w} = \mathbf{y} \otimes \mathbf{x}$ and up to some negligible function in the probability of this equality holding. Namely, the α-correctness is defined as

$$\Pr[\|\mathsf{HashEval}(\mathsf{crs}, \hat{\mathbf{x}}, \mathsf{st}_{\mathbf{y}}) + \mathsf{EncodeEval}(\mathsf{crs}, \mathsf{d}, \mathsf{st}_{\mathbf{x}}) - \mathbf{y} \otimes \mathbf{x}\|_\infty > \alpha \cdot \|\mathbf{y}\|_\infty] \leq \mathsf{negl}(\lambda).$$

Additionally, the constructions on Sections 3.2 and 3.3 of [AMR25] achieve the weaker notion of α-correctness, however, as they discuss it in the α-correctness definition (Section 3, Definition 3.2), one can bring the error α down to 0 by encoding $p/q \cdot \mathbf{x}$ for some modulus p s.t. $p/q \in \lambda^{\omega(1)}$, and using standard rounding tricks (i.e., rounding $\mathsf{HashEval}(\mathsf{crs}, \hat{\mathbf{x}}, \mathsf{st}_{\mathbf{y}})$ and $\mathsf{EncodeEval}(\mathsf{crs}, \mathsf{d}, \mathsf{st}_{\mathbf{x}})$ to the nearest multiple of p/q to compute $\mathbf{v}$ and $\mathbf{w}$). Moreover, working in the larger modulus p does not affect the succinctness except by a $\mathsf{poly}(\lambda)$ factor as $\log p = O(\lambda, \log q)$.

We first notice that in their construction, the error probability for achieving α-correctness is 0. Furthermore, note that if instead of computing $\mathbf{v}$ and $\mathbf{w}$ by rounding $\mathsf{HashEval}(\mathsf{crs}, \hat{\mathbf{x}}, \mathsf{st}_{\mathbf{y}})$ and $\mathsf{EncodeEval}(\mathsf{crs}, \mathsf{d}, \mathsf{st}_{\mathbf{x}})$, we let $\mathbf{v} = \mathsf{HashEval}(\mathsf{crs}, \hat{\mathbf{x}}, \mathsf{st}_{\mathbf{y}})$ and $\mathbf{w} = \mathsf{EncodeEval}(\mathsf{crs}, \mathsf{d}, \mathsf{st}_{\mathbf{x}})$, then by the α-correctness of their construction (with error probability 0) it holds that:

$$\lceil \mathbf{v} + \mathbf{w} \rfloor_q = \lceil p/q \cdot \mathbf{y} \otimes \mathbf{x} \pm \alpha \cdot \|\mathbf{y}\|_\infty \rfloor_q = \mathbf{y} \otimes \mathbf{x} \pm \lceil \alpha \cdot \|\mathbf{y}\|_\infty \rfloor_q = \mathbf{y} \otimes \mathbf{x},$$

where the second equality holds if p is a multiple of q and the last equality holds if $p/(2q) > \alpha \cdot q$ or equivalently $p > 2q^2 \cdot \alpha$. From their construction it holds that $\alpha = \mathsf{poly}(\lambda, \log q, \ell_{\mathbf{y}}) \cdot B'_\chi$ for some B'_χ-bounded Gaussian error that is used for the security of LWE. Thus, by choosing p as a multiple of q s.t. $p > \mathsf{poly}(\lambda, q, \ell_{\mathbf{y}}) \cdot B'_\chi$, for an appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$ we get perfectly correct OTE.

[11] In [AMR25], succinctness is defined as $|\hat{\mathbf{x}}| + |\mathsf{d}| \leq \mathsf{poly}(\lambda, \ell_{\mathbf{x}}, \log \ell_{\mathbf{y}})$ with any $\log q$ dependence left implicit under the convention $\log q = \mathsf{poly}(\lambda)$. In our setting, q may grow with additional parameters, so we make the $\log q$ dependence explicit. We also require similar polynomial size bound for the CRS and the auxiliary state $\mathsf{st}_{\mathbf{x}}$, which is already satisfied by [AMR25] and is therefore included in our definition.

Theorem 3.16 (Perfectly Correct Succinct NI-OTE (Adapted From [AMR25], See Remark 3.15))**.** Assuming LWE with modulus-to-noise ratio $p/B'_\chi > 2^{\mathsf{poly}(\log\lambda,\log q,\log \ell_{\mathbf{y}})}$ there exists a perfectly correct succinct NI-OTE (with deterministic hashing algorithm) where

$$|\mathsf{crs}|, |\mathsf{st}_{\mathbf{x}}|, |\mathsf{d}| \leq \mathsf{poly}(\lambda, \log q) \quad \text{and} \quad |\hat{\mathbf{x}}| \leq \mathsf{poly}(\lambda, \log q, \ell_{\mathbf{x}}, \log \ell_{\mathbf{y}}).$$

3.4 Correlation Intractable Hash Functions

Next we recall the definitions of searchable relations and somewhere statistically correlation intractable hash functions from [PS19]. As in [PS19], in this subsection we consider non-uniform polynomial-size adversaries.

Definition 3.17 (Searchable Relations [PS19])**.** We say that a relation $R \subseteq \mathcal{X} \times \mathcal{Y}$ is searchable in size S if there exists a function $f : \mathcal{X} \to \mathcal{Y}$ that is implementable as a boolean circuit of size S, such that if $(x, y) \in R$ then $y = f(x)$. In other words, $f(x)$ is the unique witness for x, if such a witness exists, and we say that f searches R.

Syntax. A hash function family with fake-key generation $\mathsf{CIH} = (\mathsf{Gen}, \mathsf{StatGen}, \mathsf{Hash})$, for an output length L and a size bound S, consists of the following polynomial time algorithms, where we consider 1^L and 1^S as implicit inputs to Gen and $\mathsf{StatGen}$.

$\mathsf{Gen}(1^\lambda) \to \mathsf{hk}$**.** The probabilistic key generation algorithm takes as input a security parameter 1^λ, and outputs a hash key hk.

$\mathsf{StatGen}(1^\lambda, \mathsf{aux}) \to \mathsf{hk}$**.** The probabilistic fake-key generation algorithm takes as input a security parameter 1^λ, and an auxiliary input aux, and outputs a hash key hk.

$\mathsf{Hash}(\mathsf{hk}, x) \to y$**.** The deterministic hashing algorithm takes as input a hash key hk, and an input x, and outputs a hash value y.

Definition 3.18 (Somewhere Statistical Correlation Intractability [PS19])**.** Let $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ be a relation class. A hash function family $\mathsf{CIH} = (\mathsf{Gen}, \mathsf{StatGen}, \mathsf{Hash})$ is somewhere statistically correlation intractable for $\mathcal{R}$ if there exist a negligible function $\nu(\cdot)$ and a class of auxiliary inputs $\mathsf{AUX} = \{\mathsf{AUX}_\lambda\}_{\lambda\in\mathbb{N}}$ such that the following properties hold:

Mode Indistinguishability. For every sequence $\{\mathsf{aux}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathsf{aux}_\lambda \in \mathsf{AUX}_\lambda$, it holds that

$$\{\mathsf{StatGen}(1^\lambda, \mathsf{aux}_\lambda)\}_{\lambda\in\mathbb{N}} \overset{comp}{\approx} \{\mathsf{Gen}(1^\lambda)\}_{\lambda\in\mathbb{N}}.$$

Statistical Correlation Intractability. For every $\lambda \in \mathbb{N}$ and every relation $R \in \mathcal{R}_\lambda$, there exists $\mathsf{aux}_R \in \mathsf{AUX}_\lambda$ such that

$$\Pr_{\mathsf{hk}\leftarrow\mathsf{StatGen}(1^\lambda,\mathsf{aux}_R)}[\exists x \text{ s.t. } (x, \mathsf{Hash}(\mathsf{hk}, x)) \in R] \leq \nu(\lambda).$$

We call any such aux_R an intractability guarantee for R.

Remark 3.19. Since we consider non-uniform adversaries, mode indistinguishability also holds when the adversary first chooses $\mathsf{aux} \in \mathsf{AUX}_\lambda$ (and the implicit parameters), and then receives a hash key hk that is sampled either by $\mathsf{StatGen}(1^\lambda, \mathsf{aux})$ or by $\mathsf{Gen}(1^\lambda)$. This follows by fixing the random coins of the first step of the adversary that maximize its distinguishing advantage.

Definition 3.20 (Somewhere Perfect Correlation Intractability)**.** A hash function family $\mathsf{CIH} = (\mathsf{Gen}, \mathsf{StatGen}, \mathsf{Hash})$ is somewhere perfectly correlation intractable for $\mathcal{R}$ if it satisfies Definition 3.18 with $\nu = 0$. Namely, for every $\lambda \in \mathbb{N}$, every relation $R \in \mathcal{R}_\lambda$ with intractability guarantee aux_R, every hash key hk in the support of $\mathsf{StatGen}(1^\lambda, \mathsf{aux}_R)$, and every input x, it holds that $(x, \mathsf{Hash}(\mathsf{hk}, x)) \notin R$.

Remark 3.21. In this work, we use CIH only for the relations $R_f = \{(x, f(x))\}$, where f is a circuit of size at most $S = S(\lambda)$ with output length $L = L(\lambda)$, AUX_λ consists of the descriptions of these circuits, and $\langle f \rangle$ is the intractability guarantee for R_f (as in [PS19], see Remark 3.22). Namely, we say that CIH is a *somewhere perfectly correlation intractable hash family for circuits of size S with output length L* if it satisfies Definition 3.20 for these relations, i.e., for every such circuit f, every hash key hk in the support of $\mathsf{StatGen}(1^\lambda, \langle f \rangle)$, and every input x, it holds that $\mathsf{Hash}(\mathsf{hk}, x) \neq f(x)$.

Remark 3.22 (From Somewhere Statistical to Somewhere Perfect Correlation Intractability.)**.** [PS19] constructs, from LWE with a polynomial modulus-to-noise ratio, a somewhere statistically correlation intractable hash family for the relations $R_f = \{(x, f(x))\}$, where f is any circuit of output length L and size S, and the intractability guarantee for R_f is the circuit f itself. Their construction first obtains such a hash family for the decryption circuit of a (leveled) FHE scheme, which has logarithmic depth and hence can be computed by a polynomial-length branching program, and then bootstraps it to all circuits of size S by hashing a homomorphic evaluation on an FHE encryption of the circuit. Moreover, their analysis already gives the perfect version of Definition 3.20. Namely, when the LWE error distributions (both in the base hash and in the FHE scheme used for bootstrapping) are bounded (e.g., truncated discrete Gaussians), their proof shows that for every hash key in the support of $\mathsf{StatGen}$ of the base hash there is no input that violates the relation, and their bootstrapping preserves this property since it relies on an FHE scheme whose correctness holds with probability 1.

Theorem 3.23 (Somewhere Perfect CIH (Adapted From [PS19], See Remark 3.22))**.** Assuming LWE with a polynomial modulus-to-noise ratio (against non-uniform polynomial-size adversaries), for any polynomials $L = L(\lambda)$ and $S = S(\lambda)$ with $L(\lambda) \geq \lambda^c$ for a constant $c > 0$, there exists a somewhere perfectly correlation intractable hash family for circuits of size S with output length L.

4 Fully-Succinct Multi-Key Fully Homomorphic Encryption

In this section, we show how to achieve fully-succinct ciphertext size in a multi-key homomorphic encryption scheme. Moreover, our construction supports single-round distributed decryption without simulatability. In the next section, we show how to additionally capture simulatability for partial decryption values.

Before describing our construction, we establish a relation between tensor products of evaluated ciphertexts and secret keys with MKFHE decryption procedure. Correctness of our construction heavily relies on this relation.

Recall from Definition 3.10 that in a MKFHE with split and near-linear decryption, for a circuit C with ℓ bits of output and an evaluated ciphertext $\widetilde{\mathsf{ct}} = (\widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_\ell)$ where $\widetilde{\mathsf{ct}}_j$ is a vector in $\mathbb{Z}_q^{nN}$ for any $j \in [\ell]$, it holds that $\langle \widetilde{\mathsf{ct}}_j, \overline{\mathsf{sk}} \rangle = q/2 \cdot \mathsf{MKFHE.Dec}(\overline{\mathsf{sk}}, \widetilde{\mathsf{ct}}_j) + e \mod q$ where

$\overline{\mathsf{sk}} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_N) \in \mathbb{Z}_q^{nN}$. Then, we can write the decryption of the j^{th} bit as follows:

$$\begin{aligned} q/2 \cdot \mathsf{MKFHE.Dec}(\overline{\mathsf{sk}}, \widetilde{\mathsf{ct}}_j) + e &= \langle \widetilde{\mathsf{ct}}_j, \overline{\mathsf{sk}} \rangle \\ &= \langle \widetilde{\mathsf{ct}}[(j-1)nN + 1, jnN], \overline{\mathsf{sk}} \rangle \\ &= \sum_{i=1}^{N} \langle \widetilde{\mathsf{ct}}[(j-1)nN + (i-1)n + 1, (j-1)nN + in], \mathsf{sk}_i \rangle \qquad (1) \\ &= \sum_{i=1}^{N} \sum_{k=1}^{n} \widetilde{\mathsf{ct}}[g_{N,n}(j,i,k)] \cdot \mathsf{sk}_i[k], \end{aligned}$$

where we denote $g_{N,n}(j,i,k) \coloneqq (j-1)Nn + (i-1)n + k$ to be the index in $\widetilde{\mathsf{ct}}$ that is aligned with $\mathsf{sk}_i[k]$, in the inner product computation for decryption of the j^{th} output ciphertext.

Moreover, by the definition of tensor product, for vectors $\mathbf{y} \in \mathbb{Z}_q^{n\ell N}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{Z}_q^n$, and for values $\beta \in [\ell nN]$, $i \in [N]$ and $k \in [n]$ it holds that:

$$\mathbf{y}[\beta] \cdot \mathbf{x}_i[k] = (\mathbf{y} \otimes \mathbf{x}_i)\left[(\beta - 1)\, n + k\right],$$

thus, for the above vectors and values, and any $\beta_1, \ldots, \beta_n \in [n\ell N]$ it holds that:

$$\sum_{k=1}^{n} \mathbf{y}[\beta_k] \cdot \mathbf{x}_i[k] = \sum_{k=1}^{n} (\mathbf{y} \otimes \mathbf{x}_i)\left[(\beta_k - 1)\, n + k\right],$$

therefore, for the above vectors and values, and any $\beta_{1,1}, \ldots, \beta_{N,n} \in [n\ell N]$ it holds that:

$$\sum_{i=1}^{N} \sum_{k=1}^{n} \mathbf{y}[\beta_{i,k}] \cdot \mathbf{x}_i[k] = \sum_{i=1}^{N} \sum_{k=1}^{n} (\mathbf{y} \otimes \mathbf{x}_i)\left[(\beta_{i,k} - 1)\, n + k\right],$$

hence, for any fixed $j \in [\ell]$, by letting $\mathbf{y} = \widetilde{\mathsf{ct}}$, $\mathbf{x}_i = \mathsf{sk}_i$, $\beta_{i,k} = g_{N,n}(j,i,k)$ and $h_{N,n}(j,i,k) = (g_{N,n}(j,i,k) - 1)n + k$, we get the following:

$$\sum_{i=1}^{N} \sum_{k=1}^{n} \widetilde{\mathsf{ct}}[g_{N,n}(j,i,k)] \cdot \mathsf{sk}_i[k] = \sum_{i=1}^{N} \sum_{k=1}^{n} (\widetilde{\mathsf{ct}} \otimes \mathsf{sk}_i)\left[h_{N,n}(j,i,k)\right]. \qquad (2)$$

4.1 Construction

Here we construct a fully-succinct MKFHE for $d = d(\lambda)$ depth bounded computations from a perfectly correct succinct non-interactive oblivious tensor evaluation OTE, and any MKFHE with split and near-linear decryption.

Construction 4.1. Let OTE be a perfectly correct succinct non-interactive oblivious tensor evaluation and MKFHE' be a multi-key fully homomorphic encryption with split and near-linear decryption.

$\mathsf{Setup}(1^\lambda, 1^d) \to \mathsf{PP}$**.** Sample parameters $\mathsf{pp} \leftarrow \mathsf{MKFHE}'.\mathsf{Setup}(1^\lambda, 1^d)$ and $\mathsf{ote.crs} \leftarrow \mathsf{OTE.Setup}(1^\lambda)$, and output $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs})$. We consider PP as an implicit input to all algorithms.

$\mathsf{KeyGen}(\mathsf{PP}) \to (\mathsf{PK}, \mathsf{SK})$**.** First sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{MKFHE}'.\mathsf{KeyGen}(\mathsf{pp})$, then compute $(\mathsf{ote.}\hat{\mathsf{sk}}, \mathsf{ote.st}_{\mathsf{sk}}) \leftarrow \mathsf{OTE.Encode}(\mathsf{ote.crs}, \mathsf{sk})$. Output $\mathsf{PK} = (\mathsf{pk}, \mathsf{ote.}\hat{\mathsf{sk}})$ and $\mathsf{SK} = (\mathsf{sk}, \mathsf{ote.st}_{\mathsf{sk}})$.

$\mathsf{Enc}(\mathsf{PK}, \mu) \to \mathsf{CT}$**.** Compute $\mathsf{ct} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}, \mu)$ and output ct.

$\mathsf{Eval}(C, (\mathsf{PK}_i, \mathsf{CT}_i)_{i\in[N]}) \to \widetilde{\mathsf{CT}}$**.** Do the following:

1. for $i \in [N]$, parse $\mathsf{PK}_i = (\mathsf{pk}_i, \mathsf{ote.}\hat{\mathsf{sk}}_i)$, and $\mathsf{CT}_i = \mathsf{ct}_i$,
2. let C_j be the circuit that computes the j^{th} output bit of C,
3. for $j \in [\ell]$ compute $\widetilde{\mathsf{ct}}_j = \mathsf{MKFHE}'.\mathsf{Eval}\left(C_j, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}\right)$,
4. let $\widetilde{\mathsf{ct}} = (\widetilde{\mathsf{ct}}_1 \| \ldots \| \widetilde{\mathsf{ct}}_\ell)$ and compute $(\mathsf{ote.d}, \mathsf{ote.st}_{\widetilde{\mathsf{ct}}}) \leftarrow \mathsf{OTE.Hash}(\mathsf{ote.crs}, \widetilde{\mathsf{ct}})$,
5. for $i \in [N]$, compute $\mathbf{v}_i \leftarrow \mathsf{OTE.HashEval}(\mathsf{ote.crs}, \mathsf{ote.}\hat{\mathsf{sk}}_i, \mathsf{ote.st}_{\widetilde{\mathsf{ct}}})$,
6. for $j \in [\ell]$, compute $v_j^* = \sum_{i=1}^{N} \sum_{k=1}^{n} \mathbf{v}_i\,[h_{N,n}(j,i,k)]$,

and output $\widetilde{\mathsf{CT}} = ((v_j^*)_{j\in[\ell]}, \mathsf{ote.d})$.

$\mathsf{Dec}((\mathsf{SK}_i)_{i\in[N]}, \widetilde{\mathsf{CT}}) \to \mu$**.** Do the following:

1. parse $\widetilde{\mathsf{CT}} = ((v_j^*)_{j\in[\ell]}, \mathsf{ote.d})$, and for $i \in [N]$, $\mathsf{SK}_i = (\mathsf{sk}_i, \mathsf{ote.st}_{\mathsf{sk},i})$,
2. for $i \in [N]$, compute $\mathbf{w}_i \leftarrow \mathsf{OTE.EncodeEval}(\mathsf{ote.crs}, \mathsf{ote.d}, \mathsf{ote.st}_{\mathsf{sk},i})$,
3. for $j \in [\ell]$, compute $w_j^* = \sum_{i=1}^{N} \sum_{k=1}^{n} \mathbf{w}_i\,[h_{N,n}(j,i,k)]$,

and output $\mu = (\lceil w_j^* + v_j^* \rfloor_2)_{j\in[\ell]}$.

$\mathsf{PartDec}(\widetilde{\mathsf{CT}}, (\mathsf{PK}_1, \ldots, \mathsf{PK}_N), i, \mathsf{SK}_i) \to p_i$**.** Do the following:

1. parse $\widetilde{\mathsf{CT}} = ((v_j^*)_{j\in[\ell]}, \mathsf{ote.d})$, and $\mathsf{SK}_i = (\mathsf{sk}_i, \mathsf{ote.st}_{\mathsf{sk},i})$,
2. compute $\mathbf{w}_i \leftarrow \mathsf{OTE.EncodeEval}(\mathsf{ote.crs}, \mathsf{ote.d}, \mathsf{ote.st}_{\mathsf{sk},i})$,
3. for $j \in [\ell]$, compute $w_{j,i}^* = \sum_{k=1}^{n} \mathbf{w}_i\,[h_{N,n}(j,i,k)]$,

then if $i = 1$ output $p_i = (w_{j,i}^* + v_j^*)_{j\in[\ell]}$ and otherwise output $p_i = (w_{j,i}^*)_{j\in[\ell]}$.

$\mathsf{FinDec}(p_1, \ldots, p_N) \to \mu$**.** Parse $p_i = (p_{j,i})_{j\in[\ell]}$ and output $\left(\left\lceil \sum_{i=1}^{N} p_{j,i} \right\rfloor_2\right)_{j\in[\ell]}$.

Remark 4.2 (Uniformly Random Public Parameters.)**.** Note that the public parameters in our construction are a uniformly random string since the public parameters of the underlying MKFHE', and the CRS of the underlying OTE are each sampled uniformly at random[12].

[12]Looking ahead, the public parameters of Constructions 4.7 and A.1 are the same as here, possibly with an additional uniformly random PRF key, so this property holds for them as well.

Parameter Settings. Recall from Definition 3.10 that in the underlying MKFHE' where $n = \mathsf{poly}(\lambda, d)$, $m = n \log q + \omega(\log \lambda)$, and χ is a B_χ-bounded Gaussian error distribution, the evaluation error for computing depth-d circuits is bounded by $|e| \leq (mN+1)^d \cdot \mathsf{poly}(\lambda) \cdot B_\chi$ and MKFHE' correctness holds as long as for an even modulus q, we have $q \geq |e| \cdot \mathsf{poly}(\lambda)$, and the correctness of our construction holds as long as $q > 4(|e| + nN/2)$, and MKFHE' and OTE are correct. Hence, by letting $q > B_\chi \cdot 2^{\mathsf{poly}(\log \lambda, d, \log N)}$ for some $\mathsf{poly}(\cdot,\cdot,\cdot)$, subject to $q \geq 4\left((mN+1)^d \cdot B_\chi \cdot \mathsf{poly}(\lambda) + Nn/2\right)$, the correctness of our scheme follows and $\mathsf{LWE}_{q,n-1,m,\chi}$ (required for the security of the underlying MKFHE') holds.

Moreover, by Remark 3.15 and Theorem 3.16 for the underlying OTE, we let p be a multiple of q subject to $p > \mathsf{poly}(\lambda, q, \ell_{\mathbf{y}}) \cdot B'_\chi$. Namely, let $p > B'_\chi \cdot 2^{\mathsf{poly}(\log \lambda, \log q, \log \ell_{\mathbf{y}})}$, where by replacing $\log q$ (resp. $\log \ell_{\mathbf{y}}$) with $\mathsf{poly}(\log \lambda, d, \log N)$ (resp. $\mathsf{poly}(\log \lambda, \log d, \log N)$, since $\ell_{\mathbf{y}} = \ell n N$ and $\ell = \mathsf{poly}(\lambda)$) we get $p > B'_\chi \cdot 2^{\mathsf{poly}(\log \lambda, d, \log N)}$ for an appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$.

Theorem 4.3. Assuming perfectly correct succinct non-interactive oblivious tensor evaluation and multi-key fully homomorphic encryptions with split and near-linear decryption, the Construction 4.1 is a fully-succinct MKFHE where the evaluated ciphertext size is $\mathsf{poly}(\lambda, d, \log N, \ell)$ and the partial decryption sizes are $\mathsf{poly}(\log \lambda, d, \log N, \ell)$.

Corollary 4.4. Assuming LWE with $q/B_\chi = 2^{\mathsf{poly}(\log \lambda, d, \log N)}$, the Construction 4.1 is a fully-succinct MKFHE where the evaluated ciphertext size is $\mathsf{poly}(\lambda, d, \log N, \ell)$ and the partial decryption sizes are $\mathsf{poly}(\log \lambda, d, \log N, \ell)$.

Instantiations. Assuming LWE with modulus-to-noise ratio $q/B_\chi = 2^{\mathsf{poly}(\log \lambda, d, \log N)}$ and by combining Theorems 3.11, 3.16 and 4.3 there exists a fully-succinct MKFHE where $|\widetilde{\mathsf{CT}}|, |p_i| \leq \mathsf{poly}(\lambda, d, \log N, \ell)$ and $|\mathsf{PK}|, |\mathsf{SK}| \leq \mathsf{poly}(\lambda, d, \log N, \log \ell)$.

Moreover, by assuming the circular security of LWE, the dependence on d in our parameters disappears. Namely, assuming LWE with circular security and modulus-to-noise ratio $q/B_\chi = 2^{\mathsf{poly}(\log \lambda, \log N)}$, there exists a fully-succinct MKFHE where $|\widetilde{\mathsf{CT}}| \leq \mathsf{poly}(\lambda, \log N, \ell)$ and $|\mathsf{PK}|, |\mathsf{SK}| \leq \mathsf{poly}(\lambda, \log N, \log \ell)$.

4.2 Analysis

Overview. Before the formal analysis, we briefly explain why Construction 4.1 satisfies the required properties. For correctness, by the correctness of OTE and Eq. (1), for every $j \in [\ell]$ the values v_j^* (computed by the evaluator) and w_j^* (computed using the secret keys) satisfy $v_j^* + w_j^* = p/q \cdot \langle \widetilde{\mathsf{ct}}_j, \overline{\mathsf{sk}} \rangle + e' = p/2 \cdot C_j(\mu_1, \ldots, \mu_N) + e''$ for small errors e' and e''. Hence, rounding $v_j^* + w_j^*$ recovers the j^{th} output bit, and since the partial decryptions satisfy $\sum_{i \in [N]} p_{j,i} = v_j^* + w_j^*$, the same holds for distributed decryption. For succinctness, the evaluated ciphertext consists of one element of $\mathbb{Z}_p$ per output bit and the digest $\mathsf{ote.d}$, whose size is $\mathsf{poly}(\lambda, d, \log N, \log \ell)$ by the succinctness of OTE. Finally, for semantic security, by the encoder privacy of OTE we can replace the encoding $\mathsf{ote.}\hat{\mathsf{sk}}$ in the public key with a simulated one that is independent of sk, and then security follows from the semantic security of MKFHE', since a ciphertext of our scheme is just a ciphertext of MKFHE'.

Correctness. For $j \in [\ell]$ we have:

$$
\begin{aligned}
v_j^* + w_j^* &= \sum_{i=1}^{N}\sum_{k=1}^{n} \mathbf{v}_i\left[h_{N,n}(j,i,k)\right] + \sum_{i=1}^{N}\sum_{k=1}^{n} \mathbf{w}_i\left[h_{N,n}(j,i,k)\right] \\
&= \sum_{i=1}^{N}\sum_{k=1}^{n} \left(\mathbf{v}_i\left[h_{N,n}(j,i,k)\right] + \mathbf{w}_i\left[h_{N,n}(j,i,k)\right]\right) \\
&= \sum_{i=1}^{N}\sum_{k=1}^{n} p/q \cdot \lceil \mathbf{v}_i\left[h_{N,n}(j,i,k)\right] + \mathbf{w}_i\left[h_{N,n}(j,i,k)\right] \rfloor_q + e' \\
&\qquad \text{// where } |e'| \le Nn \cdot p/(2q) \text{ by the definition of rounding} \\
&= p/q \cdot \sum_{i=1}^{N}\sum_{k=1}^{n} (\widetilde{\mathsf{ct}} \otimes \mathsf{sk}_i)\left[h_{N,n}(j,i,k)\right] + e' \qquad \text{// by OTE correctness} \\
&= p/q \cdot \sum_{i=1}^{N}\sum_{k=1}^{n} \widetilde{\mathsf{ct}}[g_{N,n}(j,i,k)] \cdot \mathsf{sk}_i[k] + e' \qquad \text{// by Eq. (2)} \\
&= p/q \cdot \left(q/2 \cdot \mathsf{MKFHE}'.\mathsf{Dec}((\mathsf{sk}_i)_{i\in[N]}, \widetilde{\mathsf{ct}}_j) + e\right) + e' \qquad \text{// by Eq. (1)} \\
&= p/2 \cdot \mathsf{MKFHE}'.\mathsf{Dec}((\mathsf{sk}_i)_{i\in[N]}, \widetilde{\mathsf{ct}}_j) + e'' \qquad \text{// where } e'' = p/q \cdot e + e' \\
&= p/2 \cdot \mathsf{MKFHE}'.\mathsf{Dec}\left((\mathsf{sk}_i)_{i\in[N]}, \mathsf{MKFHE}'.\mathsf{Eval}\left(C_j, (\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}\right)\right) + e'' \\
&= p/2 \cdot C_j((\mu_i)_{i\in[N]}) + e'' \qquad \text{// by } \mathsf{MKFHE}' \text{ correctness.}
\end{aligned}
$$

where $|e''| \le |p/q \cdot e| + |e'| \le p/q \cdot |e| + Nn \cdot p/(2q) = p/q \cdot (|e| + Nn/2)$. Now, by our parameter setting we have $q > 4(|e| + nN/2)$, thus, we get $|e''| < p/4$, which implies that:

$$
\lceil v_j^* + w_j^* \rfloor_2 = \lceil p/2 \cdot C_j((\mu_i)_{i\in[N]}) + e'' \rfloor_2 = C_j((\mu_i)_{i\in[N]}) + \lceil e'' \rfloor_2 = C_j((\mu_i)_{i\in[N]}).
$$

Correctness of Partial Decryption. For $j \in [\ell]$ we have:

$$
\left\lceil \sum_{i=1}^{N} p_{j,i} \right\rfloor_2 = \left\lceil v_j^* + \sum_{i=1}^{N} w_{j,i}^* \right\rfloor_2 = \lceil v_j^* + w_j^* \rfloor_2 = C_j((\mu_i)_{i\in[N]})
$$

where the last equality holds similar to the correctness argument.

Compactness, and Succinctness. First note that the field elements in our scheme grow with $\log p = \mathsf{poly}(\log\lambda, d, \log N)$, hence, $|\mathsf{sk}| = \mathsf{poly}(\lambda, d, \log N)$, and $|\widetilde{\mathsf{ct}}| = \mathsf{poly}(\lambda, d, N, \ell)$.

Thus, by the succinctness of OTE it holds that:

$$
|\mathsf{ote.crs}|, |\mathsf{ote.d}|, |\mathsf{ote.}\hat{\mathsf{sk}}|, |\mathsf{ote.st_{sk}}| \le \mathsf{poly}(\lambda, \log p, |\mathsf{sk}|, \log|\widetilde{\mathsf{ct}}|) \le \mathsf{poly}(\lambda, d, \log N, \log\ell).
$$

Therefore, we can bound the evaluated ciphertext size as:

$$
|\widetilde{\mathsf{CT}}| = \ell \cdot |v_j^*| + |\mathsf{ote.d}| \le \ell \cdot \mathsf{poly}(\log\lambda, d, \log N) + \mathsf{poly}(\lambda, d, \log N, \log\ell) = \mathsf{poly}(\lambda, d, \log N, \ell),
$$

and, we can bound the partial decryption sizes as:

$$
|p_i| = \ell \cdot |w_{j,i}^*| \le \ell \cdot \mathsf{poly}(\log\lambda, d, \log N) = \mathsf{poly}(\log\lambda, d, \log N, \ell).
$$

Moreover, since $|\mathsf{PP}| = |\mathsf{pp}| + |\mathsf{ote.crs}|$, $|\mathsf{PK}| = |\mathsf{pk}| + |\mathsf{ote.}\hat{\mathsf{sk}}|$ and $|\mathsf{SK}| = |\mathsf{sk}| + |\mathsf{ote.st_{sk}}|$, we can bound all as:

$$|\mathsf{PP}|, |\mathsf{PK}|, |\mathsf{SK}| \leq \mathsf{poly}(\lambda, d, \log N, \log \ell).$$

Semantic Security. We define the following hybrids:

- hyb_0**.** This is the original semantic security experiment where the adversary upon receiving $(\mathsf{PP}, \mathsf{PK})$ where $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$ and $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{KeyGen}(\mathsf{PP})$, sends (μ_0, μ_1) to the challenger. Then the challenger chooses a bit $b \leftarrow \{0,1\}$ and sends $\mathsf{CT}_b \leftarrow \mathsf{Enc}(\mathsf{PK}, \mu_b)$ to $\mathcal{A}$. Finally $\mathcal{A}$ outputs b'. Then the experiment outputs 1 if $b' = b$.
- hyb_1**.** This is similar to hyb_0 except that the challenger during the key generation, computes $\mathsf{ote.}\hat{\mathsf{sk}}$ (as part of PK) by running the simulator of the OTE encoder privacy $\mathsf{ote.}\hat{\mathsf{sk}} \leftarrow \mathsf{OTE}.\mathcal{S}(\mathsf{ote.crs}, 1^n)$.

The indistinguishability of hybrids hyb_0 and hyb_1 follows by the encoder privacy of OTE. Next, in hyb_1, there is no information about sk in PK, thus, the semantic security of our scheme follows from the semantic security of the underlying MKFHE'. More formally let $\mathsf{hyb}_i(\mathcal{A})$ denote the output of the experiment in hybrid hyb_i when run on adversary $\mathcal{A}$.

Lemma 4.5. If OTE satisfies encoder privacy, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$|\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_0(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose towards the contradiction that $\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_0(\mathcal{A}) = 1] \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We construct adversary $\mathcal{B}$ against the encoder privacy of OTE as follows:

1. Run adversary $\mathcal{A}$ to receive 1^d. Run challenger to receive $\mathsf{ote.crs}$.
2. Sample $\mathsf{pp} \leftarrow \mathsf{MKFHE}'.\mathsf{Setup}(1^\lambda, 1^d)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{MKFHE}'.\mathsf{KeyGen}(\mathsf{pp})$.
3. Send $x = \mathsf{sk}$ to the challenger, and receive $\hat{x}_{b^*}$ from the challenger.
4. Send $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs})$ and $\mathsf{PK} = (\mathsf{pk}, \hat{x}_{b^*})$ to $\mathcal{A}$, and receive (μ_0, μ_1) from $\mathcal{A}$
5. Sample $b \leftarrow \{0,1\}$, and send $\mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}, \mu_b)$ to $\mathcal{A}$.
6. Receive b' from $\mathcal{A}$ and output 1 if $b' = b$

Note that $\mathcal{B}$ perfectly simulates the challenger of (1) hyb_0 if $b^* = 0$ (in which case $\hat{x}_{b^*}$ is the output of encoding algorithm), and, (2) hyb_1 if $b^* = 1$ (in which case $\hat{x}_{b^*}$ is the output of the simulator). Hence

$$\begin{aligned}
\Pr[\mathcal{B}(\hat{x}_{b^*}) = b^*] &= 1/2 \cdot (\Pr[\mathcal{B}(\hat{x}_0) = 0] + \Pr[\mathcal{B}(\hat{x}_1) = 1]) \\
&= 1/2 \cdot (1 - \Pr[\mathcal{B}(\hat{x}_0) = 1] + \Pr[\mathcal{B}(\hat{x}_1) = 1]) \\
&= 1/2 + 1/2 \cdot (\Pr[\mathcal{B}(\hat{x}_1) = 1] - \Pr[\mathcal{B}(\hat{x}_0) = 1]) \\
&= 1/2 + 1/2 \cdot (\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_0(\mathcal{A}) = 1]) \\
&\geq 1/2 + \epsilon(\lambda)/2
\end{aligned}$$

which contradicts the encoder privacy of OTE. □

Lemma 4.6. If MKFHE' satisfies semantic security, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] \leq 1/2 + \mathsf{negl}(\lambda).$$

Proof. Suppose towards the contradiction that $\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] \geq 1/2+\epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We construct adversary $\mathcal{B}$ against the semantic security of MKFHE' as follows:

1. Run adversary $\mathcal{A}$ to receive 1^d, then send 1^d to the challenger.
2. Sample $\mathsf{ote.crs} \leftarrow \mathsf{OTE.Setup}(1^\lambda)$, and compute $\mathsf{ote.}\hat{\mathsf{sk}} \leftarrow \mathsf{OTE}.\mathcal{S}(\mathsf{ote.crs}, 1^n)$.
3. Receive $(\mathsf{pp}, \mathsf{pk})$ from the challenger and send $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs})$ and $\mathsf{PK} = (\mathsf{pk}, \mathsf{ote.}\hat{\mathsf{sk}})$ to $\mathcal{A}$.
4. Receive (μ_0, μ_1) from $\mathcal{A}$ and send (μ_0, μ_1) to the challenger.
5. Receive ct_b from the challenger and send it to $\mathcal{A}$.
6. Output whatever $\mathcal{A}$ outputs.

Note that $\mathcal{B}$ perfectly simulates the challenger of hyb_1 for $\mathcal{A}$. Hence, it has the same advantage $\epsilon(\cdot)$ in breaking the semantic security of the underlying MKFHE' that contradicts the security of MKFHE'. □

We conclude the proof by combining Lemmas 4.5 and 4.6.

4.3 Fully-Succinct MKFHE With Multi-Hop Evaluation

Bootstrapping (single-key). Gentry's bootstrapping paradigm [Gen09] upgrades a *single-hop* FHE scheme into a *multi-hop* one by homomorphically evaluating (a low-depth implementation of) the decryption procedure. The key observation is that one can "refresh" ciphertexts at each hop by evaluating (a layer of) the decryption circuit inside the scheme, provided decryption has sufficiently small depth. In prior LWE-based constructions (e.g., [GSW13]), decryption is essentially linear in the secret key, so the resulting decryption circuit has depth $O(\log |\mathsf{sk}|) = \mathsf{polylog}(\lambda)$.

Bootstrapping (multi-key). The same blueprint extends to multi-key FHE in a straightforward manner: any *single-hop* MKFHE with low decryption depth can be bootstrapped into a *multi-hop* MKFHE. Concretely, for each party $i \in [N]$ and each hop $k \in [h+1]$ (where h is an a-priori bound on the number of hops), key generation samples single-hop MKFHE keys $(\mathsf{pk}_k^{(i)}, \mathsf{sk}_k^{(i)})$ and for $k \in [h]$ additionally publishes encrypted values $\mathsf{ct}_{\mathsf{sk},k+1}^{(i)} \leftarrow \mathsf{Enc}_{\mathsf{pk}_{k+1}^{(i)}}(\mathsf{sk}_k^{(i)})$.[13] At hop k, suppose we wish to evaluate a circuit C_{Eval} on plaintext inputs $(\mu_1, \ldots, \mu_t)$, given ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_t)$ where each ct_i encrypts μ_i under $(\mathsf{pk}_k^{(1)}, \ldots, \mathsf{pk}_k^{(N)})$. The evaluator proceeds as follows:

1. Define a decryption circuit C_{Dec} with $(\mathsf{ct}_1, \ldots, \mathsf{ct}_t)$ hardwired, that takes $(\mathsf{sk}_k^{(1)}, \ldots, \mathsf{sk}_k^{(N)})$ as input and outputs $(\mu_1, \ldots, \mu_t)$.
2. Let $C_{\mathsf{Eval}'} := C_{\mathsf{Eval}} \circ C_{\mathsf{Dec}}$ denote the composition that first decrypts and then applies C_{Eval}.

[13] This key chain can be replaced by a single key pair per party that encrypts its own secret key under an appropriate circular-security assumption (e.g., for LWE).

3. Perform a single-hop *multi-key* evaluation under $(\mathsf{pk}_{k+1}^{(1)}, \ldots, \mathsf{pk}_{k+1}^{(N)})$ on inputs $(\mathsf{ct}_{\mathsf{sk},k+1}^{(1)}, \ldots, \mathsf{ct}_{\mathsf{sk},k+1}^{(N)})$ for circuit $C_{\mathsf{Eval}'}$.

The resulting ciphertext encrypts $C_{\mathsf{Eval}}(\mu_1, \ldots, \mu_t)$ under $(\mathsf{pk}_{k+1}^{(1)}, \ldots, \mathsf{pk}_{k+1}^{(N)})$. Each hop evaluates $C_{\mathsf{Eval}'}$, of depth $d_{\mathsf{Dec}} + d_{\mathsf{Eval}}$ (where d_{Dec} and d_{Eval} are the depths of C_{Dec} and C_{Eval}, respectively), on fresh encryptions of the secret keys; hence the noise does not accumulate across hops, and the modulus q only needs to support depth $d_{\mathsf{Dec}} + d_{\mathsf{Eval}}$, while the number of keys grows with h.[14] In existing MKFHE schemes (e.g., [MW16]), decryption is blockwise and linear in the aggregated secret key, which is typically the concatenation of per-party secret keys; hence decryption remains a low-depth computation.

Instantiating bootstrapping for our scheme. Following the above approach, it suffices that the decryption procedure is implementable by a short-depth circuit as a function of the secret-key inputs. Recall from Construction 4.1 that decryption in our construction primarily computes

$$\mathbf{w}_i \leftarrow \mathsf{OTE.EncodeEval}(\mathsf{ote.crs}, \mathsf{ote.d}, \mathsf{ote.st}_{\mathsf{sk},i}) \qquad \text{for all } i \in [N],$$

where $\mathsf{ote.st}_{\mathsf{sk},i}$ denotes the (per-party) secret key, and then computes

$$\mu_j = \left\lceil v_j^* + \sum_{i=1}^{N} \sum_{k=1}^{n} \mathbf{w}_i \left[h_{N,n}(j,i,k)\right] \right\rfloor_2 \qquad \text{for all } j \in [\ell].$$

Note that taking the summation over $i \in [N]$ and $k \in [n]$ can be computed by a circuit of depth $\mathsf{polylog}(N, n)$ (using a tree structure), and rounding can be implemented in constant depth. Consequently, it is enough that $\mathsf{OTE.EncodeEval}$ admits a low-depth circuit in the input $\mathsf{ote.st}_{\mathsf{sk},i}$.

We observe that in the OTE construction of [AMR25], the encode-evaluation algorithm is linear in $\mathsf{ote.st}_{\mathsf{sk},i}$. Therefore, in an unbounded fan-out circuit model, $\mathsf{OTE.EncodeEval}$ can be implemented in depth $O(\log |\mathsf{ote.st}_{\mathsf{sk},i}|)$.[15] Since our parameters satisfy $|\mathsf{ote.st}_{\mathsf{sk},i}| \leq \mathsf{poly}(\lambda, d, \log N, \log \ell)$, the resulting decryption depth is $\mathsf{polylog}(\lambda)$ and only mildly dependent on N. This yields a bootstrapped (multi-hop) variant of our scheme via the above transformation.

Looking ahead, in Construction 5.1 (where we achieve rate-1 simulatable partial decryptions) we effectively bootstrap ciphertexts produced by Construction 4.1; accordingly, we refer the reader to that construction for details on why (and how) our scheme supports bootstrapping.

4.4 Achieving Circuit Privacy

Many applications including our construction of Threshold Trapdoor Hashing, require MKFHE with circuit privacy. Next, we briefly describe how we can achieve that for free in our construction and refer the reader to Appendix A for a complete construction and analysis of our circuit private MKFHE. To achieve circuit privacy, in Construction 4.1, the evaluation samples $(\mathsf{pk}_{N+1}, \mathsf{sk}_{N+1})$ and encrypts the circuit C_j to compute $\mathsf{ct}_{C,j}$. Then it performs the MKFHE evaluation on a universal circuit with inputs $(\mathsf{ct}_1, \ldots, \mathsf{ct}_N)$ and $\mathsf{ct}_{C,j}$ to compute an evaluated ciphertext $\widetilde{\mathsf{ct}}$. Next, the evaluation computes a decryption share of $\widetilde{\mathsf{ct}}$ that aligns with sk_{N+1} and includes the decryption share, masked by a smudging noise, in the ciphertext, and finally performs the 2PC OTE computation only on the rest of ciphertext.

[14] This dependency disappears under an appropriate circular-security assumption (e.g., for LWE).
[15] With constant fan-out, the depth also incurs an additional $O(\log |\mathbf{w}_i|)$ term, i.e., $O(\log(\ell n N \log p))$ in our setting.

4.5 Rate-1 Fully-Succinct MKFHE

Recall that a MKFHE scheme is rate-1 if $|\widetilde{\mathsf{ct}}| \leq |C(\mu_1, \ldots, \mu_N)| + \mathsf{poly}(\lambda, d, N)$. Here, we show how to modify Construction 4.1 to additionally achieve rate-1 ciphertext size.

Construction 4.7. Let OTE be a perfectly correct succinct non-interactive oblivious tensor evaluation, MKFHE' be a multi-key fully homomorphic encryption with split and near-linear decryption and $F(\cdot)$ be a PRF.

$\mathsf{Setup}(1^\lambda, 1^d) \to \mathsf{PP}$**.** Sample $\mathsf{pp}, \mathsf{ote.crs}$ similar to Construction 4.1, then sample $K \leftarrow \{0,1\}^\lambda$ and let $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs}, K)$.

$\mathsf{KeyGen}(\mathsf{PP}) \to (\mathsf{PK}, \mathsf{SK})$**.** Similar to Construction 4.1.

$\mathsf{Enc}(\mathsf{PK}, \mu) \to \mathsf{CT}$**.** Similar to Construction 4.1

$\mathsf{Eval}(C, (\mathsf{PK}_i, \mathsf{CT}_i)_{i\in[N]}) \to \widetilde{\mathsf{CT}}$**.** First compute $(v_j^*)_{j\in[\ell]}$ and $\mathsf{ote.d}$ similar to Construction 4.1, then for $j \in [\ell]$ let $z_j = \left\lceil F_K(j) + v_j^* \right\rfloor_2$ and output $\widetilde{\mathsf{CT}} = ((z_j)_{j\in[\ell]}, \mathsf{ote.d})$.

$\mathsf{Dec}((\mathsf{SK}_i)_{i\in[N]}, \widetilde{\mathsf{CT}}) \to \mu$**.** First compute $(w_j^*)_{j\in[\ell]}$ similar to Construction 4.1, then for $j \in [\ell]$ let $z'_j = \left\lceil -F_K(j) + w_j^* \right\rfloor_2$ and output $\mu = (z_j + z'_j)_{j\in[\ell]}$.

$\mathsf{PartDec}(\widetilde{\mathsf{CT}}, (\mathsf{PK}_1, \ldots, \mathsf{PK}_N), i, \mathsf{SK}_i) \to p_i$**.** First compute $(w_{j,i}^*)_{j\in[\ell]}$ similar to Construction 4.1. Then if $i = 1$ output $p_i = (w_{j,i}^*, z_j)_{j\in[\ell]}$ and otherwise output $p_i = (w_{j,i}^*)_{j\in[\ell]}$.

$\mathsf{FinDec}(p_1, \ldots, p_N) \to \mu$**.** Parse $p_1 = (w_{j,1}^*, z_j)_{j\in[\ell]}$ and for $i \in [2, N]$, $p_i = (w_{j,i}^*)_{j\in[\ell]}$, then for $j \in [\ell]$ let $z'_j = \left\lceil -F_K(j) + \sum_{i=1}^N w_{j,i}^* \right\rfloor_2$ and output $\mu = (z_j + z'_j)_{j\in[\ell]}$.

Parameter Settings. In addition to the parameter setting analysis of Construction 4.1 (where $|e| \leq (mN+1)^d \cdot \mathsf{poly}(\lambda) \cdot B_\chi$), for the correctness of our construction, we require $p/(2|e''|) \geq \lambda^{\omega(1)}$ where $|e''| \leq p/q \cdot (|e| + Nn/2)$. This can be satisfied by having $q > 2(|e| + Nn/2) \cdot \lambda^{\omega(1)}$. Hence, by letting $q > B_\chi \cdot 2^{\mathsf{poly}(\lambda, d, \log N)}$ for some appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$, subject to $q \geq 2((mN+1)^d \cdot B_\chi \cdot \mathsf{poly}(\lambda) + Nn/2) \cdot \lambda^{\omega(1)}$ the correctness of our construction follows. Similarly, we get $p > B'_\chi \cdot 2^{\mathsf{poly}(\lambda, d, \log N)}$ for an appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$.

Theorem 4.8. Assuming perfectly correct succinct non-interactive oblivious tensor evaluation, multi-key fully homomorphic encryptions with split and near-linear decryption, and PRFs, the Construction 4.7 is a fully-succinct and rate-1 MKFHE where the evaluated ciphertext size is $\ell + \mathsf{poly}(\lambda, d, \log N, \log \ell)$ and the partial decryption sizes are $\mathsf{poly}(\lambda, d, \log N, \ell)$.

Corollary 4.9. Assuming LWE with a sub-exponential modulus-to-noise ratio where $q/B_\chi = 2^{\mathsf{poly}(\lambda, d, \log N)}$, the Construction 4.7 is a fully-succinct and rate-1 MKFHE where the evaluated ciphertext size is $\ell + \mathsf{poly}(\lambda, d, \log N, \log \ell)$ and the partial decryption sizes are $\mathsf{poly}(\lambda, d, \log N, \ell)$.

Correctness. Recall from the correctness analysis of Construction 4.1 that for $j \in [\ell]$, $v_j^* + w_j^* = p/2 \cdot C_j((\mu_i)_{i\in[N]}) + e''$ where $|e''| \le p/q \cdot (|e| + Nn/2)$. Next, note that by our parameter setting $p/(2|e''|) \ge \lambda^{\omega(1)}$. Moreover, PRF randomizes the shares z_j and z'_j for $j \in [\ell]$. Therefore, using Lemma 3.1, for $j \in [\ell]$ we have:

$$\Pr\left[z_j + z'_j = \left\lceil F_K(j) + v_j^* - F_K(j) + w_j^* \right\rfloor_2\right] \ge 1 - \mathsf{negl}(\lambda),$$

and, we conclude the proof using $\left\lceil v_j^* + w_j^* \right\rfloor_2 = C_j((\mu_i)_{i\in[N]})$ for $j \in [\ell]$ (which holds by the correctness of Construction 4.1).

Note that applying Lemma 3.1 is the reason we need a super-polynomial modulus-to-noise ratio in this construction.

Correctness of Partial Decryption. Similar to the correctness argument except that each party i computes $w_{j,i}^*$ and the final decryption puts them together to compute z'_j.

Compactness, Succinctness, and Rate-1. The analysis is similar to Construction 4.1 except that we can bound the evaluated ciphertext size as:

$$|\widetilde{\mathsf{CT}}| = \ell \cdot |z_j| + |\mathsf{ote.d}| \le \ell + \mathsf{poly}(\lambda, d, \log N, \log \ell),$$

and since we are using super-polynomial modulus, each field elements grows with $\log p = \mathsf{poly}(\lambda, d, \log N)$ which makes the partial decryptions grow with

$$|p_i| \le \ell \cdot (|w_{j,i}^*| + 1) \le \ell \cdot \mathsf{poly}(\lambda, d, \log N) = \mathsf{poly}(\lambda, d, \log N, \ell).$$

Semantic Security. Similar to Construction 4.1.

5 Fully-Succinct MKFHE With Rate-1 Simulatable Distributed Decryption

In this section, we show how to construct a wrapper around Construction 4.1 to achieve simulatable and rate-1 partial decryption while keeping the ciphertext size fully-succinct.

Notation. For ease of exposition, in the partial decryption algorithm in the following construction we assume $N = 2^r$. In the following construction and its analysis, for any indexed variable z_i, by z_S we denote $(z_i)_{i\in S}$. Moreover, we let $\theta = 2B_{\mathsf{eval}} + 1$, where B_{eval} is a bound on the MKFHE' error and ℓ_{hash} is the output length of CIH. For $v \in \mathbb{Z}_q$, we say that $u \in [0, 2^{\ell_{\mathsf{hash}}} - 1]$ is a *bad shift* for v if $\left||v - \theta \cdot u| - q/4\right| \le B_{\mathsf{eval}}$ (where for $w \in \mathbb{Z}_q$, $|w|$ denotes the absolute value of the representative of w in $(-q/2, q/2]$), and we let $\mathsf{Bad}(v) \in \{0,1\}^{\ell_{\mathsf{hash}}}$ be the binary representation of the smallest bad shift for v, or $0^{\ell_{\mathsf{hash}}}$ if there is none. For an FHE secret key $\mathsf{fhe.sk}$, we let $\mathsf{Bad}_{\mathsf{fhe.sk}}$ be the circuit that on input an FHE ciphertext $\widehat{\mathsf{ct}}$ outputs $\mathsf{Bad}(\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \widehat{\mathsf{ct}}))$, where the output of $\mathsf{FHE.Dec}$ is interpreted as an element of $\mathbb{Z}_q$ (by reducing the integer represented by its first $\lceil \log q \rceil$ bits modulo q, or as 0 if decryption fails). Let $S = \mathsf{poly}(\lambda, \log q)$ be the bound on the size of the circuit $\mathsf{Bad}_{\mathsf{fhe.sk}}$.

5.1 Construction

Construction 5.1. Let OTE be a perfectly correct succinct non-interactive oblivious tensor evaluation, MKFHE' be a multi-key fully homomorphic encryption with split and near-linear decryption, and FHE be a (leveled) fully homomorphic encryption scheme. We further assume perfect correctness and deterministic evaluation for both MKFHE' and FHE. Moreover, let CIH be a somewhere perfectly correlation intractable hash family for circuits of size S with output length ℓ_{hash} (see Section 3.4).

$\mathsf{Setup}(1^\lambda, 1^d) \to \mathsf{PP}$**.** Sample pp (supporting depth d') and $\mathsf{ote.crs}$ similar to Construction 4.1, then sample an FHE key pair $(\mathsf{fhe.pk}, \mathsf{fhe.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^{d_{\mathsf{fhe}}})$ and $\mathsf{hk} \leftarrow \mathsf{CIH.StatGen}(1^\lambda, \langle \mathsf{Bad}_{\mathsf{fhe.sk}} \rangle)$, and output $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs}, \mathsf{fhe.pk}, \mathsf{hk})$. See parameter settings for d' and d_{fhe}.

$\mathsf{KeyGen}(\mathsf{PP}) \to (\mathsf{PK}, \mathsf{SK})$**.** First sample and compute $(\mathsf{pk}, \mathsf{sk}, \mathsf{ote.}\hat{\mathsf{sk}}, \mathsf{ote.st}_{\mathsf{sk}})$ similar to Construction 4.1. Then sample $(\mathsf{pk}^{(t)}, \mathsf{sk}^{(t)}) \leftarrow \mathsf{MKFHE}'.\mathsf{KeyGen}(\mathsf{pp})$ for $t \in [0, \lambda]$. Then compute $\mathsf{ct}_{\mathsf{sk}}^{(t)} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}^{(t)}, \mathsf{sk}^{(t-1)})$ for $t \in [\lambda]$, $\mathsf{ct}_{\mathsf{sk}}^{(0)} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}^{(0)}, \mathsf{ote.st}_{\mathsf{sk}})$, and $\mathsf{ct}_{\mathsf{fhe}}^{(t)} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, \mathsf{sk}^{(t)})$ for $t \in [0, \lambda - 1]$. Output $\mathsf{SK} = (\mathsf{sk}, \mathsf{ote.st}_{\mathsf{sk}}, (\mathsf{sk}^{(t)})_{t\in[0,\lambda]})$ and $\mathsf{PK} = (\mathsf{pk}, \mathsf{ote.}\hat{\mathsf{sk}}, (\mathsf{pk}^{(t)})_{t\in[0,\lambda]}, (\mathsf{ct}_{\mathsf{sk}}^{(t)})_{t\in[0,\lambda]}, (\mathsf{ct}_{\mathsf{fhe}}^{(t)})_{t\in[0,\lambda-1]})$.

$\mathsf{Enc}(\mathsf{PK}, \mu) \to \mathsf{CT}$**.** Similar to Construction 4.1.

$\mathsf{Eval}(C, (\mathsf{PK}_i, \mathsf{CT}_i)_{i\in[N]}) \to \widetilde{\mathsf{CT}}$**.** Similar to Construction 4.1.

$\mathsf{Dec}((\mathsf{SK}_i)_{i\in[N]}, \widetilde{\mathsf{CT}}) \to \mu$**.** Similar to Construction 4.1.

$\mathsf{PartDec}(\widetilde{\mathsf{CT}}, (\mathsf{PK}_1, \ldots, \mathsf{PK}_N), \alpha, \mathsf{SK}_\alpha) \to p_\alpha$**.** Do the following:

1. parse $\widetilde{\mathsf{CT}} = ((v_j^*)_{j\in[\ell]}, \mathsf{ote.d})$, and $\mathsf{SK}_\alpha = (\mathsf{sk}_\alpha, \mathsf{ote.st}_{\mathsf{sk},\alpha}, (\mathsf{sk}_\alpha^{(t)})_{t\in[0,\lambda]})$, and for $i \in [N]$, $\mathsf{PK}_i = (\mathsf{pk}_i, \mathsf{ote.}\hat{\mathsf{sk}}_i, (\mathsf{pk}_i^{(t)})_{t\in[0,\lambda]}, (\mathsf{ct}_{\mathsf{sk},i}^{(t)})_{t\in[0,\lambda]}, (\mathsf{ct}_{\mathsf{fhe},i}^{(t)})_{t\in[0,\lambda-1]})$,
2. define $C_j^{(0)}(x_{[N]})$ to be the decryption circuit for j^{th} bit from Construction 4.1 (where $x_i = \mathsf{ote.st}_{\mathsf{sk},i}$); namely, it has $(\mathsf{ote.crs}, (v_j^*), \mathsf{ote.d})$ hardcoded in it and computes the following:
   (a) for $i \in [N]$ compute $\mathbf{w}_i \leftarrow \mathsf{OTE.EncodeEval}(\mathsf{ote.crs}, \mathsf{ote.d}, x_i)$,
   (b) compute $w_j^* = \sum_{i=1}^N \sum_{k=1}^n \mathbf{w}_i[h_{N,n}(j,i,k)]$,
   (c) output $\lceil w_j^* + v_j^* \rfloor_2$,
3. for $j \in [\ell]$ compute $\widetilde{\mathsf{CT}}_j^{(0)} = \mathsf{MKFHE}'.\mathsf{Eval}\left(C_j^{(0)}, \left(\mathsf{pk}_i^{(0)}, \mathsf{ct}_{\mathsf{sk},i}^{(0)}\right)_{i\in[N]}\right)$,
4. for $j \in [\ell]$ parse $\widetilde{\mathsf{CT}}_j^{(0)} = (\widetilde{\mathsf{ct}}_{j,1}^{(0)}, \ldots, \widetilde{\mathsf{ct}}_{j,N}^{(0)})$,
5. For $t \in [r]$, $k \in [2^t]$, and $j \in [\ell]$ do the following:
   (a) let $S_k^{(t)} = [a_k^{(t)}, b_k^{(t)}]$ represent the k^{th} block of size $N/2^t$ over $[N]$, more specifically, $a_k^{(t)} = (k-1)(N/2^t) + 1$ and $b_k^{(t)} = k(N/2^t)$,
   (b) let $k' = \lceil k/2 \rceil$ and define $\widehat{C}_{j,k'}^{(t)}(x_{S_{2k'-1}^{(t)}}) \coloneqq \sum_{i\in S_{2k'-1}^{(t)}} x_i \cdot \widetilde{\mathsf{ct}}_{j,i}^{(t-1)}$ where the circuit $\widehat{C}_{j,k'}^{(t)}$ has $\widetilde{\mathsf{ct}}_{j,S_{2k'-1}^{(t)}}^{(t-1)}$ hardcoded in it,

(c) compute $\widehat{\mathsf{ct}}^{(t)}_{j,k'} = \mathsf{FHE.Eval}\left(\widehat{C}^{(t)}_{j,k'}, \mathsf{fhe.pk}, \left(\mathsf{ct}^{(t-1)}_{\mathsf{fhe},i}\right)_{i\in S^{(t)}_{2k'-1}}\right)$,

(d) compute $\delta^{(t)}_{j,k'} = \theta \cdot \mathsf{CIH.Hash}(\mathsf{hk}, \widehat{\mathsf{ct}}^{(t)}_{j,k'})$, where the hash value is interpreted as an integer in $[0, 2^{\ell_{\mathsf{hash}}} - 1]$,

(e) define

$$C^{(t)}_{j,k}(x_{S^{(t)}_k}) := \left\lfloor (-1)^k \delta^{(t)}_{j,\lceil k/2\rceil} + \sum_{i\in S^{(t)}_k} x_i \cdot \widetilde{\mathsf{ct}}^{(t-1)}_{j,i} \right\rceil_2$$

where $C^{(t)}_{j,k}$ has $\left(\delta^{(t)}_{j,\lceil k/2\rceil}, \widetilde{\mathsf{ct}}^{(t-1)}_{j,S^{(t)}_k}\right)$ values hardcoded in it,

(f) compute $\widetilde{\mathsf{CT}}^{(t)}_{j,k} = \mathsf{MKFHE'.Eval}\left(C^{(t)}_{j,k}, \left(\mathsf{pk}^{(t)}_i, \mathsf{ct}^{(t)}_{\mathsf{sk},i}\right)_{i\in S^{(t)}_k}\right)$,

(g) parse $\widetilde{\mathsf{CT}}^{(t)}_{j,k} = (\widetilde{\mathsf{ct}}^{(t)}_{j,a^{(t)}_k}, \ldots, \widetilde{\mathsf{ct}}^{(t)}_{j,b^{(t)}_k})$,

and output $p_\alpha = \left(\left\lceil \widetilde{\mathsf{ct}}^{(r)}_{j,\alpha} \cdot \mathsf{sk}^{(r)}_\alpha \right\rfloor_2\right)_{j\in[\ell]}$.

$\mathsf{FinDec}(p_1, \ldots, p_N) \to \mu$. Parse $p_i = (p_{j,i})_{j\in[\ell]}$ and output $\left(\sum_{i=1}^N p_{j,i}\right)_{j\in[\ell]}$.

Remark 5.2. In the presentation of the partial decryption algorithm in Construction 5.1, each party is computing the entire bootstrapping tree, i.e., all the $\widetilde{\mathsf{ct}}^{(r)}_{j,i}$ values for $i \in [N]$, however, the partial decryption only needs the $\widetilde{\mathsf{ct}}^{(r)}_{j,\alpha}$ value. Therefore, one can optimize the partial decryption to do the bootstrapping only on the path from the root to the $\widetilde{\mathsf{ct}}^{(r)}_{j,\alpha}$ leaf.

Parameter Settings. In addition to the parameter setting analysis of Construction 4.1 (where $|e| \le (mN+1)^d \cdot \mathsf{poly}(\lambda) \cdot B_\chi$), for the correctness and simulatability of our construction, we require $|e| \le B_{\mathsf{eval}}$ for the error e of every $\mathsf{MKFHE'}$ evaluated ciphertext computed in $\mathsf{PartDec}$, and $q \ge 2^{\ell_{\mathsf{hash}}+1} \cdot \theta$ where $\theta = 2B_{\mathsf{eval}} + 1$. Since the above bound on $|e|$ grows with N, and our key generation supports up to $N = 2^\lambda$ parties (as $\mathsf{PartDec}$ uses the keys of levels $t \in [0, r]$ where $2^r = N$), we let $B_{\mathsf{eval}} = (m2^\lambda + 1)^{d'} \cdot \mathsf{poly}(\lambda) \cdot B_\chi$, which can be computed by Setup, where d' is the maximum of d and the depth of the bootstrapping circuits $C^{(0)}_j$ and $C^{(t)}_{j,k}$, which is $\mathsf{polylog}(\lambda, d, N)$ (see Section 4.3) and hence only grows polylogarithmically with q. Moreover, by Theorem 3.23, we can let the output length of CIH be $\ell_{\mathsf{hash}} = \lambda$, independently of S. Hence, by letting $q > B_\chi \cdot 2^{\mathsf{poly}(\lambda,d,\log N)}$ for some appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$, subject to $q \ge 2^{\ell_{\mathsf{hash}}+1} \cdot (2B_{\mathsf{eval}}+1)$, the correctness and simulatability of our construction follow. Similarly, we get $p > B'_\chi \cdot 2^{\mathsf{poly}(\lambda,d,\log N)}$ for an appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$. Finally, each circuit $\widehat{C}^{(t)}_{j,k'}$ computes an inner product of length at most $Nn \le 2^\lambda n$ over $\mathbb{Z}_q$, hence it can be implemented with depth $d_{\mathsf{fhe}} = \mathsf{poly}(\lambda, \log q)$, and each circuit $\mathsf{Bad}_{\mathsf{fhe.sk}}$ decrypts an FHE ciphertext of a $\lceil \log q \rceil$-bit value and then computes Bad in time $\mathsf{poly}(\log q)$ (since $\theta(2^{\ell_{\mathsf{hash}}} - 1) + 2B_{\mathsf{eval}} < q/2$, at most one rounding boundary is within distance B_{eval} of the values $v - \theta \cdot u$ for $u \in [0, 2^{\ell_{\mathsf{hash}}} - 1]$), hence it has size $S = \mathsf{poly}(\lambda, \log q)$. Note that there is no circularity in these choices, since ℓ_{hash} is chosen first, then q and B_{eval} (where d' only grows polylogarithmically with q), then FHE for depth d_{fhe}, and finally CIH for size S.

Theorem 5.3. Assuming perfectly correct succinct non-interactive oblivious tensor evaluation, perfectly correct multi-key fully homomorphic encryptions with split and near-linear decryption and deterministic evaluation, perfectly correct fully homomorphic encryptions with deterministic evaluation, and somewhere perfectly correlation intractable hash functions for circuits, the Construction 5.1 is a fully-succinct MKFHE with simulatable partial decryptions and a trusted setup where the evaluated ciphertext size is $\mathsf{poly}(\lambda, d, \log N, \ell)$ and the partial decryptions are ℓ bits.

Corollary 5.4. Assuming LWE with a sub-exponential modulus-to-noise ratio where $q/B_\chi = 2^{\mathsf{poly}(\lambda,d,\log N)}$ and LWE with a polynomial modulus-to-noise ratio (against non-uniform adversaries), the Construction 5.1 is a fully-succinct MKFHE with simulatable partial decryptions where the evaluated ciphertext size is $\mathsf{poly}(\lambda, d, \log N, \ell)$ and the partial decryptions are ℓ bits.

The corollary follows by instantiating MKFHE', OTE, FHE, and CIH with Theorem 3.11, Theorem 3.16, the scheme of [GSW13] with a bounded error distribution for depth d_{fhe} (which relies on LWE with a modulus-to-noise ratio $2^{\mathsf{poly}(\lambda,d,\log N)}$, since $d_{\mathsf{fhe}} = \mathsf{poly}(\lambda, \log q)$), and Theorem 3.23 with $\ell_{\mathsf{hash}} = \lambda$, respectively. With bounded error distributions, the schemes of Theorem 3.11 and [GSW13] are perfectly correct, and their evaluation algorithms are deterministic.

Remark 5.5 (Trusted Setup.)**.** Note that unlike Construction 4.1, the public parameters of Construction 5.1 are not a uniformly random string, and the setup has to be trusted to erase $\mathsf{fhe.sk}$ and its random coins (including those of $\mathsf{CIH.StatGen}$, which may reveal $\langle \mathsf{Bad}_{\mathsf{fhe.sk}} \rangle$ and hence $\mathsf{fhe.sk}$), since $\mathsf{fhe.sk}$ decrypts the FHE encryptions of the secret keys that are part of the public keys.

5.2 Analysis

Correctness. Similar to Construction 4.1.

Compactness, and Succinctness. The analysis is similar to Construction 4.1 except that we can bound the partial decryption sizes as:

$$|p_i| = \ell \cdot \left| \left\lceil \widetilde{\mathsf{ct}}_{j,i}^{(r)} \cdot \mathsf{sk}_i^{(r)} \right\rfloor_2 \right| = \ell.$$

Semantic Security. We define the following hybrids:

- hyb_0**.** This is the original semantic security experiment where the adversary sends 1^d, and upon receiving $(\mathsf{PP}, \mathsf{PK})$ where $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$ and $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{KeyGen}(\mathsf{PP})$, sends (μ_0, μ_1) with $|\mu_0| = |\mu_1|$ to the challenger. Then the challenger chooses a bit $b \leftarrow \{0,1\}$ and sends $\mathsf{CT}_b \leftarrow \mathsf{Enc}(\mathsf{PK}, \mu_b)$ to $\mathcal{A}$. Finally $\mathcal{A}$ outputs b'. Then the experiment outputs 1 if $b' = b$.

- hyb_1**.** This is similar to hyb_0 except that the challenger during the setup samples the hash key as $\mathsf{hk} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$ instead of $\mathsf{hk} \leftarrow \mathsf{CIH.StatGen}(1^\lambda, \langle \mathsf{Bad}_{\mathsf{fhe.sk}} \rangle)$.

- hyb_2**.** This is similar to hyb_1 except that the challenger during the key generation computes $\mathsf{ct}_{\mathsf{fhe}}^{(t)} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^{|\mathsf{sk}^{(t)}|})$ for $t \in [0, \lambda - 1]$.

- $\mathsf{hyb}_{3,\tau}$ **for** $\tau \in [0, \lambda + 1]$**.** This is similar to hyb_2 except that the challenger during the key generation computes the encryptions of the top τ levels of the key chain as encryptions of zeros. Namely, it computes $\mathsf{ct}_{\mathsf{sk}}^{(t)} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}^{(t)}, 0^{|\mathsf{sk}^{(t-1)}|})$ for every $t \in [\lambda]$ with $t \geq \lambda + 1 - \tau$,

and if $\tau = \lambda + 1$, it additionally computes $\mathsf{ct}_{\mathsf{sk}}^{(0)} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}^{(0)}, 0^{|\mathsf{ote.st}_{\mathsf{sk}}|})$. Note that $\mathsf{hyb}_{3,0}$ is the same as hyb_2.

The indistinguishability of hybrids hyb_0 and hyb_1 follows by the mode indistinguishability of CIH. Next, in hyb_1 the secret key $\mathsf{fhe.sk}$ is not used anywhere, thus the indistinguishability of hyb_1 and hyb_2 follows by the semantic security of FHE. Then, similar to the standard proof of semantic security for bootstrapping, the indistinguishability of hyb_2 and $\mathsf{hyb}_{3,\lambda+1}$ follows by the semantic security of MKFHE', by switching the encryptions in the key chain to encryptions of zeros one level at a time, starting from the last level. Finally, in $\mathsf{hyb}_{3,\lambda+1}$, none of the $\mathsf{sk}^{(t)}$ and $\mathsf{ote.st}_{\mathsf{sk}}$ values are used in PP and PK, thus the semantic security follows from the semantic security of Construction 4.1. More formally let $\mathsf{hyb}_i(\mathcal{A})$ denote the output of the experiment in hybrid hyb_i when run on adversary $\mathcal{A}$.

Lemma 5.6. If CIH satisfies mode indistinguishability, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$|\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_0(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose towards the contradiction that $|\Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_0(\mathcal{A}) = 1]| \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We construct adversary $\mathcal{B}$ against the mode indistinguishability of CIH, where $\mathcal{B}$ chooses the auxiliary input (see the discussion after Definition 3.18), as follows:

1. Run adversary $\mathcal{A}$ to receive 1^d, sample $\mathsf{pp}, \mathsf{ote.crs}$ as in Setup, and sample $(\mathsf{fhe.pk}, \mathsf{fhe.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^{d_{\mathsf{fhe}}})$.
2. Send $\mathsf{aux} = \langle \mathsf{Bad}_{\mathsf{fhe.sk}} \rangle$ to the challenger, and receive hk from the challenger.
3. Let $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs}, \mathsf{fhe.pk}, \mathsf{hk})$, compute $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{KeyGen}(\mathsf{PP})$, and send $(\mathsf{PP}, \mathsf{PK})$ to $\mathcal{A}$.
4. Receive (μ_0, μ_1) from $\mathcal{A}$, sample $b \leftarrow \{0, 1\}$, and send $\mathsf{CT}_b \leftarrow \mathsf{Enc}(\mathsf{PK}, \mu_b)$ to $\mathcal{A}$.
5. Receive b' from $\mathcal{A}$, and output 1 if $b' = b$ and 0 otherwise.

Note that $\mathcal{B}$ perfectly simulates the challenger of hyb_0 (resp. hyb_1) if $\mathsf{hk} \leftarrow \mathsf{CIH.StatGen}(1^\lambda, \mathsf{aux})$ (resp. $\mathsf{hk} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$). Hence,

$$\left| \Pr\left[\mathcal{B} = 1 \mid \mathsf{hk} \leftarrow \mathsf{CIH.StatGen}(1^\lambda, \mathsf{aux})\right] - \Pr\left[\mathcal{B} = 1 \mid \mathsf{hk} \leftarrow \mathsf{CIH.Gen}(1^\lambda)\right] \right| \geq \epsilon(\lambda),$$

which contradicts the mode indistinguishability of CIH. □

Lemma 5.7. If FHE satisfies semantic security, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$|\Pr[\mathsf{hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_1(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. For $\tau \in [0, \lambda]$, let $\mathsf{hyb}_{1,\tau}$ be similar to hyb_1 except that the challenger computes $\mathsf{ct}_{\mathsf{fhe}}^{(t)} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^{|\mathsf{sk}^{(t)}|})$ for $t \in [0, \tau - 1]$. Note that $\mathsf{hyb}_{1,0}$ is the same as hyb_1 and $\mathsf{hyb}_{1,\lambda}$ is the same as hyb_2. Suppose towards the contradiction that $|\Pr[\mathsf{hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_1(\mathcal{A}) = 1]| \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. Then, by the triangle inequality, for every λ there exists $\tau \in [0, \lambda - 1]$ such that $|\Pr[\mathsf{hyb}_{1,\tau}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_{1,\tau+1}(\mathcal{A}) = 1]| \geq \epsilon(\lambda)/\lambda$, and we assume that $\Pr[\mathsf{hyb}_{1,\tau}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_{1,\tau+1}(\mathcal{A}) = 1] \geq \epsilon(\lambda)/\lambda$ (the other case follows similarly by flipping the output of $\mathcal{B}$ below). We construct adversary $\mathcal{B}$ against the semantic security of FHE, which has τ hardcoded in it, as follows:

1. Run adversary $\mathcal{A}$ to receive 1^d, and sample $\mathsf{pp}, \mathsf{ote.crs}$ as in Setup.

2. Send $1^{d_{\mathsf{fhe}}}$ to the challenger, receive $\mathsf{fhe.pk}$ from the challenger, sample $\mathsf{hk} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$, and let $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs}, \mathsf{fhe.pk}, \mathsf{hk})$.

3. Compute all the components of $(\mathsf{PK}, \mathsf{SK})$ as in $\mathsf{KeyGen}(\mathsf{PP})$, except that compute $\mathsf{ct}_{\mathsf{fhe}}^{(t)} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^{|\mathsf{sk}^{(t)}|})$ for $t \in [0, \tau - 1]$, send $(\mu_0^*, \mu_1^*) = (\mathsf{sk}^{(\tau)}, 0^{|\mathsf{sk}^{(\tau)}|})$ to the challenger, and let $\mathsf{ct}_{\mathsf{fhe}}^{(\tau)}$ be the received ciphertext.

4. Send $(\mathsf{PP}, \mathsf{PK})$ to $\mathcal{A}$, receive (μ_0, μ_1) from $\mathcal{A}$, sample $b \leftarrow \{0, 1\}$, and send $\mathsf{CT}_b \leftarrow \mathsf{Enc}(\mathsf{PK}, \mu_b)$ to $\mathcal{A}$.

5. Receive b' from $\mathcal{A}$, and output 0 if $b' = b$ and 1 otherwise.

Note that $\mathcal{B}$ does not need $\mathsf{fhe.sk}$, since in hyb_1 the hash key is sampled by $\mathsf{CIH.Gen}$ and $\mathsf{fhe.sk}$ is not used anywhere. Moreover, $\mathcal{B}$ perfectly simulates the challenger of $\mathsf{hyb}_{1,\tau}$ (resp. $\mathsf{hyb}_{1,\tau+1}$) if the challenge ciphertext encrypts μ_0^* (resp. μ_1^*). Hence, the probability that $\mathcal{B}$ guesses the challenge bit is

$$\frac{1}{2} + \frac{1}{2}\left(\Pr[\mathsf{hyb}_{1,\tau}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_{1,\tau+1}(\mathcal{A}) = 1]\right) \geq \frac{1}{2} + \frac{\epsilon(\lambda)}{2\lambda},$$

which contradicts the semantic security of FHE. □

Lemma 5.8. If MKFHE' satisfies semantic security, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\tau \in [0, \lambda]$:

$$|\Pr[\mathsf{hyb}_{3,\tau+1}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_{3,\tau}(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose towards the contradiction that $\Pr[\mathsf{hyb}_{3,\tau}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_{3,\tau+1}(\mathcal{A}) = 1] \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$ and some $\tau = \tau(\lambda) \in [0, \lambda]$ (the other case follows similarly by flipping the output of $\mathcal{B}$ below). We construct adversary $\mathcal{B}$ against the semantic security of MKFHE', which has τ hardcoded in it, as follows:

1. Run adversary $\mathcal{A}$ to receive 1^d, send $1^{d'}$ to the challenger (where d' is computed from d as in Setup), and let $t^* = \lambda - \tau$.

2. Receive $(\mathsf{pp}, \mathsf{pk}^*)$ from the challenger, sample $\mathsf{ote.crs}$ as in Setup, sample $(\mathsf{fhe.pk}, \mathsf{fhe.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^{d_{\mathsf{fhe}}})$ and $\mathsf{hk} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$, and let $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs}, \mathsf{fhe.pk}, \mathsf{hk})$.

3. Compute $(\mathsf{pk}, \mathsf{sk}, \mathsf{ote.}\hat{\mathsf{sk}}, \mathsf{ote.st}_{\mathsf{sk}})$ as in $\mathsf{KeyGen}(\mathsf{PP})$, let $\mathsf{pk}^{(t^*)} = \mathsf{pk}^*$, and sample $(\mathsf{pk}^{(t)}, \mathsf{sk}^{(t)}) \leftarrow \mathsf{MKFHE}'.\mathsf{KeyGen}(\mathsf{pp})$ for $t \in [0, \lambda] \setminus \{t^*\}$.

4. For $t \in [0, \lambda - 1]$ compute $\mathsf{ct}_{\mathsf{fhe}}^{(t)} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^{|\mathsf{sk}^{(t)}|})$. For $t \in [t^* + 1, \lambda]$ compute $\mathsf{ct}_{\mathsf{sk}}^{(t)} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}^{(t)}, 0^{|\mathsf{sk}^{(t-1)}|})$, and for $t \in [0, t^* - 1]$ compute $\mathsf{ct}_{\mathsf{sk}}^{(t)}$ as in KeyGen.

5. Send (μ_0^*, μ_1^*) to the challenger, where $\mu_0^* = \mathsf{sk}^{(t^*-1)}$ if $t^* \geq 1$ and $\mu_0^* = \mathsf{ote.st}_{\mathsf{sk}}$ if $t^* = 0$, and $\mu_1^* = 0^{|\mu_0^*|}$, and let $\mathsf{ct}_{\mathsf{sk}}^{(t^*)}$ be the received ciphertext.

6. Send $(\mathsf{PP}, \mathsf{PK})$ to $\mathcal{A}$, receive (μ_0, μ_1) from $\mathcal{A}$, sample $b \leftarrow \{0, 1\}$, and send $\mathsf{CT}_b \leftarrow \mathsf{Enc}(\mathsf{PK}, \mu_b)$ to $\mathcal{A}$.

7. Receive b' from $\mathcal{A}$, and output 0 if $b' = b$ and 1 otherwise.

Note that $\mathcal{B}$ does not need $\mathsf{sk}^{(t^*)}$ (other than its length), since in both $\mathsf{hyb}_{3,\tau}$ and $\mathsf{hyb}_{3,\tau+1}$ the values $\mathsf{ct}_{\mathsf{fhe}}^{(t^*)}$ and $\mathsf{ct}_{\mathsf{sk}}^{(t^*+1)}$ (if $t^* < \lambda$) encrypt zeros. Moreover, $\mathcal{B}$ perfectly simulates the challenger of $\mathsf{hyb}_{3,\tau}$ (resp. $\mathsf{hyb}_{3,\tau+1}$) if the challenge ciphertext encrypts μ_0^* (resp. μ_1^*). Hence, the probability that $\mathcal{B}$ guesses the challenge bit is

$$\frac{1}{2} + \frac{1}{2}\left(\Pr[\mathsf{hyb}_{3,\tau}(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_{3,\tau+1}(\mathcal{A}) = 1]\right) \geq \frac{1}{2} + \frac{\epsilon(\lambda)}{2},$$

which contradicts the semantic security of MKFHE'. □

Lemma 5.9. If Construction 4.1 satisfies semantic security, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr[\mathsf{hyb}_{3,\lambda+1}(\mathcal{A}) = 1] \leq 1/2 + \mathsf{negl}(\lambda).$$

Proof. Suppose towards the contradiction that $\Pr[\mathsf{hyb}_{3,\lambda+1}(\mathcal{A}) = 1] \geq 1/2 + \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We construct adversary $\mathcal{B}$ against the semantic security of Construction 4.1 as follows:

1. Run adversary $\mathcal{A}$ to receive 1^d, then send $1^{d'}$ to the challenger.
2. Receive $(\mathsf{PP}', \mathsf{PK}')$ from the challenger, where $\mathsf{PP}' = (\mathsf{pp}, \mathsf{ote.crs})$ and $\mathsf{PK}' = (\mathsf{pk}, \mathsf{ote.}\hat{\mathsf{sk}})$.
3. Sample $(\mathsf{fhe.pk}, \mathsf{fhe.sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda, 1^{d_{\mathsf{fhe}}})$ and $\mathsf{hk} \leftarrow \mathsf{CIH.Gen}(1^\lambda)$, and for $t \in [0, \lambda]$ sample $(\mathsf{pk}^{(t)}, \mathsf{sk}^{(t)}) \leftarrow \mathsf{MKFHE}'.\mathsf{KeyGen}(\mathsf{pp})$.
4. For $t \in [\lambda]$ compute $\mathsf{ct}_{\mathsf{sk}}^{(t)} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}^{(t)}, 0^{|\mathsf{sk}^{(t-1)}|})$, compute $\mathsf{ct}_{\mathsf{sk}}^{(0)} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}^{(0)}, 0^{|\mathsf{ote.st}_{\mathsf{sk}}|})$, and for $t \in [0, \lambda - 1]$ compute $\mathsf{ct}_{\mathsf{fhe}}^{(t)} \leftarrow \mathsf{FHE.Enc}(\mathsf{fhe.pk}, 0^{|\mathsf{sk}^{(t)}|})$.
5. Send $(\mathsf{PP}, \mathsf{PK})$ to $\mathcal{A}$, where $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs}, \mathsf{fhe.pk}, \mathsf{hk})$ and $\mathsf{PK} = (\mathsf{pk}, \mathsf{ote.}\hat{\mathsf{sk}}, (\mathsf{pk}^{(t)})_{t\in[0,\lambda]}, (\mathsf{ct}_{\mathsf{sk}}^{(t)})_{t\in[0,\lambda]}, (\mathsf{ct}_{\mathsf{fhe}}^{(t)})_{t\in[0,\lambda-1]})$.
6. Receive (μ_0, μ_1) from $\mathcal{A}$ and send (μ_0, μ_1) to the challenger.
7. Receive CT_b from the challenger and send it to $\mathcal{A}$.
8. Output whatever $\mathcal{A}$ outputs.

Note that the encryption algorithm of Construction 5.1 is the same as that of Construction 4.1, and $\mathcal{B}$ only uses the length of $\mathsf{ote.st}_{\mathsf{sk}}$, which is determined by PP'. Hence, $\mathcal{B}$ perfectly simulates the challenger of $\mathsf{hyb}_{3,\lambda+1}$ for $\mathcal{A}$, and it has the same advantage $\epsilon(\cdot)$ in breaking the semantic security of Construction 4.1, which contradicts Theorem 4.3 (see Lemmas 4.5 and 4.6). □

We conclude the proof by combining Lemmas 5.6 and 5.7, $\lambda + 1$ applications of Lemma 5.8 (for $\tau \in [0, \lambda]$, recall that $\mathsf{hyb}_{3,0}$ is the same as hyb_2), and Lemma 5.9.

The Shifted Values Are Far From the Rounding Boundaries. We use the following lemmas in the proofs of correctness and simulatability of partial decryption. For an integer $B \geq 0$, we say that $v \in \mathbb{Z}_q$ is *B-far* if $\big||v| - q/4\big| > B$, i.e., v is at distance more than B from the rounding boundaries $\pm q/4$ of $\lceil\cdot\rfloor_2$. Note that u is a bad shift for v if and only if $v - \theta \cdot u$ is not B_{eval}-far.

Lemma 5.10. Let $a \in \mathbb{Z}_q$ be B-far. Then, for any $M \in \{0,1\}$ and any $e \in \mathbb{Z}_q$ with $|e| \leq B$, it holds that

$$\lceil a + q/2 \cdot M + e\rfloor_2 = \lceil a\rfloor_2 + M \qquad \text{and} \qquad \lceil q/2 \cdot M + e - a\rfloor_2 = \lceil a\rfloor_2 + M.$$

Proof. For $x \in \mathbb{Z}_q$ with $|x| \neq q/4$, we have $\lceil x\rfloor_2 = 0$ if $|x| < q/4$ and $\lceil x\rfloor_2 = 1$ if $|x| > q/4$. Since $|-x| = |x|$ and $|x + q/2| = q/2 - |x|$, for any such x it holds that $\lceil -x\rfloor_2 = \lceil x\rfloor_2$ and $\lceil x + q/2 \cdot M\rfloor_2 = \lceil x\rfloor_2 + M$. Now since $\big||a \pm e| - |a|\big| \leq |e| \leq B < \big||a| - q/4\big|$, both $|a + e|$ and $|a - e|$ are on the same side of $q/4$ as $|a|$, hence $\lceil a + e\rfloor_2 = \lceil a - e\rfloor_2 = \lceil a\rfloor_2$. Therefore, $\lceil a+q/2\cdot M+e\rfloor_2 = \lceil a+e\rfloor_2+M = \lceil a\rfloor_2+M$ and $\lceil q/2\cdot M+e-a\rfloor_2 = \lceil -(a-e)\rfloor_2+M = \lceil a\rfloor_2+M$. □

Lemma 5.11. For any $v \in \mathbb{Z}_q$, there exists at most one bad shift for v.

Proof. Suppose towards the contradiction that $u \neq u'$ are bad shifts for v. Then each of $v - \theta u$ and $v - \theta u'$ is within distance B_{eval} of $q/4$ or $-q/4$, hence their difference $\theta(u' - u)$ satisfies $|\theta(u' - u)| \leq 2B_{\mathsf{eval}}$ or $|\theta(u' - u) - q/2| \leq 2B_{\mathsf{eval}}$. This is impossible, since $|\theta(u' - u)| = \theta|u' - u|$ and $2B_{\mathsf{eval}} < \theta \leq \theta|u' - u| \leq \theta(2^{\ell_{\mathsf{hash}}} - 1) < q/2 - 2B_{\mathsf{eval}}$, where the last inequality holds since $\theta > 2B_{\mathsf{eval}}$ and $q \geq 2^{\ell_{\mathsf{hash}}+1} \cdot \theta$. □

Lemma 5.12. For any PP in the support of $\mathsf{Setup}(1^\lambda, 1^d)$, any key pairs $(\mathsf{PK}_i, \mathsf{SK}_i)$ in the support of $\mathsf{KeyGen}(\mathsf{PP})$ for $i \in [N]$, any $\widetilde{\mathsf{CT}}$, and any $j \in [\ell]$, $t \in [r]$ and $k' \in [2^{t-1}]$, it holds that $-\delta^{(t)}_{j,k'} + \sum_{i \in S^{(t)}_{2k'-1}} y^{(t-1)}_{j,i}$ is B_{eval}-far, where $\delta^{(t)}_{j,k'}$ and $\widetilde{\mathsf{ct}}^{(t-1)}_{j,i}$ are computed as in $\mathsf{PartDec}$ and $y^{(t-1)}_{j,i} = \widetilde{\mathsf{ct}}^{(t-1)}_{j,i} \cdot \mathsf{sk}^{(t-1)}_i$.

Proof. Fix any random coins of Setup that output PP, let $\mathsf{fhe.sk}$ be the FHE secret key sampled with these coins, let $v = \sum_{i \in S^{(t)}_{2k'-1}} y^{(t-1)}_{j,i}$, and let $u = \mathsf{CIH.Hash}(\mathsf{hk}, \widehat{\mathsf{ct}}^{(t)}_{j,k'})$ (as an integer), so that $\delta^{(t)}_{j,k'} = \theta \cdot u$. Since each $\mathsf{ct}^{(t-1)}_{\mathsf{fhe},i}$ encrypts $\mathsf{sk}^{(t-1)}_i$ and $\widehat{C}^{(t)}_{j,k'}$ has depth at most d_{fhe}, by the perfect correctness of FHE we have $\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \widehat{\mathsf{ct}}^{(t)}_{j,k'}) = \widehat{C}^{(t)}_{j,k'}(\mathsf{sk}^{(t-1)}_{S^{(t)}_{2k'-1}}) = v$, i.e., $\mathsf{Bad}_{\mathsf{fhe.sk}}(\widehat{\mathsf{ct}}^{(t)}_{j,k'}) = \mathsf{Bad}(v)$. Moreover, since hk is in the support of $\mathsf{CIH.StatGen}(1^\lambda, \langle \mathsf{Bad}_{\mathsf{fhe.sk}} \rangle)$, by the somewhere perfect correlation intractability of CIH we have $u \neq \mathsf{Bad}_{\mathsf{fhe.sk}}(\widehat{\mathsf{ct}}^{(t)}_{j,k'}) = \mathsf{Bad}(v)$. By Lemma 5.11, v has at most one bad shift, and if it has one, then it is $\mathsf{Bad}(v)$, the smallest bad shift for v. Since $u \neq \mathsf{Bad}(v)$, u is not a bad shift for v, i.e., $v - \theta \cdot u$ is B_{eval}-far. □

Correctness of Partial Decryption. First note that by the construction, partial decryption correctness holds if for $j \in [\ell]$ it holds that $\Pr\left[\sum_{i \in [N]} p_{j,i} = C_j\left(\mu_{[N]}\right)\right] \geq 1 - \mathsf{negl}(\lambda)$. Moreover, since $\mathsf{MKFHE'.Eval}$, $\mathsf{FHE.Eval}$, and $\mathsf{CIH.Hash}$ are deterministic, and all the values computed in $\mathsf{PartDec}$ other than p_α only depend on public values, all the parties compute the same ciphertexts $\widetilde{\mathsf{CT}}^{(t)}_{j,k}$ and the same shifts $\delta^{(t)}_{j,k'}$, which we use in the following. In fact, our proof shows that the events in the following probabilities always hold, i.e., for any fixed choice of the values in the correctness experiment.

Recall from the construction that $S_k^{(t)} = [a_k^{(t)}, b_k^{(t)}]$, $a_k^{(t)} = (k-1)(N/2^t)+1$ and $b_k^{(t)} = k(N/2^t)$, and let $y_{j,i}^{(t)} = \widetilde{\mathsf{ct}}_{j,i}^{(t)} \cdot \mathsf{sk}_i^{(t)}$. To prove partial decryption correctness, we show that for $j \in [\ell]$ and $t \in [0, r]$ it holds that:

$$\Pr\left[\sum_{k\in[2^t]} \left\lfloor \sum_{i\in S_k^{(t)}} y_{j,i}^{(t)} \right\rceil_2 = C_j(\mu_{[N]})\right] \geq 1 - \mathsf{negl}(\lambda). \tag{3}$$

First note that if Eq. (3) holds for $t = r$ (where $2^r = N$ and $S_k^{(r)} = \{k\}$) we have that:

$$\begin{aligned}
\Pr\left[\sum_{i\in[N]} p_{j,i} = C_j\left(\mu_{[N]}\right)\right] &= \Pr\left[\sum_{i\in[N]} \left\lceil \widetilde{\mathsf{ct}}_{j,i}^{(r)} \cdot \mathsf{sk}_i^{(r)} \right\rfloor_2 = C_j\left(\mu_{[N]}\right)\right] \\
&= \Pr\left[\sum_{i\in[N]} \left\lceil y_{j,i}^{(r)} \right\rfloor_2 = C_j\left(\mu_{[N]}\right)\right] \\
&= \Pr\left[\sum_{k\in[2^r]} \left\lfloor \sum_{i\in S_k^{(r)}} y_{j,i}^{(r)} \right\rceil_2 = C_j(\mu_{[N]})\right] \\
&\geq 1 - \mathsf{negl}(\lambda).
\end{aligned}$$

Next we prove the correctness of Eq. (3) inductively. For the base case, i.e., when $t = 0$ and $k = 1$ (where $2^0 = 1$, $S_1^{(0)} = [N]$) we have:

$$\begin{aligned}
\left\lfloor \sum_{i\in[N]} y_{j,i}^{(0)} \right\rceil_2 &= \left\lfloor \sum_{i\in[N]} \widetilde{\mathsf{ct}}_{j,i}^{(0)} \cdot \mathsf{sk}_i^{(0)} \right\rceil_2 \\
&= \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{[N]}^{(0)}, \widetilde{\mathsf{CT}}_j^{(0)}\right) \\
&= \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{[N]}^{(0)}, \mathsf{MKFHE}'.\mathsf{Eval}\left(C_j^{(0)}, \left(\mathsf{pk}_i^{(0)}, \mathsf{ct}_{\mathsf{sk},i}^{(0)}\right)_{i\in[N]}\right)\right) \\
&= C_j^{(0)}(\mathsf{ote.st}_{\mathsf{sk},[N]}) \qquad \text{// by MKFHE}' \text{ correctness} \\
&= C_j(\mu_{[N]}) \qquad \text{// by the definition of } C_j^{(0)} \text{ and correctness of our construction.}
\end{aligned}$$

Next suppose the correctness of Eq. (3) holds for $t-1$; to prove that it holds for t as well we prove:

$$\Pr\left[\sum_{k\in[2^t]} \left\lfloor \sum_{i\in S_k^{(t)}} y_{j,i}^{(t)} \right\rceil_2 = \sum_{k\in[2^{t-1}]} \left\lfloor \sum_{i\in S_k^{(t-1)}} y_{j,i}^{(t-1)} \right\rceil_2\right] \geq 1 - \mathsf{negl}(\lambda).$$

The correctness of the above claim follows from the following:

$$\begin{aligned}
\sum_{k\in[2^t]}\left\lceil\sum_{i\in S_k^{(t)}} y_{j,i}^{(t)}\right\rceil_2 &= \sum_{k\in[2^t]}\left\lceil\sum_{i\in S_k^{(t)}} \widetilde{\mathsf{ct}}_{j,i}^{(t)}\cdot \mathsf{sk}_i^{(t)}\right\rceil_2 \\
&= \sum_{k\in[2^t]} \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{S_k^{(t)}}^{(t)}, \widetilde{\mathsf{CT}}_{j,k}^{(t)}\right) \\
&= \sum_{k\in[2^t]} \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{S_k^{(t)}}^{(t)}, \mathsf{MKFHE}'.\mathsf{Eval}\left(C_{j,k}^{(t)}, \left(\mathsf{pk}_i^{(t)}, \mathsf{ct}_{\mathsf{sk},i}^{(t)}\right)_{i\in S_k^{(t)}}\right)\right) \\
&= \sum_{k\in[2^t]} C_{j,k}^{(t)}\left(\mathsf{sk}_{S_k^{(t)}}^{(t-1)}\right) \\
&= \sum_{k\in[2^t]}\left\lceil(-1)^k\delta_{j,\lceil k/2\rceil}^{(t)} + \sum_{i\in S_k^{(t)}} \mathsf{sk}_i^{(t-1)}.\widetilde{\mathsf{ct}}_{j,i}^{(t-1)}\right\rceil_2 \\
&= \sum_{k'\in[2^{t-1}]}\left(\left\lceil -\delta_{j,k'}^{(t)} + \sum_{i\in S_{2k'-1}^{(t)}} y_{j,i}^{(t-1)}\right\rceil_2 \right. \\
&\qquad \left. + \left\lceil \delta_{j,k'}^{(t)} + \sum_{i\in S_{2k'}^{(t)}} y_{j,i}^{(t-1)}\right\rceil_2\right) \\
&= \sum_{k'\in[2^{t-1}]}\left(\left\lceil Z_{0,k'}\right\rfloor_2 + \left\lceil Z_{1,k'}\right\rfloor_2\right)
\end{aligned} \tag{4}$$

where we let $Z_{0,k'} = -\delta_{j,k'}^{(t)} + \sum_{i\in S_{2k'-1}^{(t)}} y_{j,i}^{(t-1)}$ and $Z_{1,k'} = \delta_{j,k'}^{(t)} + \sum_{i\in S_{2k'}^{(t)}} y_{j,i}^{(t-1)}$. Now note that the following hold: (i) $a_{2k'-1}^{(t)} = a_{k'}^{(t-1)}$, (ii) $b_{2k'-1}^{(t)} = (2k'-1)(N/2^t)$, (iii) $a_{2k'}^{(t)} = (2k'-1)(N/2^t)+1$, and (iv) $b_{2k'}^{(t)} = b_{k'}^{(t-1)}$; therefore $S_{2k'-1}^{(t)} \cup S_{2k'}^{(t)} = S_{k'}^{(t-1)}$ and for any $k' \in [2^{t-1}]$ we have:

$$\begin{aligned}
Z_{0,k'} + Z_{1,k'} &= \sum_{i\in S_{k'}^{(t-1)}} y_{j,i}^{(t-1)} \\
&= \sum_{i\in S_{k'}^{(t-1)}} \widetilde{\mathsf{ct}}_{j,i}^{(t-1)}\cdot \mathsf{sk}_i^{(t-1)} \\
&= q/2\cdot \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{S_{k'}^{(t-1)}}^{(t-1)}, \widetilde{\mathsf{CT}}_{j,k'}^{(t-1)}\right) + e'
\end{aligned} \tag{5}$$

where e' is the MKFHE' error, thus $|e'| \le B_{\mathsf{eval}}$. Now let $Z_{k'} = \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{S_{k'}^{(t-1)}}^{(t-1)}, \widetilde{\mathsf{CT}}_{j,k'}^{(t-1)}\right)$. By Lemma 5.12, $Z_{0,k'}$ is B_{eval}-far, and by Eq. (5) we have $Z_{1,k'} = q/2\cdot Z_{k'} + e' - Z_{0,k'}$, hence by Lemma 5.10 it holds that $\lceil Z_{1,k'}\rfloor_2 = \lceil Z_{0,k'}\rfloor_2 + Z_{k'}$, or equivalently:

$$\left\lceil Z_{0,k'}\right\rfloor_2 + \left\lceil Z_{1,k'}\right\rfloor_2 = Z_{k'}.$$

Moreover, by Eq. (5) and since $|e'| \leq B_{\mathsf{eval}} < q/4$, we have:

$$\left\lceil \sum_{i \in S_{k'}^{(t-1)}} y_{j,i}^{(t-1)} \right\rfloor_2 = \left\lceil q/2 \cdot Z_{k'} + e' \right\rfloor_2 = Z_{k'},$$

now since the above argument works for any $k' \in [2^{t-1}]$, by summing over $k' \in [2^{t-1}]$ we get:

$$\sum_{k' \in [2^{t-1}]} \left(\left\lceil Z_{0,k'} \right\rfloor_2 + \left\lceil Z_{1,k'} \right\rfloor_2 \right) = \sum_{k' \in [2^{t-1}]} \left\lceil \sum_{i \in S_{k'}^{(t-1)}} y_{j,i}^{(t-1)} \right\rfloor_2, \tag{6}$$

and we conclude the inductive proof of Eq. (3) by combining Eqs. (4) and (6).

Simulatability of Partial Decryption. For simplicity of the analysis we show the simulatability of p_1 and the simulatability of the other shares follows by the same approach because of the symmetry of the partial decryption in our construction (see the end of this paragraph). Let $y_{j,i}^{(t)} = \widetilde{\mathsf{ct}}_{j,i}^{(t)} \cdot \mathsf{sk}_i^{(t)}$, $f_j^{(t)} = \delta_{j,1}^{(t)}$, and $e_j^{(t)}$ be the MKFHE' error while evaluating circuit $C_{j,1}^{(t)}$. We construct the simulator $\mathcal{S}(\tilde{\mu}, \widetilde{\mathsf{CT}}, 1, \mathsf{SK}_{[2,N]})$ where $\tilde{\mu} = C(\mu_{[N]})$ as follows:

1. compute $(\widetilde{\mathsf{ct}}_{j,S_2^{(t)}}^{(t-1)})_{j \in [\ell], t \in [r]}$ and $(f_j^{(t)})_{j \in [\ell], t \in [r]}$ similar to the partial decryption algorithm,
2. output $p_1' = \left(\tilde{\mu}_j + \sum_{t \in [r]} \left\lceil -f_j^{(t)} - \sum_{i \in S_2^{(t)}} y_{j,i}^{(t-1)} \right\rfloor_2 \right)_{j \in [\ell]}$.

Next we claim that during the partial decryption for any $r' \in [3, r+1]$ it holds that:

$$\begin{aligned} &\left\lceil -f_j^{(r'-1)} + \sum_{i \in S_1^{(r'-1)}} y_{j,i}^{(r'-2)} \right\rfloor_2 \\ &\qquad = \left\lceil -f_j^{(r'-2)} + \sum_{i \in S_1^{(r'-2)}} y_{j,i}^{(r'-3)} \right\rfloor_2 + \left\lceil -f_j^{(r'-1)} - \sum_{i \in S_2^{(r'-1)}} y_{j,i}^{(r'-2)} \right\rfloor_2. \end{aligned} \tag{7}$$

To prove the above claim we rewrite the left hand side as follows:

$$
\begin{aligned}
LHS &= \left\lceil -f_j^{(r'-1)} + \sum_{i \in S_1^{(r'-1)}} y_{j,i}^{(r'-2)} \right\rfloor_2 \\
&= \left\lceil \sum_{i \in S_1^{(r'-2)}} y_{j,i}^{(r'-2)} - f_j^{(r'-1)} - \sum_{i \in S_2^{(r'-1)}} y_{j,i}^{(r'-2)} \right\rfloor_2 \\
&= \left\lceil q/2 \cdot \mathsf{MKFHE'.Dec}\left(\mathsf{sk}_{S_1^{(r'-2)}}^{(r'-2)}, \widetilde{\mathsf{CT}}_{j,1}^{(r'-2)}\right) + e_j^{(r'-2)} - f_j^{(r'-1)} - \sum_{i \in S_2^{(r'-1)}} y_{j,i}^{(r'-2)} \right\rfloor_2 \\
&= \mathsf{MKFHE'.Dec}\left(\mathsf{sk}_{[2^{r-r'+2}]}^{(r'-2)}, \widetilde{\mathsf{CT}}_{j,1}^{(r'-2)}\right) + \left\lceil -f_j^{(r'-1)} - \sum_{i \in S_2^{(r'-1)}} y_{j,i}^{(r'-2)} \right\rfloor_2
\end{aligned}
$$

where the last equality holds as follows. Let $M = \mathsf{MKFHE'.Dec}\left(\mathsf{sk}_{S_1^{(r'-2)}}^{(r'-2)}, \widetilde{\mathsf{CT}}_{j,1}^{(r'-2)}\right)$ and $a = -f_j^{(r'-1)} + \sum_{i \in S_1^{(r'-1)}} y_{j,i}^{(r'-2)}$, so that $LHS = \lceil a \rfloor_2$ and, by the above, $-f_j^{(r'-1)} - \sum_{i \in S_2^{(r'-1)}} y_{j,i}^{(r'-2)} = a + q/2 \cdot M - e_j^{(r'-2)}$ (note that $-q/2 \cdot M = q/2 \cdot M \bmod q$). Now, since $f_j^{(r'-1)} = \delta_{j,1}^{(r'-1)}$ and $S_1^{(r'-1)}$ is the first child of $S_1^{(r'-2)}$, by Lemma 5.12 a is B_{eval}-far, and since $|e_j^{(r'-2)}| \leq B_{\mathsf{eval}}$, by Lemma 5.10 we get

$$
\left\lceil -f_j^{(r'-1)} - \sum_{i \in S_2^{(r'-1)}} y_{j,i}^{(r'-2)} \right\rfloor_2 = \left\lceil a + q/2 \cdot M - e_j^{(r'-2)} \right\rfloor_2 = \lceil a \rfloor_2 + M = LHS + M,
$$

which implies the last equality since $-M = M \bmod 2$. Thus, we only need to prove

$$
\mathsf{MKFHE'.Dec}\left(\mathsf{sk}_{[2^{r-r'+2}]}^{(r'-2)}, \widetilde{\mathsf{CT}}_{j,1}^{(r'-2)}\right) = \left\lceil -f_j^{(r'-2)} + \sum_{i \in S_1^{(r'-2)}} y_{j,i}^{(r'-3)} \right\rfloor_2 .
$$

Now note that

$$
\widetilde{\mathsf{CT}}_{j,1}^{(r'-2)} = \mathsf{MKFHE'.Eval}\left(C_{j,1}^{(r'-2)}, \left(\mathsf{pk}_i^{(r'-2)}, \mathsf{ct}_{\mathsf{sk},i}^{(r'-2)}\right)_{i \in S_1^{(r'-2)}}\right)
$$

where $S_1^{(r'-2)} = [2^{r-r'+2}]$, $\mathsf{ct}_{\mathsf{sk},i}^{(r'-2)} \leftarrow \mathsf{MKFHE'.Enc}(\mathsf{pk}_i^{(r'-2)}, \mathsf{sk}_i^{(r'-3)})$ and

$$
C_{j,1}^{(r'-2)}(x_{S_1^{(r'-2)}}) \coloneqq \left\lceil -\delta_{j,1}^{(r'-2)} + \sum_{i \in S_1^{(r'-2)}} x_i . \widetilde{\mathsf{ct}}_{j,i}^{(r'-3)} \right\rfloor_2 .
$$

Putting them together we get that

$$\mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{[2^{r-r'+2}]}^{(r'-2)}, \widetilde{\mathsf{CT}}_{j,1}^{(r'-2)}\right) = \left\lceil -\delta_{j,1}^{(r'-2)} + \sum_{i\in S_1^{(r'-2)}} \mathsf{sk}_i^{(r'-3)}.\widetilde{\mathsf{ct}}_{j,i}^{(r'-3)} \right\rfloor_2$$
$$= \left\lceil -f_j^{(r'-2)} + \sum_{i\in S_1^{(r'-2)}} y_{j,i}^{(r'-3)} \right\rfloor_2$$

which concludes the proof of Eq. (7).

Next, we rewrite the partial decryption $p_{j,1}$ as follows:

$$\begin{aligned}
p_{j,1} &= \lceil \widetilde{\mathsf{ct}}_{j,1}^{(r)} \cdot \mathsf{sk}_1^{(r)} \rfloor_2 \\
&= \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_1^{(r)}, \widetilde{\mathsf{CT}}_{j,1}^{(r)}\right) \\
&= \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_1^{(r)}, \mathsf{MKFHE}'.\mathsf{Eval}\left(C_{j,1}^{(r)}, \left(\mathsf{pk}_1^{(r)}, \mathsf{ct}_{\mathsf{sk},1}^{(r)}\right)\right)\right) \\
&= C_{j,1}^{(r)}(\mathsf{sk}_1^{(r-1)}) \\
&= \left\lceil -f_j^{(r)} + \mathsf{sk}_1^{(r-1)} \cdot \widetilde{\mathsf{ct}}_{j,1}^{(r-1)} \right\rfloor_2 \\
&= \left\lceil -f_j^{(r)} + \sum_{i\in S_1^{(r)}} y_{j,i}^{(r-1)} \right\rfloor_2 && \text{// rewriting as LHS of Eq. (7) where } r' = r+1 \\
&= \left\lceil -f_j^{(1)} + \sum_{i\in S_1^{(1)}} y_{j,i}^{(0)} \right\rfloor_2 + \sum_{t\in[2,r]} \left\lceil -f_j^{(t)} - \sum_{i\in S_2^{(t)}} y_{j,i}^{(t-1)} \right\rfloor_2 \\
&&& \text{// by repeatedly applying Eq. (7) until } r' = 3 \\
&= \mathsf{MKFHE}'.\mathsf{Dec}\left(\mathsf{sk}_{[N]}^{(0)}, \widetilde{\mathsf{CT}}_j^{(0)}\right) + \sum_{t\in[r]} \left\lceil -f_j^{(t)} - \sum_{i\in S_2^{(t)}} y_{j,i}^{(t-1)} \right\rfloor_2 \\
&&& \text{// by similar analysis to the correctness of Eq. (7)} \\
&= \tilde{\mu}_j + \sum_{t\in[r]} \left\lceil -f_j^{(t)} - \sum_{i\in S_2^{(t)}} y_{j,i}^{(t-1)} \right\rfloor_2 \\
&&& \text{// by induction base case proof of correctness of Eq. (3).}
\end{aligned}$$

First note that the last equation above is where we crucially rely on the perfect correctness of the underlying MKFHE. Now, recall that the simulator outputs $p'_{j,1} = \tilde{\mu}_j + \sum_{t\in[r]} \left\lceil -f_j^{(t)} - \sum_{i\in S_2^{(t)}} y_{j,i}^{(t-1)} \right\rfloor_2$. Therefore, $p'_{j,1} = p_{j,1}$ for every $j \in [\ell]$, i.e., the simulated partial decryption p'_1 is identical to the

real partial decryption p_1 for any fixed choice of the values in the simulatability experiment (and in particular $p_1 \overset{stat}{\approx} p'_1$).

Finally, for a party $\alpha \in [N]$, let $S^{(t)}_{k_t}$ be the node at level $t \in [r]$ that contains α. The simulator for p_α is defined similarly, where for every $t \in [r]$ and $k' = \lceil k_t/2 \rceil$, if $k_t = 2k' - 1$ it adds $\lceil -\delta^{(t)}_{j,k'} - \sum_{i \in S^{(t)}_{2k'}} y^{(t-1)}_{j,i} \rfloor_2$ (as above), and if $k_t = 2k'$ it adds $\lceil -\delta^{(t)}_{j,k'} + \sum_{i \in S^{(t)}_{2k'-1}} y^{(t-1)}_{j,i} \rfloor_2$, both of which only depend on the secret keys of the parties in $[N] \setminus \{\alpha\}$. The analysis is the same as above (where the MKFHE' errors are now those of the ciphertexts $\widetilde{\mathsf{CT}}^{(t)}_{j,k_t}$), except that for the levels t where $k_t = 2k'$, the analogue of Eq. (7) follows directly from the second equality of Lemma 5.10, applied to the B_{eval}-far value $-\delta^{(t)}_{j,k'} + \sum_{i \in S^{(t)}_{2k'-1}} y^{(t-1)}_{j,i}$ (see Lemma 5.12).

6 Threshold Trapdoor Hashing

As discussed in the introduction and technical overview, in this work we consider a variant of trapdoor hash functions [DGI+19] in which the hashing and encoding procedures are merged into a single algorithm. In this section, we formalize a corresponding threshold notion (TTDH) for this merged interface and then show how to instantiate it from our fully-succinct MKFHE. Note that, due to our instantiation, the parameters depend on the depth d of a circuit that computes the predicates, hence, our definition of TTDH is parametrized by d.

6.1 Definition

Consider a class of predicates $\mathcal{F} = \{\mathcal{F}_{\ell,d}\}_{\ell,d \in \mathbb{N}}$, where $\mathcal{F}_{\ell,d} \coloneqq \{f : \{0,1\}^\ell \to \{0,1\}\}$ and for any $f \in \mathcal{F}_{\ell,d}$, f can be computed by a circuit of depth at most d.

Syntax. A threshold trapdoor hash $\mathsf{TTDH} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Hash\&Enc}, \mathsf{Decode})$ scheme for $\mathcal{F}$ consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d) \to \mathsf{hk}$**.** The probabilistic setup algorithm takes as input a security parameter 1^λ, an input length ℓ in unary, a maximum depth d in unary, and outputs a hash key hk.

$\mathsf{Gen}(\mathsf{hk}, f) \to (\mathsf{ek}, \mathsf{td})$**.** The probabilistic generation algorithm takes as input a hash key hk, and a function f, and outputs an encoding key ek, and a trapdoor td.

$\mathsf{Hash\&Enc}(\mathsf{hk}, (\mathsf{ek}_i)_{i \in [N]}, x) \to (\mathsf{d}, e)$**.** The probabilistic hash and encoding algorithm takes as input a hash key hk, a set of encoding keys $(\mathsf{ek}_i)_{i \in [N]}$, and an input x, and outputs an encoding e and a digest d.

$\mathsf{Decode}((\mathsf{td}_i)_{i \in [N]}, \mathsf{d}) \to (e_0, e_1)$**.** The decoding algorithm takes as input a set of trapdoors $(\mathsf{td}_i)_{i \in [N]}$, and a digest d, and outputs two encoding values e_0 and e_1.

Definition 6.1 (TTDH)**.** A threshold trapdoor hash $\mathsf{TTDH} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Hash\&Enc}, \mathsf{Decode})$ scheme is required to satisfy the following properties:

Correctness. For any polynomials $N = N(\lambda)$, $d = d(\lambda)$ and $\ell = \ell(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, any predicate $f \in \mathcal{F}_{\ell,d}$, and any input $x \in \{0,1\}^\ell$, it holds that:

$$\Pr\left[\begin{array}{l} e = e_{f(x)} \wedge e \neq e_{1-f(x)} \end{array} : \begin{array}{l} \mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^d) \\ (\mathsf{ek}_i, \mathsf{td}_i) \leftarrow \mathsf{Gen}(\mathsf{hk}, f) \text{ for } i \in [N] \\ (\mathsf{d}, e) \leftarrow \mathsf{Hash\&Enc}(\mathsf{hk}, (\mathsf{ek}_i)_{i\in[N]}, x) \\ (e_0, e_1) \leftarrow \mathsf{Decode}((\mathsf{td}_i)_{i\in[N]}, \mathsf{d}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Function Privacy. For any stateful PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds that:

$$\Pr\left[\begin{array}{l} b \leftarrow \mathcal{A}(\mathsf{ek}_b) \end{array} : \begin{array}{l} 1^\ell, 1^d \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^d) \\ (f_0, f_1) \leftarrow \mathcal{A}(\mathsf{hk}), b \leftarrow \{0,1\} \\ (\mathsf{ek}_b, \mathsf{td}_b) \leftarrow \mathsf{Gen}(\mathsf{hk}, f_b) \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda).$$

Threshold Input Privacy. For any stateful PPT adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds that:

$$\Pr\left[\begin{array}{l} b' = b \end{array} : \begin{array}{l} 1^\ell, 1^d \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^d), b \leftarrow \{0,1\} \\ (f, x_0, x_1, j) \leftarrow \mathcal{A}(\mathsf{hk}) \\ (\mathsf{ek}_i, \mathsf{td}_i) \leftarrow \mathsf{Gen}(\mathsf{hk}, f) \text{ for } i \in [N] \\ (\mathsf{d}_b, e_b) \leftarrow \mathsf{Hash\&Enc}(\mathsf{hk}, (\mathsf{ek}_i)_{i\in[N]}, x_b) \\ b' \leftarrow \mathcal{A}(\mathsf{d}_b, e_b, (\mathsf{ek}_i)_{i\in[N]}, (\mathsf{td}_i)_{i\in[N]\setminus\{j\}}) \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda).$$

Compactness. There exists a fixed polynomial $\mathsf{poly}(\cdot,\cdot,\cdot)$ s.t. for any polynomials $N = N(\lambda)$, $\ell = \ell(\lambda)$, $d = d(\lambda)$, all $\lambda \in \mathbb{N}$, all hk in the support of $\mathsf{Setup}(1^\lambda, 1^\ell, 1^d)$, all d-depth bounded functions $f \in \mathcal{F}_{\ell,d}$, all $(\mathsf{ek}_i, \mathsf{td}_i)$ in the support of $\mathsf{Gen}(\mathsf{hk}, f)$ for $i \in [N]$, all inputs $x \in \{0,1\}^\ell$, the digest d output by $\mathsf{Hash\&Enc}(\mathsf{hk}, (\mathsf{ek}_i)_{i\in[N]}, x)$ satisfies $|\mathsf{d}| \leq \mathsf{poly}(\lambda, d, \log N)$.

Definition 6.2 (Rate-1 TTDH)**.** The rate of a TTDH scheme is defined as $\dfrac{1}{\ell_{\mathsf{enc}}}$ where $\ell_{\mathsf{enc}} = |e|$. Thus, a TTDH is said to be rate-1 if $\ell_{\mathsf{enc}} = 1$.

6.2 Construction

Here we construct threshold trapdoor hash functions for N-out-of-N thresholds with $N \geq 2$. We present our construction generically in ℓ_{enc}. In the following construction by $\langle f \rangle$ we denote the description of a predicate f, and we define $C_{x,e,e'}$ to be a circuit that on input $(\langle f \rangle_i)_{i\in[N]}$ does the following:

- if $\langle f \rangle_k \neq \langle f \rangle_{k'}$ for some $k, k' \in [N]$, output $\perp$ (padded to be of size $2\ell_{\mathsf{enc}}$),
- output (e, e') if $f_1(x) = 0$ and (e', e) otherwise.

Moreover, we let $d' = d'(d, \log N)$ be the maximum depth of $C_{x,e,e'}$ where d is the maximum depth of a circuit computing any f_i.[16]

[16] Note that by Cook and Hoover [CH85] evaluating f_1 on x has depth $O(d)$, and checking that all the descriptions are equal has depth $O(\log N + \log |\langle f \rangle|)$, hence it holds that $d' = O(d + \log N + \log |\langle f \rangle|)$.

Construction 6.3. Let MKFHE be a fully-succinct multi-key fully homomorphic encryption with circuit privacy.

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d) \to \mathsf{hk}$**.** Let $\mathsf{mkfhe.pp} \leftarrow \mathsf{MKFHE.Setup}(1^\lambda, 1^{d'})$ and output $\mathsf{hk} = \mathsf{mkfhe.pp}$.

$\mathsf{Gen}(\mathsf{hk}, f) \to (\mathsf{ek}, \mathsf{td})$**.** Compute $(\mathsf{mkfhe.pk}, \mathsf{mkfhe.sk}) \leftarrow \mathsf{MKFHE.KeyGen}(\mathsf{mkfhe.pp})$ and compute $\mathsf{ct} \leftarrow \mathsf{MKFHE.Enc}(\mathsf{mkfhe.pk}, \langle f \rangle)$ and output $\mathsf{ek} = (\mathsf{mkfhe.pk}, \mathsf{ct})$, and $\mathsf{td} = \mathsf{mkfhe.sk}$.

$\mathsf{Hash\&Enc}(\mathsf{hk}, (\mathsf{ek}_i)_{i\in[N]}, x) \to (\mathsf{d}, e)$**.** Do the following:

1. for $i \in [N]$ parse $\mathsf{ek}_i = (\mathsf{mkfhe.pk}_i, \mathsf{ct}_i)$,
2. sample random encodings $e, e' \leftarrow \{0,1\}^{\ell_{\mathsf{enc}}}$ s.t. $e \neq e'$,
3. compute $\widetilde{\mathsf{ct}} \leftarrow \mathsf{MKFHE.Eval}(C_{x,e,e'}, (\mathsf{mkfhe.pk}_i, \mathsf{ct}_i)_{i\in[N]})$,

and output $(\mathsf{d} = \widetilde{\mathsf{ct}}, e)$.

$\mathsf{Decode}((\mathsf{td}_i)_{i\in[N]}, \mathsf{d}) \to (e_0, e_1)$**.** Do the following:

1. Parse $\mathsf{d} = \widetilde{\mathsf{ct}}$ and for $i \in [N]$, $\mathsf{td}_i = \mathsf{mkfhe.sk}_i$,
2. compute $z = \mathsf{MKFHE.Dec}((\mathsf{mkfhe.sk}_i)_{i\in[N]}, \widetilde{\mathsf{ct}})$, and let $z = (e_0, e_1)$,

and output (e_0, e_1).

Remark 6.4 (Rate-1.)**.** While our construction is presented generically in ℓ_{enc} (i.e., ℓ_{enc} can be any polynomial in λ), we can w.l.o.g. let $\ell_{\mathsf{enc}} = 1$, thus our construction is rate-1.

6.3 Analysis

Theorem 6.5. Assuming fully-succinct multi-key fully homomorphic encryption with circuit privacy, the Construction 6.3 is a rate-1 TTDH with silent setup.

Corollary 6.6. Assuming LWE with a sub-exponential modulus-to-noise ratio where $q/B_\chi = 2^{\mathsf{poly}(\lambda, d', \log N)}$, there exists a rate-1 TTDH with silent setup.

Remark 6.7 (Silent Setup.)**.** We note that the generation algorithm only depends on the hash key and the function, and does not take other parties' encoding keys/trapdoors as input. Thus, encoding keys can be published simultaneously and without interaction.

Remark 6.8 (Secret Encoding.)**.** Note that the encoding $e' = e_{1-f(x)}$ in our construction is chosen uniformly at random, therefore, when $\ell_{\mathsf{enc}} = \omega(\log \lambda)$, our construction satisfies the [reusable] secret encodings property that is required for some of the applications in [DGI+19][17].

[17] This property is mainly used in private laconic oblivious transfer construction in [DGI+19]. However, their construction relies on separate hashing and encoding algorithms. Thus, it is not clear to us how one can construct [threshold] private laconic OT using our TTDH. We leave this as an open problem in this work.

Correctness. If $f(x) = 0$, then with high probability it holds that:

$$\begin{aligned}(e_0, e_1) &= \mathsf{Decode}((\mathsf{td}_i)_{i\in[N]}, \mathsf{d}) \\ &= \mathsf{MKFHE.Dec}((\mathsf{mkfhe.sk}_i)_{i\in[N]}, \widetilde{\mathsf{ct}}) \\ &= \mathsf{MKFHE.Dec}((\mathsf{mkfhe.sk}_i)_{i\in[N]}, \mathsf{MKFHE.Eval}(C_{x,e,e'}, (\mathsf{mkfhe.pk}_i, \mathsf{ct}_i)_{i\in[N]})) \\ &= C_{x,e,e'}((\langle f\rangle_i)_{i\in[N]}) \\ &= (e, e').\end{aligned}$$

Similarly, if $f(x) = 1$ then with high probability it holds that $(e_0, e_1) = (e', e)$. Thus, with high probability, $e = e_{f(x)}$ and $e \neq e_{1-f(x)} = e'$.[18]

Function Privacy. This property follows from the semantic security of the underlying MKFHE in a straightforward manner.

Threshold Input Privacy. We define the following hybrids:

- hyb_0**.** This is the original input privacy experiment where the adversary sends $(1^\ell, 1^d)$, then upon receiving (hk) where $\mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^d)$, sends (f, x_0, x_1, j) to the challenger. Then the challenger first generates $(\mathsf{ek}_i, \mathsf{td}_i) \leftarrow \mathsf{Gen}(\mathsf{hk}, f)$ for $i \in [N]$, then chooses a bit $b \leftarrow \{0,1\}$ and computes $(\mathsf{d}_b, e_b) \leftarrow \mathsf{Hash\&Enc}(\mathsf{hk}, (\mathsf{ek}_i)_{i\in[N]}, x_b)$, and sends $(\mathsf{d}_b, e_b, (\mathsf{ek}_i)_{i\in[N]}, (\mathsf{td}_i)_{i\in[N]\setminus\{j\}})$ to $\mathcal{A}$. Finally $\mathcal{A}$ outputs b'.
- hyb_1**.** This is similar to hyb_0 except that the challenger chooses f' s.t. $\langle f\rangle' \neq \langle f\rangle$ and $|\langle f\rangle'| = |\langle f\rangle|$, and generates $(\mathsf{ek}_j, \mathsf{td}_j) \leftarrow \mathsf{Gen}(\mathsf{hk}, f')$.
- hyb_2**.** This is similar to hyb_1 except that the challenger chooses C' (independently from (x_0, x_1)) s.t. $|C'| = |C_{x,e,e'}|$, and C' on any input, outputs $\perp$ (padded to be of size $2\ell_{\mathsf{enc}}$) and uses C' instead of $C_{x,e,e'}$ in $\mathsf{Hash\&Enc}$ algorithm.

First, note that the indistinguishability of hyb_0 and hyb_1 follows from the semantic security of MKFHE as the adversary doesn't get to see $\mathsf{td}_j = \mathsf{mkfhe.sk}_j$. Next, by the description of circuit $C_{x,e,e'}$, it holds that $C_{x,e,e'}((\langle f\rangle)_{i\in[j-1]}, \langle f\rangle', (\langle f\rangle)_{i\in[j+1,N]}) = \perp$ since $\langle f\rangle' \neq \langle f\rangle$ and $N \geq 2$. Thus, the circuits $C_{x,e,e'}$ in hyb_1 and C' in hyb_2, have the same functionality on the inputs $((\langle f\rangle)_{i\in[j-1]}, \langle f\rangle', (\langle f\rangle)_{i\in[j+1,N]})$. Therefore, the indistinguishability of hyb_1 and hyb_2 follows from the circuit privacy of MKFHE. Finally, note that in hyb_2, nothing about the choice of b is being revealed in $(\mathsf{d}_b, e_b, (\mathsf{ek}_i)_{i\in[N]}, (\mathsf{td}_i)_{i\in[N]\setminus\{j\}})$ as all of them are being generated independent from (x_0, x_1).

Compactness. Recall from the construction that $\mathsf{d} = \widetilde{\mathsf{ct}}$, thus, the digest size grows with $\mathsf{poly}(\lambda, d, \log N, |C_{x,e,e'}(\langle f\rangle_1, \ldots, \langle f\rangle_N)|) = \mathsf{poly}(\lambda, d, \log N, \ell_{\mathsf{enc}})$. Therefore, by letting encodings be of size $\mathsf{poly}(\lambda)$, we get that $|\mathsf{d}| = \mathsf{poly}(\lambda, d, \log N)$.

[18] Note that Construction A.1 is perfectly correct when OTE and MKFHE' are, hence, instantiating the underlying MKFHE with Construction A.1 implies a perfectly correct TTDH.

Acknowledgements. We thank Yuval Ishai for graciously answering our questions about multiparty computation and its connections to MKFHE. We also thank the anonymous CRYPTO 2026 reviewers for their thoughtful feedback and suggestions, which helped improve the presentation of this work.

References

[ACG+24] Abtin Afshar, Jiaqi Cheng, Rishab Goyal, Aayush Yadav, and Saikumar Yadugiri. Encrypted ram delegation: Homomorphic nizks, malicious mpc with optimal communication complexity, and more. *Cryptology ePrint Archive*, 2024.

[AG26] Abtin Afshar and Rishab Goyal. Fully-succinct multi-key FHE and rate-1 simulatable threshold decryption from LWE. In Nadia Heninger and Mike Rosulek, editors, *CRYPTO 2026, Part II*, volume 16801 of *LNCS*, pages 591–623. Springer, Heidelberg, August 2026.

[AJLA+12a] 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 *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 483–501. Springer, 2012.

[AJLA+12b] 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 *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 483–501. Springer, 2012.

[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 *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 1875–1886, 2025.

[ARS24] Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 301–330. Springer, 2024.

[BB16] Marina Blanton and Fattaneh Bayatbabolghani. Efficient server-aided secure two-party function evaluation with applications to genomic computation. 2016.

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th annual IEEE symposium on foundations of computer science*, pages 517–526. IEEE, 2009.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 533–556. Springer, 2014.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Annual International Cryptology Conference*, pages 565–596. Springer, 2018.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *Theory of Cryptography Conference*, pages 645–677. Springer, 2017.

[BJSS25] Elette Boyle, Abhishek Jain, Sacha Servan-Schreiber, and Akshayaram Srinivasan. Simultaneous-message and succinct secure computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 207–239. Springer, 2025.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. Nizk from lpn and trapdoor hash via correlation intractability for approximable relations. In *Annual International Cryptology Conference*, pages 738–767. Springer, 2020.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2019.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 575–584, 2013.

[BOCG+06] Michael Ben-Or, Claude Crépeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 249–260. IEEE, 2006.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual international cryptology conference*, pages 190–213. Springer, 2016.

[BPTG14] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. *Cryptology ePrint Archive*, 2014.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

[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 *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 395–412, 2019.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[CGS02] Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multi-party quantum computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 643–652, 2002.

[CH85] Stephen A Cook and H James Hoover. A depth-universal circuit. *SIAM Journal on Computing*, 14(4):833–839, 1985.

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled fhe from learning with errors. In *Annual Cryptology Conference*, pages 630–656. Springer, 2015.

[CO17] Wutichai Chongchitmate and Rafail Ostrovsky. Circuit-private multi-key fhe. In *IACR International Workshop on Public Key Cryptography*, pages 241–270. Springer, 2017.

[CZW17] Long Chen, Zhenfeng Zhang, and Xueqing Wang. Batched multi-hop multi-key fhe from ring-lwe with compact ciphertext extension. In *Theory of Cryptography Conference*, pages 597–627. Springer, 2017.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2019.

[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.

[DGMR22] Nico Döttling, Vipul Goyal, Giulio Malavolta, and Justin Raizes. Interaction-preserving compilers for secure computation. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, 2022.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122. Springer, Heidelberg, August 2016.

[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 *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476, 2013.

[GHO20] Sanjam Garg, Mohammad Hajiabadi, and Rafail Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 ot and more. In *Theory of Cryptography Conference*, pages 88–116. Springer, 2020.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[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 *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Annual Cryptology Conference*, pages 503–523. Springer, 2015.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Annual Cryptology Conference*, pages 132–150. Springer, 2011.

[HS14] Shai Halevi and Victor Shoup. Algorithms in helib. In *Annual Cryptology Conference*, pages 554–571. Springer, 2014.

[HS21] Shai Halevi and Victor Shoup. Bootstrapping for helib. *Journal of Cryptology*, 34(1):7, 2021.

[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 *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 726–740, 2023.

[KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. *Cryptology ePrint Archive*, 2011.

[KMR12] Seny Kamara, Payman Mohassel, and Ben Riva. Salus: a system for server-aided secure function evaluation. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 797–808, 2012.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

[MOR16] Payman Mohassel, Ostap Orobets, and Ben Riva. Efficient server-aided 2pc for mobile phones. *Proceedings on Privacy Enhancing Technologies*, 2016.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 735–763. Springer, 2016.

[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 *Theory of cryptography conference*, pages 217–238. Springer, 2016.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 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.

[Wic15] Daniel Wichs. Fully homomorphic encryption and signatures. Video, Simons Institute Workshop: The Mathematics of Modern Cryptography, July 2015. Co-presenter: Zvika Brakerski. Calvin Lab Auditorium, July 6, 2015.

A Fully-Succinct MKFHE With Circuit Privacy

In this section we construct a fully-succinct MKFHE with circuit privacy from a perfectly correct succinct non-interactive oblivious tensor evaluation OTE, and any perfectly correct multi-key FHE with split and near-linear decryption. In fact we show how to apply simple modifications to Construction 4.1 to achieve circuit privacy for free. The same approach can be used to make any of our other constructions (Construction 4.7 and Construction 5.1) circuit private as well. Before we present our circuit private construction, we formally define universal circuits below.

Universal Circuits. A circuit family $\{U\}_{s,d}$ is universal for circuits of size at most s and depth at most d if for every such circuit C and every input x, $U_{s,d}(x, \langle C\rangle) = C(x)$, where $\langle C\rangle$ is an encoding/description of circuit C. Note that once the circuit size and depth are fixed, the universal circuit $U_{s,d}$ is also fixed; hence, for simplicity of the notation and when it is clear from the context, we use $U(\cdot,\cdot)$ to denote a universal circuit. Moreover, we may slightly abuse the notation and, for a circuit C with n inputs $(x_1,\ldots,x_n)$, denote a universal circuit for computing $C(x_1,\ldots,x_n)$ by $U(x_1,\ldots,x_n,\langle C\rangle)$. Using the universal circuit of Cook and Hoover [CH85], it follows that $U(\cdot,\cdot)$ has depth $O(d)$.

A.1 Construction

Construction A.1. Let OTE be a perfectly correct succinct non-interactive oblivious tensor evaluation and MKFHE' be a perfectly correct multi-key fully homomorphic encryption with split and near-linear decryption.

$\mathsf{Setup}(1^\lambda, 1^d) \to \mathsf{PP}$**.** Same as in Construction 4.1, except that pp supports depth d', the depth of the universal circuit U (see parameter settings).

$\mathsf{KeyGen}(\mathsf{PP}) \to (\mathsf{PK}, \mathsf{SK})$**.** Same as in Construction 4.1.

$\mathsf{Enc}(\mathsf{PK}, \mu) \to \mathsf{CT}$**.** Same as in Construction 4.1.

$\mathsf{Eval}(C, (\mathsf{PK}_i, \mathsf{CT}_i)_{i\in[N]}) \to \widetilde{\mathsf{CT}}$**.** Do the following:

1. for $i \in [N]$, parse $\mathsf{PK}_i = (\mathsf{pk}_i, \mathsf{ote.}\hat{\mathsf{sk}}_i)$, and $\mathsf{CT}_i = \mathsf{ct}_i$,
2. let $\langle C_j\rangle$ be the description of a circuit C_j that computes the j^{th} output bit of C, and let U be a universal circuit for all $C_j(\mu_1,\ldots,\mu_N)$ for $j \in [\ell]$, i.e., $U(\mu_1,\ldots,\mu_N,\langle C_j\rangle) = C_j(\mu_1,\ldots,\mu_N)$ for $j \in [\ell]$,
3. sample $(\mathsf{pk}_{N+1}, \mathsf{sk}_{N+1}) \leftarrow \mathsf{MKFHE}'.\mathsf{KeyGen}(\mathsf{pp})$,
4. for $j \in [\ell]$ compute $\mathsf{ct}_{C,j} \leftarrow \mathsf{MKFHE}'.\mathsf{Enc}(\mathsf{pk}_{N+1}, \langle C_j\rangle)$,
5. for $j \in [\ell]$ compute $\widetilde{\mathsf{ct}}_j = \mathsf{MKFHE}'.\mathsf{Eval}\left(U, \left((\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}, (\mathsf{pk}_{N+1}, \mathsf{ct}_{C,j})\right)\right)$,
6. for $j \in [\ell]$ compute a partial decryption $\mathsf{pd}_j = \langle \mathsf{sk}_{N+1}, \widetilde{\mathsf{ct}}_j[nN+1, n(N+1)]\rangle$,
7. let $\widetilde{\mathsf{pct}} = (\widetilde{\mathsf{ct}}_1[1,nN] \| \ldots \| \widetilde{\mathsf{ct}}_\ell[1,nN])$ be a partial ciphertext,
8. compute $(\mathsf{ote.d}, \mathsf{ote.st}_{\widetilde{\mathsf{pct}}}) \leftarrow \mathsf{OTE.Hash}(\mathsf{ote.crs}, \widetilde{\mathsf{pct}})$,
9. for $i \in [N]$, compute $\mathbf{v}_i \leftarrow \mathsf{OTE.HashEval}(\mathsf{ote.crs}, \mathsf{ote.}\hat{\mathsf{sk}}_i, \mathsf{ote.st}_{\widetilde{\mathsf{pct}}})$,

10. for $j \in [\ell]$, sample $e_{\mathsf{sm},j} \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ (see parameter settings),
11. for $j \in [\ell]$, compute $v_j^* = p/q \cdot \mathsf{pd}_j + \sum_{i=1}^{N} \sum_{k=1}^{n} \mathbf{v}_i \left[h_{N,n}(j,i,k)\right] + e_{\mathsf{sm},j}$,

and output $\widetilde{\mathsf{CT}} = ((v_j^*)_{j\in[\ell]}, \mathsf{ote.d})$.

$\mathsf{Dec}((\mathsf{SK}_i)_{i\in[N]}, \widetilde{\mathsf{CT}}) \to \mu$. Same as in Construction 4.1.

$\mathsf{PartDec}(\widetilde{\mathsf{CT}}, (\mathsf{PK}_1, \ldots, \mathsf{PK}_N), i, \mathsf{SK}_i) \to p_i$. Same as in Construction 4.1.

$\mathsf{FinDec}(p_1, \ldots, p_N) \to \mu$. Same as in Construction 4.1.

Parameter Settings. First, in the parameter setting analysis of Construction 4.1 where $|e| \leq (mN+1)^d \cdot \mathsf{poly}(\lambda) \cdot B_\chi$ and $q \geq 4\left((mN+1)^d \cdot B_\chi \cdot \mathsf{poly}(\lambda) + Nn/2\right)$, we need to replace d with d' (and N with $N+1$ in the bound on $|e|$, as U is evaluated on $N+1$ keys), i.e., $q > B_\chi \cdot 2^{\mathsf{poly}(\lambda,d,\log N)}$ for some $\mathsf{poly}(\cdot,\cdot,\cdot)$, subject to $q \geq 4\left((m(N+1)+1)^{d'} \cdot B_\chi \cdot \mathsf{poly}(\lambda) + Nn/2\right)$, where $d' = O(d)$ is determined by the depth of the universal circuit [CH85].

Moreover, for the security of our construction we require $B_{\mathsf{smdg}}/|e''| > \lambda^{\omega(1)}$ and for the correctness, we require $p > 4(|e''| + B_{\mathsf{smdg}})$ where $|e''| \leq p/q \cdot (|e| + Nn/2)$. By letting $B_{\mathsf{smdg}} = 2^\lambda \cdot p/q \cdot ((m(N+1)+1)^{d'} \cdot \mathsf{poly}(\lambda) \cdot B_\chi + Nn/2) \geq 2^\lambda \cdot |e''|$, and letting $q > B_\chi \cdot 2^{\mathsf{poly}(\lambda,d,\log N)}$ for some appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$, subject to $q \geq (2^\lambda+1) \cdot \left(4\left((m(N+1)+1)^{d'} \cdot \mathsf{poly}(\lambda) \cdot B_\chi\right) + 2Nn\right) \geq (2^\lambda+1) \cdot (4|e| + 2Nn)$ the correctness of our construction follows. Similarly, we get $p > B'_\chi \cdot 2^{\mathsf{poly}(\lambda,d,\log N)}$ for an appropriate $\mathsf{poly}(\cdot,\cdot,\cdot)$.

A.2 Analysis

Theorem A.2. Assuming perfectly correct succinct non-interactive oblivious tensor evaluation, and perfectly correct multi-key fully homomorphic encryptions with split and near-linear decryption, the Construction A.1 is a fully-succinct MKFHE with circuit privacy where the evaluated ciphertext size is $\mathsf{poly}(\lambda, d, \log N, \ell)$.

Corollary A.3. Assuming LWE with a sub-exponential modulus-to-noise ratio where $q/B_\chi = 2^{\mathsf{poly}(\lambda,d,\log N)}$, the Construction A.1 is a fully-succinct MKFHE with circuit privacy where the evaluated ciphertext size is $\mathsf{poly}(\lambda, d, \log N, \ell)$.

Correctness. For $j \in [\ell]$ we have:

$$
\begin{aligned}
v_j^* + w_j^* &= p/q \cdot \mathsf{pd}_j + \sum_{i=1}^{N}\sum_{k=1}^{n} \mathbf{v}_i\left[h_{N,n}(j,i,k)\right] + e_{\mathsf{sm},j} + \sum_{i=1}^{N}\sum_{k=1}^{n} \mathbf{w}_i\left[h_{N,n}(j,i,k)\right] \\
&= p/q \cdot \mathsf{pd}_j + \sum_{i=1}^{N}\sum_{k=1}^{n} \left(\mathbf{v}_i\left[h_{N,n}(j,i,k)\right] + \mathbf{w}_i\left[h_{N,n}(j,i,k)\right]\right) + e_{\mathsf{sm},j} \\
&= p/q \cdot \mathsf{pd}_j + \sum_{i=1}^{N}\sum_{k=1}^{n} p/q \cdot \lceil \mathbf{v}_i\left[h_{N,n}(j,i,k)\right] + \mathbf{w}_i\left[h_{N,n}(j,i,k)\right] \rfloor_q + e' + e_{\mathsf{sm},j} \\
&\qquad \text{// where } |e'| \le Nn \cdot p/(2q) \text{ by the definition of rounding} \\
&= p/q \cdot \left(\mathsf{pd}_j + \sum_{i=1}^{N}\sum_{k=1}^{n} (\widetilde{\mathsf{pct}} \otimes \mathsf{sk}_i)\left[h_{N,n}(j,i,k)\right]\right) + e' + e_{\mathsf{sm},j} \qquad \text{// by OTE correctness} \\
&= p/q \cdot \left(\mathsf{pd}_j + \sum_{i=1}^{N}\sum_{k=1}^{n} \widetilde{\mathsf{pct}}[g_{N,n}(j,i,k)] \cdot \mathsf{sk}_i[k]\right) + e' + e_{\mathsf{sm},j} \qquad \text{// by Eq. (2)} \\
&= p/q \cdot \left(\sum_{k=1}^{n} \widetilde{\mathsf{ct}}_j[nN+k] \cdot \mathsf{sk}_{N+1}[k] + \sum_{i=1}^{N}\sum_{k=1}^{n} \widetilde{\mathsf{pct}}[g_{N,n}(j,i,k)] \cdot \mathsf{sk}_i[k]\right) + e' + e_{\mathsf{sm},j} \\
&= p/q \cdot \sum_{i=1}^{N+1}\sum_{k=1}^{n} \widetilde{\mathsf{ct}}[g_{N+1,n}(j,i,k)] \cdot \mathsf{sk}_i[k] + e' + e_{\mathsf{sm},j} \\
&= p/q \cdot \left(q/2 \cdot \mathsf{MKFHE}'.\mathsf{Dec}((\mathsf{sk}_i)_{i\in[N+1]}, \widetilde{\mathsf{ct}}_j) + e\right) + e' + e_{\mathsf{sm},j} \qquad \text{// by Eq. (1)} \\
&= p/2 \cdot \mathsf{MKFHE}'.\mathsf{Dec}((\mathsf{sk}_i)_{i\in[N+1]}, \widetilde{\mathsf{ct}}_j) + e'' + e_{\mathsf{sm},j} \qquad \text{// where } e'' = p/q \cdot e + e' \\
&= p/2 \cdot \mathsf{MKFHE}'.\mathsf{Dec}\left((\mathsf{sk}_i)_{i\in[N+1]}, \mathsf{MKFHE}'.\mathsf{Eval}\left(U, \left((\mathsf{pk}_i, \mathsf{ct}_i)_{i\in[N]}, (\mathsf{pk}_{N+1}, \mathsf{ct}_{C,j})\right)\right)\right) + e'' + e_{\mathsf{sm},j} \\
&= p/2 \cdot U((\mu_i)_{i\in[N]}, \langle C_j \rangle) + e'' + e_{\mathsf{sm},j} \qquad \text{// by } \mathsf{MKFHE}' \text{ correctness} \\
&= p/2 \cdot C_j((\mu_i)_{i\in[N]}) + e'' + e_{\mathsf{sm},j} \qquad \text{// by universal circuit correctness.}
\end{aligned}
$$

where $|e''| \le p/q \cdot (|e| + Nn/2)$, and $|e_{\mathsf{sm},j}| \le 2^\lambda \cdot |e''|$ and by our parameter setting we have $q > (2^\lambda + 1) \cdot (4|e| + 2nN)$, thus, we get $|e'' + e_{\mathsf{sm},j}| < p/4$, which implies that:

$$
\lceil v_j^* + w_j^* \rfloor_2 = \lceil p/2 \cdot C_j((\mu_i)_{i\in[N]}) + e'' + e_{\mathsf{sm},j} \rfloor_2 = C_j((\mu_i)_{i\in[N]}) + \lceil e'' + e_{\mathsf{sm},j} \rfloor_2 = C_j((\mu_i)_{i\in[N]}).
$$

Correctness of Partial Decryption. Same as in Construction 4.1.

Compactness, and Succinctness. Same as in Construction 4.1.

Semantic Security. Same as in Construction 4.1.

Circuit Privacy. Let $\tilde{\mu} = C_0((\mu_i)_{i\in[N]}) = C_1((\mu_i)_{i\in[N]})$. We define the following hybrids:

hyb_0**.** This is the original circuit privacy experiment where $b = 0$. Namely, the adversary $\mathcal{A}$ sends 1^d, and upon receiving $(\mathsf{PP}, (\mathsf{PK}_i, \mathsf{SK}_i)_{i\in[N]})$ where $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$ and $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow$

$\mathsf{KeyGen}(\mathsf{PP})$ for $i \in [N]$, sends $((\mu_i)_{i\in[N]}, C_0, C_1)$ to the challenger s.t. $C_0((\mu_i)_{i\in[N]}) = C_1((\mu_i)_{i\in[N]})$. Then, the challenger computes $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{PK}_i, \mu_i)$ for $i \in [N]$, and $\widetilde{\mathsf{CT}}_0 \leftarrow \mathsf{Eval}(C_0, (\mathsf{PK}_i, \mathsf{CT}_i)_{i\in[N]})$, and sends $(\widetilde{\mathsf{CT}}_0, (\mathsf{CT}_i)_{i\in[N]})$ to $\mathcal{A}$. Finally $\mathcal{A}$ outputs b', and the experiment outputs b'.

hyb_1**.** Similar to hyb_0 except that the challenger computes $\mathsf{ote.d}$ (as part of $\widetilde{\mathsf{CT}}$) similar to the evaluation algorithm, however, for $j \in [\ell]$, instead of using sk_{N+1} to compute v_j^* (as part of $\widetilde{\mathsf{CT}}$) as does the evaluation algorithm, it computes w_j^* using $\mathsf{ote.d}$ and $(\mathsf{SK}_i)_{i\in[N]}$ similar to the decrypting algorithm, then samples $e'_{\mathsf{sm},j} \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ and computes $v_j^* = p/2 \cdot \tilde{\mu}_j - w_j^* + e'_{\mathsf{sm},j}$.

hyb_2**.** Similar to hyb_1 except that the challenger replaces C_0 with C_1 in the evaluation algorithm.

hyb_3**.** Similar to hyb_2 except that it reverts the changes in hyb_1, i.e., it computes v_j^* for $j \in [\ell]$, using sk_{N+1} (similar to the evaluation algorithm). Note that this is the original circuit privacy experiment where $b = 1$.

Now note that the indistinguishability of hyb_0 and hyb_1 and the indistinguishability of hyb_2 and hyb_3 follow by the correctness of the scheme and smudging lemma (see Lemma 3.2). Moreover, since in hybrids hyb_1 and hyb_2 the challenger does not use sk_{N+1}, the indistinguishability of these hybrids follows from the semantic security of MKFHE' and the fact that $C_0((\mu_i)_{i\in[N]}) = C_1((\mu_i)_{i\in[N]})$. More formally, let $\mathsf{hyb}_i(\mathcal{A})$ denote the output of the experiment in hybrid hyb_i when run on adversary $\mathcal{A}$.

Lemma A.4. If OTE and MKFHE' are perfectly correct, then $\mathsf{hyb}_0 \overset{stat}{\approx} \mathsf{hyb}_1$.

Proof. By OTE and MKFHE' perfect correctness, using the argument in correctness of Construction A.1 it holds that $v_j^* + w_j^* = p/2 \cdot \tilde{\mu}_j + e'' + e_{\mathsf{sm},j}$. Moreover, using Lemma 3.2 and by our parameter settings we have $e'' + e_{\mathsf{sm},j} \overset{stat}{\approx} e'_{\mathsf{sm},j}$, thus, $v_j^* = p/2 \cdot \tilde{\mu}_j - w_j^* + e'' + e_{\mathsf{sm},j}$ as computed in hyb_0 and $v_j^* = p/2 \cdot \tilde{\mu}_j - w_j^* + e'_{\mathsf{sm},j}$ as computed in hyb_1 are statistically indistinguishable. Note that the only difference between hyb_0 and hyb_1 is in how they compute v_j^*, therefore $\mathsf{hyb}_0 \overset{stat}{\approx} \mathsf{hyb}_1$. □

Lemma A.5. If MKFHE' satisfies semantic security, then there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$|\Pr[\mathsf{hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_1(\mathcal{A}) = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Suppose towards the contradiction that $\Pr[\mathsf{hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{hyb}_1(\mathcal{A}) = 1] \geq \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We construct adversary $\mathcal{B}$ against the semantic security of MKFHE' as follows:

1. Run adversary $\mathcal{A}$ to receive 1^d, and send $1^{d'}$ to the challenger.
2. Receive $(\mathsf{pp}, \mathsf{pk})$ from the challenger, and sample $\mathsf{ote.crs} \leftarrow \mathsf{OTE.Setup}(1^\lambda)$.
3. Let $\mathsf{PP} = (\mathsf{pp}, \mathsf{ote.crs})$ and for $i \in [N]$ sample $(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{KeyGen}(\mathsf{PP})$.
4. Send $(\mathsf{PP}, (\mathsf{PK}_i, \mathsf{SK}_i)_{i\in[N]})$ to $\mathcal{A}$ and receive $((\mu_i)_{i\in[N]}, C_0, C_1)$ from $\mathcal{A}$.
5. Let $\tilde{\mu} = C_0((\mu_i)_{i\in[N]})$, and for $i \in [N]$ compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{PK}_i, \mu_i)$.

6. Send $(\langle C_0 \rangle, \langle C_1 \rangle)$ to the challenger and receive ct_C from the challenger.
7. Parse $\mathsf{ct}_C = (\mathsf{ct}_{C,1}, \ldots, \mathsf{ct}_{C,\ell})$, and let $\mathsf{pk}_{N+1} = \mathsf{pk}$.
8. Compute $\mathsf{ote.d}$ similar to the Eval algorithm (Note that sk_{N+1} is not used here).
9. For $j \in [\ell]$ compute w_j^* using $\mathsf{ote.d}$ and $(\mathsf{SK}_i)_{i\in[N]}$ similar to the decrypting algorithm, then sample $e'_{\mathsf{sm},j} \leftarrow [-B_{\mathsf{smdg}}, B_{\mathsf{smdg}}]$ and compute $v_j^* = p/2 \cdot \tilde{\mu}_j - w_j^* + e'_{\mathsf{sm},j}$.
10. Let $\widetilde{\mathsf{CT}} = ((v_j^*)_{j\in[\ell]}, \mathsf{ote.d})$ and send it to $\mathcal{A}$.
11. Output whatever $\mathcal{A}$ outputs.

Note that $\mathcal{B}$ perfectly simulates the challenger of hyb_1 (resp. hyb_2) if $\langle C_0 \rangle$ (resp. $\langle C_1 \rangle$) is encrypted by the challenger. Hence, it has the same advantage $\epsilon(\cdot)$ in breaking the semantic security of the underlying MKFHE' that contradicts the security of MKFHE'. □

Lemma A.6. If OTE and MKFHE' are perfectly correct, then $\mathsf{hyb}_2 \overset{stat}{\approx} \mathsf{hyb}_3$.

Proof. Similar to the proof of Lemma A.4. □

We conclude the proof by combining Lemmas A.4 to A.6.