Somewhat Homomorphic Encryption from Linear Homomorphism and Sparse LPN

Henry Corrigan-Gibbs
MIT CSAIL
henrycg@csail.mit.edu

Alexandra Henzinger
MIT CSAIL
ahenz@csail.mit.edu

Yael Tauman Kalai
MIT CSAIL
tauman@mit.edu

Vinod Vaikuntanathan
MIT CSAIL
vinodv@csail.mit.edu

May 24, 2025

Abstract

We construct a somewhat-homomorphic encryption scheme from the sparse learning-parities-with-noise problem, along with any assumption that implies linearly homomorphic encryption (e.g., the decisional Diffie-Hellman or decisional composite residuosity assumption). Our resulting schemes support an a-priori bounded number of homomorphic operations: $O(\log \lambda / \log \log \lambda)$ multiplications followed by $\text{poly}(\lambda)$ additions, where $\lambda \in \mathbb{N}$ is a security parameter. These schemes have compact ciphertexts: before and after homomorphic evaluation, the bit length of each ciphertext is a fixed polynomial in the security parameter λ, independent of the number of homomorphic operations that the scheme supports. This gives the first constructions of somewhat homomorphic encryption that can evaluate the class of bounded-degree polynomials without relying on lattice assumptions or bilinear maps.

Our new encryption schemes are conceptually simple: much as in Gentry, Sahai, and Waters' fully homomorphic encryption scheme, ciphertexts in our scheme are matrices, homomorphic addition is matrix addition, and homomorphic multiplication is matrix multiplication. Moreover, when encrypting many messages at once and performing many homomorphic evaluations at once, the bit length of the ciphertexts in (some of) our schemes can be made arbitrarily close to the bit length of the plaintexts. The main limitation of our schemes is that they require a large evaluation key, whose size scales with the complexity of the homomorphic computation performed, though this key can be re-used across any polynomial number of encryptions and evaluations. Our construction builds on recent work of Dao, Ishai, Jain, and Lin, who construct a homomorphic secret-sharing scheme from the sparse-LPN assumption.

This is the full version of a paper of the same title at Eurocrypt 2025.

Contents

1 Introduction **3**
1.1 Our Results 4
1.2 Technical Overview 6
1.3 Related Work 12

2 Background and Definitions **13**
2.1 Definition of Somewhat Homomorphic Encryption 14
2.2 Definition of Linearly Homomorphic Encryption 15
2.3 Definition of Sparse LPN 15

3 Somewhat Homomorphic Encryption from Sparse LPN and Linearly Homomorphic Encryption **17**

4 Optimizing the Bit Length of Ciphertexts **21**
4.1 Shrinking the Bit Length of Fresh Ciphertexts 22
4.2 Shrinking the Bit Length of Evaluated Ciphertexts 22
4.3 Proof Sketch for Theorem 4.1 25

5 Open Question: Can We Bootstrap? **25**

References **26**

A Additional Material on Sparse LPN **34**

B Additional Material on Somewhat Homomorphic Encryption **35**
B.1 Proof of Theorem 3.1 35
B.2 Proof of Remark 3.3 46

C Additional Material on Optimizations and Batching **47**
C.1 Syntax for Batch Somewhat Homomorphic Encryption 47
C.2 Additional Material for Section 4.1 49
C.3 Additional Material for Section 4.2 51
C.4 Proof of Theorem 4.1 56

1 Introduction

An encryption scheme is *homomorphic* for a class of computations $\mathcal{C}$ if, for all functions $f \in \mathcal{C}$, it is possible to map the ciphertexts $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_m)$ to a new ciphertext $\mathsf{Enc}(f(x_1, \ldots, x_m))$, without any knowledge of the underlying encrypted values [RAD78]. To make the notion of homomorphic encryption both non-trivial and useful, such a scheme must be *compact*: that is, the bit length of ciphertexts $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_m)$, and $\mathsf{Enc}(f(x_1, \ldots, x_m))$ should not grow too much with the complexity of the computation f performed on them [IKR23]. Without this requirement, any encryption scheme could be made trivially homomorphic by concatenating ciphertexts and appending a description of the function f to be applied.

The cryptographer's toolkit today contains many types of homomorphic encryption: *linearly* homomorphic encryption can perform only additions on encrypted data [GM84]; *fully* homomorphic encryption can perform any polynomial-time computation on encrypted data [Gen09]; and *somewhat* homomorphic encryption falls in between these two notions, in that it can perform a bounded class of computations on encrypted data. Despite this restricted expressivity, somewhat homomorphic encryption has wide-ranging applications to the theory of cryptography [OSI05, DPSZ12, CLR17, LT22, CGHK22, LMW23] and the design of privacy-preserving systems [WH12, BPTG14, DGBL+17, ACLS18, HHCP19, JVC18, HLC+22, RCK+21, MW22].

However, constructions of somewhat homomorphic encryption are known from precious few hardness assumptions. Boneh, Goh, and Nissim showed how to homomorphically evaluate degree-two polynomials on encrypted data using bilinear maps [BGN05]. Encryption schemes that can homomorphically evaluate polynomials of degree larger than two rely on the learning-with-errors problem [Reg09,MGH10,BV11,Bra12,BGH13,GSW13,BV14,BGV14], its instantiation over rings [LPR10, GHS12,GHPS13,BLLN13,ASP14,DM15,CGGI16,CKKS17], the NTRU problem [HPS98,LATV12], or the approximate integer greatest-common-divisor problem [VDGHV10], all of which are based on underlying lattice problems (directly or indirectly). Only two alternative approaches to somewhat homomorphic encryption exist. The first is via the route of indistinguishability obfuscation (iO) [CLTV15]. All known iO schemes with a non-vacuous proof of security build on bilinear maps (along with other assumptions) and remain tremendously expensive [JLS21, JLS22, RVV24]. The second is via the route of private information retrieval [CGKS95, KO97], which enables homomorphic evaluation of branching programs [IP07, DGI+19]. However, the bit length of ciphertexts in these schemes grows with the branching program's length. While complexity-theoretic barriers have precluded proving strong lower bounds on the length of branching programs, it is not known how to use these schemes to homomorphically evaluate degree-two polynomials with better than trivial ciphertext size. After many years of study, the pathways to constructing somewhat homomorphic encryption thus remain few and far between; to homomorphically evaluate polynomials of degree two or higher, lattices and bilinear maps are the only known building blocks.

In this paper, we present a new approach for constructing somewhat homomorphic encryption. Our idea is to take a linearly homomorphic encryption scheme and "lift" it into one that can perform a bounded amount of non-linear homomorphic computation. We achieve this lifting step by relying on a coding-theoretic assumption: sparse learning parities with noise [Ale03, AIK06b, IKOS08, ABW10, DIJL23]. This gives the first constructions of somewhat homomorphic encryption for the class of polynomials of degree two (or higher) from assumptions *not* based on lattices or bilinear maps. A key technical idea for our constructions comes from a recent paper of Dao, Ishai, Jain and Lin [DIJL23], who show how to construct a homomorphic secret sharing scheme for certain non-linear functions from sparse LPN.

Our result places somewhat homomorphic encryption on broader foundations: diversifying the assumptions from which it can be built, presenting a range of simple constructions with new efficiency trade-offs, and demonstrating that homomorphic evaluation of bounded-degree polynomials is possible even in a world where lattice problems are easy and iO is impractical.

1.1 Our Results

This paper builds somewhat homomorphic encryption from the sparse learning-parities-with-noise assumption ("sparse LPN"), a coding-theoretic assumption [Ale03,AIK06b,IKOS08,ABW10,DIJL23, RVV24] introduced by Alekhnovich in 2003 [Ale03]. While the standard LPN assumption [GKL93, BFKL93] posits that it is hard to solve a system of random linear equations, over a finite field, corrupted in some locations with random noise, the sparse-LPN assumption states that this task remains hard even when the linear equations are sparse, i.e., only a few variables appear in each equation [Ale03,DIJL23]. An alternate view of the sparse-LPN problem is as the task of decoding a noisy linear code, in which each symbol of the codeword is a random linear combination of a small number of symbols of the message. The sparse-LPN assumption is closely related to the hardness of random constraint-satisfaction problems [Gol00, CM01, Fei02, AIK06a, ABW10, AOW15, AL16, KMOW17].

In more detail, the sparse-LPN assumption (defined formally in Section 2) is parameterized by a secret dimension n, a number of samples m, a sparsity k, a prime modulus q, and an error parameter δ. The assumption asserts that, for a matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ with k random non-zero entries in each row, a secret vector $\mathbf{s} \xleftarrow{R} \mathbb{F}_q^n$, and an error vector $\mathbf{e} \in \mathbb{F}_q^m$ in which each entry is a random non-zero value in $\mathbb{F}_q$ with probability $n^{-\delta}$ and zero otherwise, it is computationally hard to distinguish $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ from $(\mathbf{A}, \mathbf{r})$, for a uniform random $\mathbf{r} \xleftarrow{R} \mathbb{F}_q^m$. Given a security parameter $\lambda \in \mathbb{N}$, our results work in the following setting: the dimensions n, m are polynomials in λ, the sparsity k is polylogarithmic in λ, and the error parameter $\delta \in (0, 1)$ is a constant, meaning that each sparse-LPN sample has inverse-polynomial noise rate $n^{-\delta}$.

Somewhat homomorphic encryption using sparse LPN. From sparse LPN and a secret-key linearly homomorphic encryption scheme, we show how to construct a secret-key somewhat homomorphic encryption scheme for the class of polynomials with bounded degree and a bounded number of monomials. In Section 3, we prove the following result:

Informal Theorem 1.1. *On security parameter $\lambda \in \mathbb{N}$ and given a prime modulus $q \geq 3$ of size at most exponential in λ, assume that sparse LPN with modulus q, $O(\sqrt{\log \lambda})$-sparsity, and any inverse-polynomial noise rate is hard. Assume the existence of a linearly homomorphic encryption scheme with message space $\mathbb{F}_q$. Then, for every constant $c \in \mathbb{N}$, there exists a somewhat homomorphic encryption scheme capable of evaluating multivariate polynomials over $\mathbb{F}_q$ with total degree $c \cdot \log \lambda / \log \log \lambda$ and up to λ^c monomials.*

Our somewhat homomorphic encryption scheme is compact: the bit length of all ciphertexts is a fixed polynomial in λ, independent of the class of homomorphic computations that the scheme supports. However, to achieve this efficiency, our scheme requires the transmission of a large evaluation key to any party that performs computations on ciphertexts. The size of this evaluation key scales polynomially with the number of additions and exponentially with the number of multiplications that can be performed. Concretely, given the above parameter $c \in \mathbb{N}$, which determines how expressive the class of homomorphic computations is that our scheme supports, the bit length of the evaluation key grows exponentially with c. At the same time, the holder of a secret key needs

only to publish its evaluation key once; thereafter, anyone can use this evaluation key to evaluate any polynomial number of functions on any polynomial number of ciphertexts (encrypted under the corresponding secret key) — regardless of the value of c. In this way, it is possible to amortize away the cost of communicating the large evaluation key.

Instantiating our construction with concrete linearly homomorphic encryption schemes [ElG85, Pai99, DJ01], we obtain new somewhat homomorphic encryption schemes from sparse LPN and the hardness of either the decisional Diffie-Hellman (DDH) [DH76, Bon98] or the decisional composite residuosity (DCR) [Pai99] problems. (For ways to construct an analogous scheme from quadratic residuosity [GM84], see Remark 3.7.) Our schemes are the first constructions of homomorphic encryption for the class of polynomials of degree two or higher that do not require assumptions based on lattices or pairings.

Finally, we show how to shrink the bit length of the ciphertexts in our somewhat homomorphic schemes, when they are used to encrypt many messages at once and evaluate many polynomials at once. In this setting, we show that the sum of the ciphertexts' bit lengths — before and after homomorphic evaluation — can be arbitrarily close to the bit lengths of the plaintexts that they encrypt. In Section 4, we describe a scheme with the following efficiency:

Informal Theorem 1.2. *On security parameter $\lambda \in \mathbb{N}$ and any prime modulus $q \geq 3$ of size at most polynomial in λ, assume that sparse LPN with modulus q, $O(\sqrt{\log \lambda})$-sparsity, and any inverse-polynomial noise rate is hard and that DDH is hard. Then, there exists a somewhat homomorphic encryption scheme with message space $\mathbb{F}_q$ and the homomorphic capabilities of Informal Theorem 1.1, such that, when evaluating a batch of t polynomials, each in $\mathbb{F}_q^m \to \mathbb{F}_q$, on a batch of tm inputs,*

- *all tm input ciphertexts together consist of $tm \cdot \log q + m \cdot \operatorname{poly}\log(\lambda)$ bits, in addition to a one-time evaluation key of $t^2 \cdot \operatorname{poly}(\lambda)$ bits, and*
- *all t output ciphertexts together consist of $t \cdot \log q + \operatorname{poly}(\lambda)$ bits.*

Here, when the number of polynomials t evaluated in a batch grows large, each encrypted input and output is represented using roughly one element in $\mathbb{F}_q$ — this matches the bit length of the plaintext space, up to additive factors. As before, when evaluating sufficiently many batches of polynomials, we can amortize away the cost of transmitting the evaluation key. We say that such a scheme has "batch rate" one.

Our constructions draw inspiration from two sources: a recent homomorphic secret-sharing scheme from sparse LPN [DIJL23] and the Gentry-Sahai-Waters fully homomorphic encryption scheme [GSW13] based on the learning-with-errors assumption (LWE). In particular, our results can be viewed in two ways:

In the first view, our schemes modify Dao, Ishai, Jain, and Lin's homomorphic secret-sharing scheme from sparse LPN [DIJL23] to work with only a single party, rather than multiple non-colluding parties. To be more precise, Dao et al. construct a homomorphic secret-sharing scheme from sparse LPN and an arbitrary linear secret-sharing scheme $\mathcal{L}$. If we instantiate the linear *secret-sharing* scheme $\mathcal{L}$ in their construction with a linearly homomorphic *encryption* scheme, we immediately obtain a somewhat homomorphic encryption scheme from sparse LPN. This basic scheme does not satisfy our ciphertext-compactness notion, so we apply an extra ciphertext-compression step to arrive at our final construction.

Taking another perspective, our schemes closely resemble Gentry, Sahai, and Waters' fully homomorphic encryption from LWE [GSW13], ported to the setting of code-based encryption. This

connection lets us draw upon many optimizations developed in the context of lattice-based encryption [PVW08, BV14, BGV14, CGGI20, dCHI+22] to improve our schemes' efficiency. Like the Gentry-Sahai-Waters encryption scheme, our somewhat homomorphic encryption is simple: ciphertexts are sparse matrices, homomorphic addition is just matrix addition, and homomorphic multiplication is just matrix multiplication, without even the bit-decomposition of the Gentry-Sahai-Waters scheme. The main limitation of our schemes is that they remain much less homomorphic than their lattice-based counterparts, which are fully homomorphic.

Can we bootstrap? One question left open by our work is whether it is possible to bootstrap our schemes to construct fully homomorphic encryption from sparse LPN and linearly homomorphic encryption (possibly with another mild assumption). We discuss why this appears challenging in Section 5.

In particular, a natural approach [Gen09] would be to instantiate our constructions with a linearly homomorphic encryption scheme whose decryption circuit can be homomorphically evaluated by our somewhat homomorphic schemes — that is, we need linearly homomorphic encryption whose decryption algorithm can be written as a multivariate polynomial of total degree $O(\log \lambda / \log\log \lambda)$ with $\mathrm{poly}(\lambda)$ monomials. However, this route likely runs into trouble: first, we do not know of such a linearly homomorphic scheme. Second, if there were to be such a linearly homomorphic encryption scheme, together with our constructions, it appears to imply a gadget that can generate many sparse-LPN samples from a small number of them. This in turn would render such a bootstrapped scheme vulnerable to attacks on sparse LPN in the many-sample regime.

1.2 Technical Overview

We begin with an overview of each of the constructions and results in this work.

Sparse learning parity with noise. The sparse learning-parity-with-noise assumption posits that it is computationally hard to solve a sparse system of linear equations over a finite field, which has been corrupted with random noise in a fraction of locations. In more detail, given a prime modulus $q \geq 2$, dimensions $m, n \in \mathbb{N}$, a sparsity parameter $k \leq n$, and a constant error parameter $\delta \in (0,1)$, we write $\mathcal{S}_{k,m,n,q}$ to denote the set of matrices in $\mathbb{F}_q^{m \times n}$ whose rows are each *k-sparse* (i.e., contain exactly k non-zero entries). We write $\mathsf{RandBern}_{n^{-\delta},q}$ to denote a Bernoulli random variable that takes on a random value in $\mathbb{F}_q \setminus \{0\}$ with probability $n^{-\delta}$, and takes on the value 0 otherwise.

Then, the sparse-LPN assumption with parameters (n, m, q, δ, k), which implicitly grow with an underlying security parameter, states that the following two distributions are computationally indistinguishable:

$$\left\{ (\mathbf{A}, \mathbf{As}+\mathbf{e}) : \begin{array}{r} \mathbf{A} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,m,n,q} \\ \mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \\ \mathbf{e} \leftarrow \mathsf{RandBern}_{n^{-\delta},q}^m \end{array} \right\} \overset{c}{\approx} \left\{ (\mathbf{A}, \mathbf{u}) : \begin{array}{r} \mathbf{A} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,m,n,q} \\ \mathbf{u} \xleftarrow{\mathrm{R}} \mathbb{F}_q^m \end{array} \right\}.$$

At first glance, sparse LPN looks very similar to the LWE assumption: the assumptions differ only in their choice of the $\mathbf{A}$-matrix (taken to be sparse in sparse LPN) and their error distribution (LWE uses low-norm errors that perturb every location, while sparse LPN uses high-norm errors that completely corrupt a small fraction of locations). However, while LWE gives rise to some of the most powerful cryptographic tools including fully homomorphic encryption, only a handful of

primitives are known from sparse LPN: among them, public-key encryption [Ale03, ABW10], cryptographic tools with constant overhead [AIK06b, IKOS08, ADI+17], and multi-party computation with sublinear communication [DIJL23].

Building somewhat homomorphic encryption. In Section 3, we introduce new constructions of somewhat homomorphic encryption from sparse LPN, along with an assumption that implies linearly homomorphic encryption (e.g., DDH or DCR). At its core, our construction relies on two key design steps:

Step 1: Leveraging sparse LPN's structure to support homomorphic operations. The reason why sparse LPN implies an encryption scheme that can support both homomorphic additions and multiplications — unlike many other cryptographic assumptions — is because it is powerful enough to homomorphically evaluate its own decryption circuit (a bounded number of times)[1]. Dao, Ishai, Jain, and Lin implicitly exploited this structural property of sparse LPN to build a homomorphic secret sharing scheme from sparse LPN [DIJL23].

In particular, we can build a Regev-style [Reg09] encryption scheme from sparse LPN that, given a secret key $\mathbf{s} \in \mathbb{F}_q^n$ and a message $\mu \in \mathbb{F}_q$, produces the ciphertext

$$\mathsf{RegevEnc}_{\mathbf{s}}(\mu) = (\mathbf{a}, \mathbf{a}^\intercal \mathbf{s} + e + \mu) \in \mathbb{F}_q^n \times \mathbb{F}_q,$$

for a random k-sparse vector $\mathbf{a} \in \mathbb{F}_q^n$ and a Bernoulli error $e \leftarrow \mathsf{RandBern}_{n^{-\delta},q}$. The sparse-LPN assumption implies that the value $\mathbf{a}^\intercal \mathbf{s} + e$ is computationally indistinguishable from random, so the ciphertext hides the message μ. In addition, this scheme has the following properties:

1. It can homomorphically evaluate linear functions with few (i.e., up to $O(n^\delta)$) variables. More precisely, we can add up c such ciphertexts encrypted with the same secret key $\mathbf{s}$; the result will be an encryption of the sum of the messages under secret key $\mathbf{s}$, with c times larger error probability.

2. Its decryption algorithm is a linear function in only a few variables. In particular, given a ciphertext $(\mathbf{a}, b) \in \mathbb{F}_q^n \times \mathbb{F}_q$ encrypted with secret key $\mathbf{s} \in \mathbb{F}_q^n$, decryption evaluates the function

$$f_{\mathbf{a},b}(x_1, \ldots, x_n) = b - \mathbf{a}^\intercal \mathbf{x} = b - \sum_{i=1}^{n} a_i \cdot x_i$$

on input $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{F}_q^n$, where the vector $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}_q^n$ is public and has only a few non-zero entries. Decryption succeeds if the error e in the ciphertext is zero; this happens with good probability, as long as not too many homomorphic additions have been performed.

Combining these two properties, we follow the template of Gentry, Sahai, and Waters [GSW13] and Micciancio's [Mic19] "linear-decrypt-and-multiply" framework — applied previously in the context of lattice-based encryption — to support homomorphic multiplications. To do so, for each message μ and each entry s_i of the secret key $\mathbf{s} \in \mathbb{F}_q^n$, we publish the encryption $\mathsf{RegevEnc}_{\mathbf{s}}(\mu \cdot s_i)$.

[1] Unlike in prior fully homomorphic encryption schemes, the fact that we can homomorphically evaluate the decryption circuit does not give us a scheme that is "bootstrappable." This is because the amount of error present in our ciphertexts is always larger after homomorphically evaluating the decryption circuit than before. So, we can make use of homomorphic decryption only to support multiplications, not to control error growth.

(Here, as we will see in Section 2.3, to ensure key-dependent-message security from sparse LPN, we sample the $\mathbf{a}$-components of such a ciphertext to be k-sparse vectors with a non-zero value in their i^{th} entry, and we work over a modulus $q \geq 3$.) Then, given the encryption of two inputs $\mathsf{RegevEnc}_{\mathbf{s}}(\mu)$ and $\mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu})$, along with special encryptions of the form $\mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu} \cdot s_i)$ for each $i \in [n]$, we compute $\mathsf{RegevEnc}_{\mathbf{s}}(\mu\hat{\mu})$ by:

- thinking of each component $(\mathbf{a}, b)$ of $\mathsf{RegevEnc}_{\mathbf{s}}(\mu)$ as a "plaintext" in $\mathbb{F}_q^n \times \mathbb{F}_q$,
- computing the plaintext multiplication $\mathsf{ct}_{\mu\cdot\hat{\mu}} = b \cdot \mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu})$, and
- homomorphically "decrypting" this ciphertext $\mathsf{ct}_{\mu\cdot\hat{\mu}}$ by subtracting the inner-product of the plaintext vector $\mathbf{a}$ with the ciphertext vector $\mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu} \cdot \mathbf{s})$.

By the above two properties, this final step produces exactly the output we need:

$$\begin{aligned}
b \cdot \mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu}) - \mathbf{a}^{\mathsf{T}} \cdot \mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu} \cdot \mathbf{s}) &= b \cdot \mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu}) - \sum_{i=1}^{n} a_i \cdot \mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu} \cdot s_i) \\
&\approx \mathsf{RegevEnc}_{\mathbf{s}}(b\hat{\mu} - \hat{\mu} \cdot \mathbf{a}^{\mathsf{T}}\mathbf{s}) \qquad \text{(property 1.)} \\
&= \mathsf{RegevEnc}_{\mathbf{s}}((b - \mathbf{a}^{\mathsf{T}}\mathbf{s}) \cdot \hat{\mu}) \\
&= \mathsf{RegevEnc}_{\mathbf{s}}(f_{\mathbf{a},b}(\mathbf{s}) \cdot \hat{\mu}) \\
&\approx \mathsf{RegevEnc}_{\mathbf{s}}(\mu\hat{\mu}). \qquad \text{(property 2.)}
\end{aligned}$$

We can similarly multiply $\mathsf{RegevEnc}_{\mathbf{s}}(\hat{\mu})$ by each $\mathsf{RegevEnc}_{\mathbf{s}}(\mu \cdot s_i)$ to produce $\mathsf{RegevEnc}_{\mathbf{s}}(\mu\hat{\mu} \cdot s_i)$. This lets us compute exactly the inputs that we need to perform another multiplication — namely, the encryptions $\mathsf{RegevEnc}_{\mathbf{s}}(w)$ and $\mathsf{RegevEnc}_{\mathbf{s}}(w \cdot \mathbf{s})$ for every intermediate wire value $w \in \mathbb{F}_q$ — allowing us to keep multiplying and adding until the error growth becomes unmanageable.

Our somewhat homomorphic encryption schemes follow exactly this template. More concretely, given a secret key $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{F}_q^n$ and a message $\mu \in \mathbb{F}_q$, we compute a ciphertext matrix $\mathbf{C} \in \mathbb{F}_q^{(n+1)\times(n+1)}$ encrypting μ as

$$\mathbf{C} = \begin{bmatrix} \mathsf{RegevEnc}_{\mathbf{s}}(-\mu \cdot s_1) \\ \mathsf{RegevEnc}_{\mathbf{s}}(-\mu \cdot s_2) \\ \vdots \\ \mathsf{RegevEnc}_{\mathbf{s}}(-\mu \cdot s_n) \\ \mathsf{RegevEnc}_{\mathbf{s}}(\mu) \end{bmatrix} = \left[\mathbf{A} \quad || \quad \mathbf{A}\mathbf{s} + \mathbf{e} + \mu \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix}\right], \tag{1}$$

where $\mathbf{A} \in \mathbb{F}_q^{(n+1)\times n}$ is a matrix whose rows are each k-sparse and the vector $\mathbf{e} \in \mathbb{F}_q^{n+1}$ has entries sampled from the Bernoulli error distribution. To decrypt ciphertext $\mathbf{C}$, we run Regev decryption (property 2 above) on the last row of the ciphertext matrix. To perform homomorphic additions on ciphertexts $\mathbf{C}_1$ and $\mathbf{C}_2$, we add the ciphertext matrices: this operation exactly adds the underlying Regev encryptions, stored in each row of the matrices. To perform homomorphic multiplications on ciphertexts $\mathbf{C}_1$ and $\mathbf{C}_2$, we multiply the ciphertext matrices: this operation exactly implements the ciphertext tensoring and decryption steps described above to multiply.

We give an alternate view of this scheme, which closely resembles Gentry, Sahai, and Waters' fully homomorphic encryption from lattices [GSW13] (referred to as the "GSW scheme"), in our

proof of correctness in Section 3 and Appendix B.1. At a high level, this view focuses on the fact that our ciphertext $\mathbf{C}$ (constructed as in Equation (1)) is a sparse matrix that has the secret key as an "approximate" eigenvector and the encrypted message as the corresponding eigenvalue: namely, it holds that

$$\mathbf{C} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = \mu \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}. \tag{2}$$

(Here, unlike in GSW, the "approximate" eigenvector is corrupted by sparse-LPN errors $\mathbf{e} \in \mathbb{F}_q^{n+1}$, which are high-norm but only non-zero in a small fraction of locations.)

As matrix addition/multiplication preserves the eigenvectors of a matrix while adding/multiplying the eigenvalues, we can perform homomorphic operations. In more detail, for any pair of matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ that satisfy Equation (2) with respect to messages $\mu_1, \mu_2 \in \mathbb{F}_q$ and error vectors $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_q^{n+1}$, we observe that:

- *Homomorphic addition.* Equation (2) holds for the summed matrix $\mathbf{C}_1 + \mathbf{C}_2$, with respect to new message $(\mu_1 + \mu_2) \in \mathbb{F}_q$ and new error vector $(\mathbf{e}_1 + \mathbf{e}_2) \in \mathbb{F}_q^{n+1}$:

$$(\mathbf{C}_1 + \mathbf{C}_2) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = \mathbf{C}_1 \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{C}_2 \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = \underbrace{(\mu_1 + \mu_2)}_{\text{new message}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \underbrace{(\mathbf{e}_1 + \mathbf{e}_2)}_{\text{new error}}.$$

  Since vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are each non-zero in only a few locations, we can conclude that the new error vector $(\mathbf{e}_1 + \mathbf{e}_2)$ is also only non-zero in a few locations. As a result, with each homomorphic addition, the error-rate grows additively and can be bounded.

- *Homomorphic multiplication.* Equation (2) holds for the product matrix $\mathbf{C}_1 \cdot \mathbf{C}_2$, with respect to new message $(\mu_1 \cdot \mu_2) \in \mathbb{F}_q$ and new error vector $(\mu_2 \cdot \mathbf{e}_1 + \mathbf{C}_1 \cdot \mathbf{e}_2) \in \mathbb{F}_q^{n+1}$:

$$(\mathbf{C}_1\mathbf{C}_2) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = \mathbf{C}_1 \cdot \left(\mu_2 \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}_2 \right) = \underbrace{(\mu_1\mu_2)}_{\text{new message}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \underbrace{(\mu_2\mathbf{e}_1 + \mathbf{C}_1\mathbf{e}_2)}_{\text{new error}}.$$

  Here, since vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are each non-zero in only a few locations, and since matrix $\mathbf{C}_1$ is sparse, we can conclude that the new error vector $(\mu_2 \cdot \mathbf{e}_1 + \mathbf{C}_1 \cdot \mathbf{e}_2)$ is also only non-zero in a few locations. As a result, with each homomorphic multiplication, the error-rate grows multiplicatively (in the ciphertext matrix $\mathbf{C}_1$'s sparsity) and can also be bounded.

  Crucially, this step only succeeds when the ciphertext matrices are sufficiently sparse — this is why our construction requires sparse LPN instead of plain LPN.

Unlike in lattice-based cryptography, and in the GSW scheme in particular, we must deal with much more aggressive error growth since any errors in the ciphertexts corrupt all computations that they touch (rather than just corrupting the lowest-order bits). This limits the number of homomorphic operations that the scheme can support.

Step 2: Using hybrid encryption to achieve compactness. While sparse LPN on its own is sufficient to implement homomorphic additions and multiplications, these operations are not compact. To ensure that the final output of a computation will not get corrupted by the Bernoulli errors (with good probability), the more homomorphic operations we want to perform, the larger we need to set the sparse-LPN dimension n to be, and the bulkier our fresh ciphertexts — which are sparse

matrices of dimension $(n+1)$-by-$(n+1)$ — must get. Worse yet, with each operation, the ciphertext matrices grow less sparse, and thus require more bits to represent. As a result, representing a ciphertext matrix $\mathbf{C}$ always requires more bits than just representing the function f that has been homomorphically applied to it.

To circumvent this issue, we use homomorphism to translate between different ciphertext representations. In doing so, we obtain a somewhat homomorphic encryption scheme that uses the non-compact "GSW-style" ciphertexts (from step 1 above) to perform homomorphic operations, but where all input and output ciphertexts are made compact. In particular, we have our encryption algorithm publish compact, Regev-style encryptions of the input messages (secure under sparse LPN). Then, we let any party that wishes to compute on these ciphertexts (a) translate these encryptions into much larger, GSW-style ciphertexts (again, secure under sparse LPN), (b) perform homomorphic operations on these GSW-style ciphertexts, and (c) translate the GSW-style output ciphertext into a single ciphertext from a compact, linearly homomorphic encryption scheme (e.g., secure under DDH or DCR). Under the hood, each of these translations from one encryption scheme to the next runs "homomorphic decryption" on the former scheme's ciphertexts, using an encryption of the former scheme's secret key under the next scheme.

In more detail, our final scheme proceeds as follows:

1. We have the encryptor sample two sparse-LPN secrets $\mathbf{s}, \mathbf{t} \in \mathbb{F}_q^n$, and publish a one-time "evaluation key" which holds the GSW-style encryption of each entry of the vector $\mathbf{t}$, under the secret key $\mathbf{s}$. In addition, we have the encryptor publish the encryptions of each entry of the vector $\mathbf{s} \in \mathbb{F}_q^n$ under a second, only *linearly* homomorphic encryption scheme (with message space $\mathbb{F}_q$) in the evaluation key.

2. Then, for each input message $\mu \in \mathbb{F}_q$, the encryptor publishes only a much smaller, Regev-style encryption of message μ under secret key $\mathbf{t}$ — i.e., $\mathsf{RegevEnc}_{\mathbf{t}}(\mu)$ — which is a sparse vector whose bit length scales only logarithmically with the large dimension n.

3. Given the evaluation key, anyone can convert these small, Regev-style encryptions under the secret key $\mathbf{t}$ into a large, GSW-style encryption of the same message under secret key $\mathbf{s}$: it suffices to homomorphically decrypt the Regev-style ciphertexts, using the GSW-style encryption of their secret key. This step relies on the homomorphism of our non-compact GSW-style ciphertexts, which lets us homomorphically evaluate the linear Regev-decryption function.[2]

4. After performing all desired operations on these (non-compact) GSW-style ciphertexts, any party that knows the evaluation key can homomorphically decrypt them under the linearly homomorphic encryption scheme, to obtain (compact) ciphertexts that encrypt the same underlying message. This step takes advantage of the fact that, in our "GSW-style" sparse-LPN encryption, decryption is a *linear* function of the secret key $\mathbf{s} \in \mathbb{F}_q^n$ — and so it can be evaluated under encryption, using a linearly homomorphic scheme.

This gives our main result: somewhat homomorphic encryption that can homomorphically evaluate multivariate polynomials of bounded total degree $O(\log \lambda / \log\log \lambda)$ with an a-priori bounded $\mathrm{poly}(\lambda)$ number of monomials (Informal Theorem 1.1).

[2]Though we describe it here in the language of homomorphic decryption, this technique is analogous to the "external product" operation for switching Regev to GSW ciphertexts in lattice-based encryption [CGGI20, MW22].

Achieving "batch rate" one. In Section 4, we show how to reduce the bit length of our scheme's ciphertexts, when encrypting many values at once and homomorphically evaluating many polynomials at once. That is, when computing over any prime modulus $q = \text{poly}(\lambda)$ and when homomorphically evaluating a batch of t polynomials, each mapping $\mathbb{F}_q^m \to \mathbb{F}_q$, on tm input ciphertexts, the bit length of all tm input ciphertexts together can be arbitrarily close to $tm \cdot \log q$, as t grows large. This is the minimum number of bits necessary to represent the input plaintexts. The bit length of all t output ciphertexts together can be arbitrarily close to $t \cdot \log q$, as t grows large. This is the minimum number of bits necessary to represent the output plaintexts. Finally, by evaluating sufficiently many batches of polynomials, we can amortize away the cost of transmitting the one-time evaluation key (Informal Theorem 1.2).

To achieve this efficiency, in Section 4.1, we fix and re-use large portions of the sparse-LPN ciphertexts at only a polynomial loss in security, following a standard technique in the lattice literature [PVW08]. At this point, each fresh ciphertext consists of *one* value in $\mathbb{F}_q$ (plus a single, re-usable preamble of $\text{poly}(\lambda)$ bits) — when encrypting a sufficiently large batch of values, this essentially matches the plaintext space. In Section 4.2, we shrink the bit length of ciphertexts after homomorphic evaluation. To this end, we construct a linearly homomorphic encryption scheme from DDH that can evaluate linear functions over an arbitrarily small modulus q (that is of size at most $\text{poly}(\lambda)$) and that achieves high rate in a specific context: namely, when homomorphically applying a batch of many *different* linear functions to many ciphertexts, each output ciphertext consists of roughly one element in $\mathbb{F}_q$. Our linearly homomorphic encryption scheme from DDH combines ideas from recent "distributed discrete logarithm" protocols [BGI16, DGI+19, BBD+20, BBDP22] and rate-one encryption schemes from LWE [dCHI+22]. Using this linearly homomorphic scheme, our somewhat homomorphic encryption can perform "batch compaction" after evaluating multiple different polynomials on different ciphertexts. When doing so with batch size t, its output is a "packed" DDH ciphertext that holds all t polynomial evaluations, each encoded as a value in $\mathbb{F}_q$ (plus a single preamble of λ bits) — again, when encrypting a sufficiently large batch of outputs, this essentially matches the plaintext space.

Discussion: The power of sparse LPN. At first sight, sparse LPN seems like an assumption with limited utility: in the parameter regime we work in, sparse LPN does not necessarily imply that a hard problem in the complexity class SZK exists (where SZK denotes the class of languages with statistical zero-knowledge proofs), a prerequisite for building any form of compact homomorphic encryption [BLVW19, DJ24]. Moreover, our schemes also work in sparse-LPN parameter settings that are not known to imply public-key encryption (which has only been constructed from LPN with constant sparsity [ABW10, DIJL23, RVV24], sufficiently low noise rate [Ale03], or subexponential hardness [YZ16]). Dao et al. use sparse LPN, in the same parameter regime as we do, to build homomorphic secret-sharing schemes that are not known to exist under secret-key assumptions [DIJL23]. Their results suggest that sparse LPN may be more powerful than it might appear at first.

Indeed, once paired with other relatively mild cryptographic assumptions — in our case, the existence of linearly homomorphic encryption — sparse LPN seems to act as a powerful catalyst towards constructing more advanced cryptography. An analogous, equally intriguing phenomenon appears in the study of indistinguishability obfuscation [JLS21, JLS22, RVV24], where sparse LPN together with other standard assumptions (specifically, the DLIN assumption on bilinear maps and LPN over large fields) gives iO. Our work makes progress on harnessing the power of sparse LPN in a different, albeit more modest, setting.

Assumption(s)	Homomorphic encryption for...	Asymptotic bit length: *ciphertexts*	Asymptotic bit length: *evaluation key*
Linearly homomorphic enc.			
DDH, QR, DCR, or LWE	degree-1 polynomials	1	none
Somewhat homomorphic enc.			
Bilinear maps	degree-2 polynomials	1	none
DDH, DCR, QR, or LWE	branching programs of length ℓ	$\mathrm{poly}(\ell)$	none
Sparse LPN + DDH/DCR	polynomials of degree $d = O(\log \lambda / \log\log \lambda)$ with M monomials	1	$\mathrm{poly}(M(\log \lambda)^d)$
Fully homomorphic enc.			
LWE	any degree-d polynomial	1	$\log d$
circular-secure LWE	any polynomial	1	1
iO + re-randomizable enc.	any polynomial	1	1

Table 1: The assumptions that give rise to homomorphic encryption, compared based on (1) the class of homomorphic computations that they support, (2) the asymptotic bit length of ciphertexts, and (3) the asymptotic bit length of the one-time evaluation key. All schemes implied by LWE are also implied by the ring-LWE assumption. For simplicity, we suppress fixed polynomials in the security parameter λ.

Yet a different interpretation of our results is that they "decompose" the properties of the learning-with-errors (LWE) assumption needed to build somewhat homomorphic encryption. As described earlier, in our schemes, sparse LPN provides linear decryption, while linearly homomorphic encryption provides compression. LWE has *both* of these properties baked in.

1.3 Related Work

Homomorphic encryption. Rivest, Adleman, and Dertouzos first proposed the notion of homomorphic encryption in 1978 [RAD78]. Many early encryption schemes from number-theoretic assumptions support natural but limited types of homomorphism: they can compute only multiplications [ElG85] or only additions [Rab79, GM84, ElG85, Ben87, Pai99, DJ01] on their ciphertexts. Using bilinear maps [BF01], Boneh, Goh, and Nissim gave an encryption scheme supporting any number of additions and one multiplication (i.e., degree-two polynomials) [BGN05]. Gentry, Halevi, and Vaikuntanathan later showed how to evaluate this same class of functions with an encryption scheme from LWE [GHV10]. However, these works left open the question of how to build homomorphic encryption for a larger class of functions, e.g., polynomials of higher degree.

In his 2009 thesis, Gentry gave a fully homomorphic encryption scheme from ideal lattices [Gen09], kicking off a cascade of works building lattice-based fully homomorphic schemes from standard assumptions [LPR10, VDGHV10, BV11, Bra12, LATV12, BLLN13, BV14] with better efficiency [GHS12, GHPS13, BGH13, BGV14, ASP14, DM15, CGGI16, CKKS17, GH19, BDGM19, CGGI20] and simpler constructions [GSW13]. Melchor, Gaborit, and Herranz leveraged a certain flavor of linearly homomorphic encryption to evaluate low-degree polynomials and instantiated their framework from lattice-based assumptions [MGH10, MBGH11].

Two main lines of work have constructed somewhat/fully homomorphic encryption from standard assumptions *not* based on bilinear maps or lattices (Table 1). First, in 2007, Ishai and Paskin described a generic transformation that uses any linearly homomorphic encryption with sufficiently short ciphertexts (known today from DCR, QR, DDH, and LWE) to evaluate branching programs

under encryption [IP07,DGI+19]. Ishai and Paskin's scheme can homomorphically evaluate any log-space or NC^1 computation. However, unlike the schemes in this paper, Ishai and Paskin's scheme is only weakly compact: the size of evaluated ciphertexts scales polynomially with the branching program's length, but is independent of its size. To date, it is not known how to use Ishai and Paskin's scheme to homomorphically evaluate the class of polynomials of degree two (or higher) with better than trivial ciphertext size. Second, recent constructions of indistinguishability obfuscation together with any re-randomizable encryption scheme (e.g., based on DDH, DCR, or QR) imply fully homomorphic encryption [CLTV15]. However, known iO constructions are tremendously inefficient, requiring passing through the intermediate notions of functional encryption or iO with exponential efficiency [AJ15, BV15, LPST16]. Our schemes are vastly simpler than the constructions implied by iO, under many complexity notions: number of lines of code to implement, number of pages to describe, etc. Moreover, all known iO schemes [JLS21, JLS22, RVV24] require bilinear maps, while our schemes do not.

Brakerski gave evidence that known LPN-based encryption schemes cannot be somewhat homomorphic [Bra13]: to break a candidate scheme [BL11], he proved that any homomorphic encryption that is powerful enough to evaluate the majority function cannot have a "weakly learnable" decryption circuit, as standard LPN-based schemes with linear decryption do. Our schemes get around this impossibility result by working with a second assumption (DDH or DCR) and thus the decryption circuit of our schemes is not linear.

Connections to homomorphic secret sharing. Homomorphic secret sharing is the multi-party counterpart of homomorphic encryption: data is shared among multiple parties such that no individual party can recover the data, but together the parties can perform computations on it without interacting [BGI16, BGI+18]. Micciancio [Mic19] showed that Gentry, Sahai, and Waters' fully homomorphic encryption from lattices [GSW13] and recent homomorphic secret sharing schemes from computational assumptions [BGI16,BKS19,FGJI17,OSY21,RS21,DIJL23] are closely related: both make use of a "linear-decrypt-and-multiply" operation to implement homomorphic multiplications. Our new somewhat homomorphic encryption also falls under this same framework. One view of our schemes is that they make Dao, Ishai, Jain, and Lin's recent homomorphic secret-sharing scheme from sparse LPN [DIJL23] work with only a single party — rather than multiple non-colluding parties — by substituting the use of a linear secret-sharing scheme with linearly homomorphic encryption.

2 Background and Definitions

We begin by defining the cryptographic primitives and hardness assumptions used in this work.

Notation. Throughout, we assume that sparse vectors and matrices are represented efficiently in memory, and in particular we define the bit length of a vector in $\mathbb{F}_q^n$ with $k \ll n$ non-zero entries to be $k \cdot \log q \cdot \log n$. Our model of computation is a RAM machine with $O(1)$ cost per memory access and with word-size linear in the security parameter, $\lambda \in \mathbb{N}$. We measure the cost of algorithms and computations in terms of the number of RAM operations they require.

We write $\mathcal{D} \overset{c}{\approx} \mathcal{D}'$ to denote that samples from two distributions $\mathcal{D}$ and $\mathcal{D}'$ are computationally indistinguishable. We use $\mathrm{negl}(\lambda)$ to denote a function that is negligibly small in λ, $\mathrm{poly}(\lambda)$ to denote one that is polynomial in λ, and $\exp(\lambda)$ to denote one that is exponential in λ. We write the entries of a vector $\mathbf{v} \in \mathbb{F}_q^n$ as $(v_1, \ldots, v_n)$. We write the set $\{1, 2, \ldots, n\}$ as $[n]$. For a set of elements $\mathcal{S}$, we

write $v \xleftarrow{R} \mathcal{S}$ to denote sampling a uniform random element v from $\mathcal{S}$. For a probability distribution $\mathcal{D}$, we write $v \leftarrow \mathcal{D}$ to denote sampling an element v from $\mathcal{D}$. For a randomized algorithm Alg, we write $v \leftarrow \mathsf{Alg}$ to denote the output of a run of Alg. We write $v := x$ to assign the value of x to v. We write $\mathbb{1}_{\mathsf{cond}}$ to denote the indicator variable that is "1" if condition cond is true, and "0" otherwise. We omit the notation $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ and treat values like n^ϵ and n/k as integers. All logarithms are base two.

Standard primitives. We use the standard definition of pseudorandom functions (PRFs) [Gol01]. On key space $\mathcal{K}$, input space $\mathcal{I}$, and output space $\mathcal{O}$, we denote a PRF as $\mathsf{PRF} : \mathcal{K} \times \mathcal{I} \to \mathcal{O}$.

2.1 Definition of Somewhat Homomorphic Encryption

An encryption scheme is "somewhat homomorphic" if a bounded class of computations may be performed on its ciphertexts. Following Halevi's definition [Hal17], we describe such a scheme in terms of a security parameter $\lambda \in \mathbb{N}$, which governs how hard it is to break the scheme's security, and a functionality parameter $\tau \in \mathbb{N}$, which governs how many homomorphic operations the scheme supports. Then, we define a somewhat homomorphic encryption scheme relative to a correctness failure probability $\epsilon = \epsilon(\lambda) \in [0, 1]$ and a function class $\mathcal{F}_\tau \subseteq \{f : \mathcal{M}^* \to \mathcal{M}\}$, which comprises the computations that can be homomorphically performed on ciphertexts:

Definition 2.1 (Somewhat Homomorphic Encryption)**.** Given a key space $\mathcal{K}$, a message space $\mathcal{M}$, and a ciphertext space $\mathcal{C}$, a *somewhat homomorphic encryption* (SHE) scheme for the function class $\mathcal{F}_\tau$ is a tuple of four polynomial-time algorithms:

- $\mathsf{Gen}(1^\lambda, 1^\tau) \to (\mathsf{sk}, \mathsf{ek})$, a randomized algorithm that takes as input a security parameter $\lambda \in \mathbb{N}$ and a functionality parameter $\tau \in \mathbb{N}$ and outputs a secret key $\mathsf{sk} \in \mathcal{K}$ and an evaluation key $\mathsf{ek} \in \mathcal{K}$.
- $\mathsf{Enc}(\mathsf{sk}, \mu) \to \mathsf{ct}$, a randomized algorithm that takes as input a secret key $\mathsf{sk} \in \mathcal{K}$ and a message $\mu \in \mathcal{M}$ and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.
- $\mathsf{Eval}(\mathsf{ek}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \to \mathsf{ct}_{\mathsf{out}}$, a deterministic algorithm that takes as input an evaluation key $\mathsf{ek} \in \mathcal{K}$, a function $f : \mathcal{M}^\ell \to \mathcal{M}$, and ℓ ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell \in \mathcal{C}$ and outputs a ciphertext $\mathsf{ct}_{\mathsf{out}} \in \mathcal{C}$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mu$, a deterministic algorithm that takes as input a secret key $\mathsf{sk} \in \mathcal{K}$ and a ciphertext $\mathsf{ct} \in \mathcal{C}$ and outputs a message $\mu \in \mathcal{M}$.

Given a failure probability $\epsilon(\lambda)$ and a function class $\mathcal{F}_\tau$, we require a somewhat homomorphic encryption scheme to satisfy three properties: correctness, semantic security, and compactness.

Correctness. For all parameters $\lambda \in \mathbb{N}$ and $\tau \in \mathbb{N}$, for all functions $f : \mathcal{M}^\ell \to \mathcal{M}$ in the class $\mathcal{F}_\tau$, and for all ℓ messages $\mu_1, \ldots, \mu_\ell \in \mathcal{M}$, we require that:

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_{\mathsf{out}}) \neq f(\mu_1, \ldots, \mu_\ell) \;\middle|\; \begin{array}{rl} \mathsf{sk}, \mathsf{ek} & \leftarrow \mathsf{Gen}(1^\lambda, 1^\tau) \\ \mathsf{ct}_i & \leftarrow \mathsf{Enc}(\mathsf{sk}, \mu_i) \text{ for } i \in [\ell] \\ \mathsf{ct}_{\mathsf{out}} & \leftarrow \mathsf{Eval}(\mathsf{ek}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \end{array}\right] \leq \epsilon(\lambda).$$

Semantic security. For all parameters $\lambda \in \mathbb{N}$ and $\tau \in \mathbb{N}$, for any number of messages $m = \mathrm{poly}(\lambda) \in \mathbb{N}$, and for any two vectors $\mathbf{u}, \mathbf{v} \in \mathcal{M}^m$, their encryptions must be computationally indistinguishable:

$$\begin{aligned} & \left\{ \mathsf{ek}, \mathsf{Enc}(\mathsf{sk}, u_1), \ldots, \mathsf{Enc}(\mathsf{sk}, u_m) \;\middle|\; \mathsf{sk}, \mathsf{ek} \leftarrow \mathsf{Gen}(1^\lambda, 1^\tau) \right\} \\ \overset{c}{\approx} \quad & \left\{ \mathsf{ek}, \mathsf{Enc}(\mathsf{sk}, v_1), \ldots, \mathsf{Enc}(\mathsf{sk}, v_m) \;\middle|\; \mathsf{sk}, \mathsf{ek} \leftarrow \mathsf{Gen}(1^\lambda, 1^\tau) \right\}. \end{aligned}$$

Compactness. There exists a polynomial $p(\cdot)$ such that, for every $\tau \in \mathbb{N}$, there exists a constant $c \in \mathbb{N}$ where, for all parameters $\lambda > c$, for all functions $f : \mathcal{M}^\ell \to \mathcal{M}$ in the class $\mathcal{F}_\tau$, and for all ℓ messages $\mu_1, \ldots, \mu_\ell \in \mathcal{M}$, let:

$$\begin{aligned} \mathsf{sk}, \mathsf{ek} &\leftarrow \mathsf{Gen}(1^\lambda, 1^\tau) \\ \mathsf{ct}_i &\leftarrow \mathsf{Enc}(\mathsf{sk}, \mu_i) \text{ for } i \in [\ell] \\ \mathsf{ct}_{\mathsf{out}} &\leftarrow \mathsf{Eval}(\mathsf{ek}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell). \end{aligned}$$

Then, the bit length of each of the ciphertexts ct_i, for $i \in [\ell]$, and of the ciphertext $\mathsf{ct}_{\mathsf{out}}$ is at most $p(\lambda)$, independent of τ.

2.2 Definition of Linearly Homomorphic Encryption

An encryption scheme is "linearly homomorphic" if it supports additions and scalar multiplications on its ciphertexts. We formalize this notion as follows:

Definition 2.2 (Linearly Homomorphic Encryption)**.** For any modulus $q \geq 2$, a *linearly homomorphic encryption* (LHE) scheme over message space $\mathbb{Z}_q$ is a SHE scheme whose function class $\mathcal{F}_\tau$ contains all affine functions over $\mathbb{Z}_q$ in up to τ variables.

Linearly homomorphic encryption is implied by the decisional Diffie-Hellman (DDH) assumption [ElG85] and the decisional composite residuosity (DCR) assumption [Pai99], among others. We say a LHE scheme has "rate one" if the ratio between the bit length of ciphertexts and the bit length of plaintexts approaches one—that is, each ciphertext requires roughly as many bits to represent as each plaintext. The Damgård-Jurik cryptosystem from DCR is a LHE scheme with rate one: as the message space grows large, the bit length of ciphertexts can be arbitrarily close to the bit length of plaintexts [DJ01].

From DDH, no LHE scheme with rate one is known. However, prior work has constructed LHE from DDH such that, when evaluating the same affine function many times on many ciphertexts, the bit length of a batch of ciphertexts after homomorphic evaluation can be made arbitrarily close to the bit length of plaintexts [DGI+19, BBD+20, BBDP22]. In Section 4.2, we build a new LHE scheme from DDH such that, when evaluating many *different* affine functions on many ciphertexts, the bit length of a batch of ciphertexts after homomorphic evaluation can be made arbitrarily close to the bit length of plaintexts (Lemma 4.4).

2.3 Definition of Sparse LPN

At a high level, the learning-parity-with-noise (LPN) assumption [BFKL93] states that it is computationally hard to solve systems of linear equations over a finite field, when a fraction of the equations have been corrupted with random noise. In the language of codes, this is equivalent to the task of

decoding a linear code, when the codeword has been randomly corrupted in a fraction of locations. The *sparse* learning-parity-with-noise assumption [Ale03,AIK06b,IKOS08,ABW10,DIJL23,RVV24] posits that this problem remains hard, even when the system of linear equations (or, equivalently, the generating matrix of the linear code) is sparse. Other research communities have studied the sparse-LPN problem, which is also referred to as the noisy k-LIN problem or noisy k-XOR in the case of the binary field. Very recently, Jain, Lin, and Saha [JLS24] and Bangachev, Bresler, Tiegel, and Vaikuntanathan [BBTV25] demonstrated that, in some parameter regimes (where the noise rate is larger than required in this work), the hardness of sparse LPN is in fact implied by the near-exponential hardness of LPN.

On their own, the LPN and sparse-LPN assumptions give rise to a handful of cryptographic primitives, including public-key encryption [Ale03, ABW10, DMQN12, KMP14, YZ16], pseudorandom generators with linear stretch and constant locality [AIK06b], cryptographic primitives with constant computational overhead [IKOS08, ADI+17], certain forms of pseudorandom correlation generators [BCGI18], and sublinear-communication multi-party computation using homomorphic secret sharing [DIJL23]. However, in combination with other assumptions (in particular, Diffie-Hellman-like assumptions on groups that admit bilinear maps), LPN and sparse LPN are a crucial ingredient in constructions of indistinguishability obfuscation [JLS21,JLS22,RVV24], which in turn imply an extensive array of cryptographic primitives [SW14, CLTV15, AS16, KNY17, WW24]. One benefit of LPN and sparse LPN is plausible *post-quantum* security: they appear to resist attacks even from quantum algorithms [BLVW19, DJ24].

Defining sparse LPN. Our definition uses the following distributions and sets, parameterized by an error probability $\nu \in (0,1)$, an integer modulus $q \geq 2$, dimensions $m, n \in \mathbb{N}$, and an integer sparsity $k \leq n$:

- *Bernoulli errors.* We let $\mathsf{RandBern}_{\nu,q}$ be the distribution over $\mathbb{F}_q$ that
  - with probability ν, takes on a uniform random value in $\mathbb{F}_q \setminus \{0\}$, and
  - otherwise, takes on the value 0.
- *Sparse matrices.* We let $\mathcal{S}_{k,m,n,q}$ denote the set of matrices in $\mathbb{F}_q^{m \times n}$ whose rows each contain exactly k non-zero entries. Throughout, we refer to $\mathcal{S}_{k,m,n,q}$ as the set of *k-sparse matrices.*
- *Sparse diagonal matrices.* We let $\mathsf{Diag}(\mathcal{S}_{k,m,n,q})$ denote the subset of matrices in $\mathcal{S}_{k,m,n,q}$ for which, in every chunk of $(n+1)$ rows, the values along the "diagonal" must be non-zero. More formally, this corresponds to the matrices in $\mathbb{F}_q^{m \times n}$ where (a) each row must contain k non-zero entries, and (b) for each index $i \in [m]$ where $i \bmod (n+1) \neq 0$, the ith row in the matrix contains a non-zero value in its $(i \bmod (n+1))^{\text{th}}$ entry.

Definition 2.3 (Decisional (n, m, q, δ, k)-sparse LPN)**.** On security parameter $\lambda \in \mathbb{N}$, secret dimension $n = n(\lambda) \in \mathbb{N}$, number of samples $m = m(\lambda) \in \mathbb{N}$, prime modulus $q = q(\lambda) \geq 2$, constant error parameter $\delta \in (0,1)$, and sparsity $k = k(\lambda) \leq n$, the *decision version of (n, m, q, δ, k)-sparse LPN* asserts that the following distributions are computationally indistinguishable:

$$\left\{ (\mathbf{A}, \mathbf{As}+\mathbf{e}) : \begin{array}{r} \mathbf{A} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,m,n,q} \\ \mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \\ \mathbf{e} \leftarrow \mathsf{RandBern}_{n^{-\delta},q}^m \end{array} \right\} \overset{c}{\approx} \left\{ (\mathbf{A}, \mathbf{u}) : \begin{array}{r} \mathbf{A} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,m,n,q} \\ \mathbf{u} \xleftarrow{\mathrm{R}} \mathbb{F}_q^m \end{array} \right\}.$$

In this work, we rely on the sparse-LPN assumption with a secret dimension n that is polynomial in λ, any modulus q that is at most exponential in λ, any constant error parameter $\delta \in (0,1)$, and any sparsity k that is a super-constant function in λ. In this setting, for any number of samples $m = \mathrm{poly}(\lambda)$, the sparse-LPN assumption plausibly holds against polynomial-time algorithms. We write "(n, q, δ, k)-*sparse LPN*" to denote the assumption that decisional (n, m, q, δ, k)-sparse LPN is true for any $m = \mathrm{poly}(\lambda)$.

Key-dependent-message security. Looking ahead, for a certain flavor of key-dependent-message (KDM) security [BHHO08] to hold with sparse LPN, we will need the $\mathbf{A}$-matrix to be a sparse *diagonal* matrix. Dao, Ishai, Jain, and Lin show that, if sparse LPN is hard, then LPN with sparse diagonal matrices is hard and satisfies KDM security for linear functions [DIJL23, Lemma 4.1]. We recall this statement, which is proved via a reduction that polynomially increases the number of samples m, roughly doubles the sparsity k, increases the error parameter δ by $o(1)$, and requires the modulus q to be a prime ≥ 3:

Lemma 2.4 (KDM Security of Sparse Diagonal LPN [DIJL23, Lemma 4.1])**.** *Under $(n, q, \delta', (k+1)/2)$-sparse LPN with a prime modulus $q \geq 3$, error rate $\delta' = \delta - 1/\log n$, and super-constant, odd sparsity $k = o(\sqrt{n})$, for any $m = \mathrm{poly}(\lambda)$ messages $\mu_1, \ldots, \mu_m \in \mathbb{F}_q$, it holds that:*

$$\left\{ \left\{ \mathbf{A}_i, \mathbf{A}_i\mathbf{s} + \mathbf{e}_i + \mu_i \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \right\}_{i\in[m]} : \begin{array}{c} \mathbf{A}_i \xleftarrow{\mathrm{R}} \mathsf{Diag}(\mathcal{S}_{k,n+1,n,q}) \text{ for } i \in [m] \\ \mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \\ \mathbf{e}_i \leftarrow \mathsf{RandBern}^{n+1}_{n^{-\delta},q} \text{ for } i \in [m] \end{array} \right\}$$
$$\overset{c}{\approx} \left\{ \{\mathbf{A}_i, \mathbf{u}_i\}_{i\in[m]} : \begin{array}{c} \mathbf{A}_i \xleftarrow{\mathrm{R}} \mathsf{Diag}(\mathcal{S}_{k,n+1,n,q}) \text{ for } i \in [m] \\ \mathbf{u}_i \xleftarrow{\mathrm{R}} \mathbb{F}_q^{n+1} \text{ for } i \in [m] \end{array} \right\}.$$

For completeness, we give a proof of Lemma 2.4 in Appendix A.

3 Somewhat Homomorphic Encryption from Sparse LPN and Linearly Homomorphic Encryption

We now present a construction of somewhat homomorphic encryption from sparse LPN and a linearly homomorphic encryption scheme. On security parameter $\lambda \in \mathbb{N}$ and functionality parameter $\tau = \mathrm{poly}(\lambda) \in \mathbb{N}$, our scheme can homomorphically evaluate multivariate polynomials of total degree $\log \tau / \log\log \tau$ with up to τ monomials. Our scheme has an arbitrarily small, inverse-polynomial probability of correctness errors, which can be driven down to negligible via parallel repetitions (Remark 3.4).

Our construction proves the following theorem statement:

Theorem 3.1 (SHE from Sparse LPN and LHE)**.** *On security parameter $\lambda \in \mathbb{N}$, functionality parameter $\tau = \mathrm{poly}(\lambda) \in \mathbb{N}$, and constant correctness parameter $c \in \mathbb{N}$, assume that $(n, q, \delta', (k+1)/2)$-sparse LPN holds with a prime modulus $q \geq 3$ of size at most $\exp(\lambda)$, error parameter $\delta' = \delta - 1/\log n$, secret dimension $n \geq \tau^{2/\delta} \cdot \lambda^{c/\delta}$, and super-constant, odd sparsity $k \leq \sqrt{\log \tau} - 1$. Let Ψ be a semantically-secure LHE scheme with message space $\mathbb{F}_q$ and $\epsilon_{\mathsf{LHE}}(\lambda)$ probability of correctness failures. Then, Construction 3.2 parameterized by (n, q, δ, k, Ψ) is a semantically-secure SHE scheme with message space $\mathbb{F}_q$ and the following properties:*

Construction 3.2 (SHE from sparse LPN and LHE)**.** Parameterized by LPN parameters (n, q, δ, k), where $n = n(\lambda)$, $q = q(\lambda)$, $\delta \in (0,1)$, and $k = k(\lambda)$, and a linearly homomorphic encryption scheme $(\mathsf{LHE.Gen}, \mathsf{LHE.Enc}, \mathsf{LHE.Eval}, \mathsf{LHE.Dec})$ with message space $\mathbb{F}_q$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$. Let $\ell := n + 1$. Define sets $\mathcal{S}_{k,1,n,q}$ and $\mathsf{Diag}(\mathcal{S}_{k,\ell,n,q})$ and distribution $\mathsf{RandBern}_{n^{-\delta},q}$ as in Section 2.3.

$\mathsf{Gen}(1^\lambda, 1^\tau) \to (\mathsf{sk}, \mathsf{ek})$

- Let $\mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^\ell)$.
- Let $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{F}_q^n$ and $\tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal$.
- Let $\mathbf{t} \xleftarrow{\mathrm{R}} \mathbb{F}_q^n$ and $\tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal$.
- $\forall\, i \in [\ell]$, $\mathbf{C}_{\mathsf{ek},i} \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i)$.
- $\forall\, i \in [\ell]$, $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \tilde{s}_i)$.
- Output $\mathsf{sk} := (\mathbf{s}, \mathbf{t}, \mathsf{sk}_{\mathsf{LHE}})$ and $\mathsf{ek} := (\mathsf{ek}_{\mathsf{LHE}}, \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},\ell}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.

$\mathsf{GSWEnc}(\mathbf{s} \in \mathbb{F}_q^n, \mu \in \mathbb{F}_q) \to \mathbf{C} \in \mathbb{F}_q^{\ell \times \ell}$

- Sample $\mathbf{A} \xleftarrow{\mathrm{R}} \mathsf{Diag}(\mathcal{S}_{k,\ell,n,q})$.
- Sample $\mathbf{e} \leftarrow \mathsf{RandBern}_{n^{-\delta},q}^\ell$.
- Let $\tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^\ell$.
- Compute $\mathbf{b} := \mathbf{A}\mathbf{s} + \mathbf{e} + \mu \cdot \tilde{\mathbf{s}}$.
- Output $\mathbf{C} := \begin{bmatrix} \mathbf{A} & \| & \mathbf{b} \end{bmatrix}$.

$\mathsf{Enc}(\mathsf{sk}, \mu \in \mathbb{F}_q) \to \mathbf{c} \in \mathbb{F}_q^\ell$

- Parse $\mathbf{t} \in \mathbb{F}_q^n$ from sk.
- Sample $\mathbf{a} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,1,n,q}$ and $e \leftarrow \mathsf{RandBern}_{n^{-\delta},q}$.
- Output $\mathbf{c} := \begin{bmatrix} \mathbf{a} & \| & \langle \mathbf{a}, \mathbf{t} \rangle + e + \mu \end{bmatrix}$.

$\mathsf{Expand}(\mathsf{ek}, \mathbf{c} \in \mathbb{F}_q^\ell) \to \mathbf{C} \in \mathbb{F}_q^{\ell \times \ell}$

- Parse $(\mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1})$ from ek.
- Output $\mathbf{C} := \sum_{i \in [\ell]} c_i \cdot \mathbf{C}_{\mathsf{ek},i}$.

$\mathsf{Add}(\mathbf{C}_1 \in \mathbb{F}_q^{\ell \times \ell}, \mathbf{C}_2 \in \mathbb{F}_q^{\ell \times \ell}) \to \mathbf{C} \in \mathbb{F}_q^{\ell \times \ell}$

- Output $\mathbf{C} := \mathbf{C}_1 + \mathbf{C}_2$.

$\mathsf{Mul}(\mathbf{C}_1 \in \mathbb{F}_q^{\ell \times \ell}, \mathbf{C}_2 \in \mathbb{F}_q^{\ell \times \ell}) \to \mathbf{C} \in \mathbb{F}_q^{\ell \times \ell}$

- Output $\mathbf{C} := \mathbf{C}_1 \cdot \mathbf{C}_2$.

$\mathsf{Compact}(\mathsf{ek}, \mathbf{C} \in \mathbb{F}_q^{\ell \times \ell}) \to \mathsf{ct} \in \mathcal{C}$

- Parse $(\mathsf{ek}_{\mathsf{LHE}}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ from ek.
- Let $\mathbf{c} \in \mathbb{F}_q^\ell$ be the last row of $\mathbf{C}$.
- Let $f(x_1, \ldots, x_\ell) = \sum_{i \in [\ell]} c_i x_i$.
- Output $\mathsf{ct} := \mathsf{LHE.Eval}(\mathsf{ek}_{\mathsf{LHE}}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct} \in \mathcal{C}) \to \mu \in \mathbb{F}_q$

- Parse $\mathsf{sk}_{\mathsf{LHE}} \in \mathcal{K}$ from sk.
- Output $\mu := \mathsf{LHE.Dec}(\mathsf{sk}_{\mathsf{LHE}}, \mathsf{ct})$.

Figure 2: Construction of somewhat homomorphic encryption from sparse LPN and LHE.

- Correctness: *decryption succeeds with probability* $1 - \lambda^{-c} - \epsilon_{\mathsf{LHE}}(\lambda)$.
- Homomorphism: *the function class* $\mathcal{F}_\tau$ *contains all multivariate polynomials over* $\mathbb{F}_q$ *with total degree up to* $\log \tau / \log\log \tau$ *and up to* τ *monomials.*

Put differently, our new somewhat homomorphic encryption can perform up to $\log \tau / \log\log \tau$ homomorphic multiplications, followed by τ homomorphic additions. By instantiating Theorem 3.1 with a standard LHE scheme (e.g., El Gamal [ElG85], Paillier [Pai99], Damgård-Jurik [DJ01]), we obtain the first constructions of somewhat homomorphic encryption for the function class $\mathcal{F}_\tau$ from sparse LPN and either DDH or DCR. On their own, none of these assumptions are known to imply somewhat homomorphic encryption for polynomials with bounded degree and a bounded number of monomials.

As a corollary, if there exists a plausibly post-quantum LHE scheme that is based on non-lattice assumptions, then Theorem 3.1 gives the first plausibly post-quantum somewhat homomorphic encryption scheme that is based on non-lattice assumptions. In other words, Theorem 3.1 reduces

the task of building post-quantum SHE without lattices to the simpler task of building post-quantum LHE without lattices.

Construction. We present our somewhat homomorphic encryption scheme from sparse LPN and a LHE scheme in Construction 3.2. To implement the Eval routine (defined in Section 2.1), Construction 3.2 uses four polynomial-time algorithms: Expand, Add, Mul, and $\mathsf{Compact}$. The Expand algorithm transforms ciphertexts output by Enc into ones that can be passed as inputs to Add and Mul (arbitrarily many times). The Add and Mul algorithms apply homomorphic addition and multiplication gates to ciphertexts. We do not require their outputs to be compact — instead, the bit length of ciphertexts may grow as operations are performed. Later on, after all homomorphic operations have been completed, each ciphertext is passed to the $\mathsf{Compact}$ algorithm, which outputs a fixed-length ciphertext encrypting the same message but which no longer has (many of its) homomorphic capabilities.

We formally analyze Construction 3.2's correctness, security, and compactness in Appendix B.1. At a high level, our argument works as follows:

1. *Correctness*: We demonstrate that the ciphertext matrices output by $\mathsf{Enc}(\mathsf{sk}, \cdot)$ and then passed to $\mathsf{Expand}(\mathsf{ek}, \cdot)$ have the following special structure: the "extended" secret key $\begin{bmatrix}-\mathbf{s} & 1\end{bmatrix}^\intercal$ is an "approximate" eigenvector of the ciphertext matrix, with the encrypted message μ as the corresponding eigenvalue. Then, as long as each ciphertext matrix remains sufficiently sparse, we can add and multiply these matrices — which has the effect of adding and multiplying the underlying messages — without the error rate growing too large. As in GSW [BV14], the sparsity and error rate of our ciphertexts grow additively with every Add and multiplicatively with every Mul. As long as the error growth is not too large, the algorithms $\mathsf{Compact}(\mathsf{ek}, \cdot)$ and $\mathsf{Dec}(\mathsf{sk}, \cdot)$ will recover the correct output of the computation.

2. *Security*: The scheme is semantically secure, assuming sparse LPN and the semantic security of the underlying LHE scheme. This is because all fresh ciphertexts are masked by a sparse-LPN sample with secret key $\mathbf{t} \in \mathbb{F}_q^n$, and the evaluation key consists of (a) the entries of secret key $\mathbf{t} \in \mathbb{F}_q^n$, masked by sparse-LPN samples with secret key $\mathbf{s} \in \mathbb{F}_q^n$, and (b) LHE encryptions of the entries of secret key $\mathbf{s}$. Since the evaluation key uses a "chain" of encryptions of one key under the next, we do not need a circular security assumption.

3. *Compactness*: The ciphertexts output by Enc and by $\mathsf{Compact}$ have a bit length that is a fixed polynomial in λ, independent of the number of computations performed on them. This is because fresh ciphertexts are sparse vectors, which require

$$\begin{aligned} O(k \cdot \log n \cdot \log q) &= O(\sqrt{\log \tau} \cdot \log(\tau^{2/\delta} \cdot \lambda^{c/\delta}) \cdot \log q) \\ &= O((\log \lambda)^3 \cdot \log q) = \mathrm{poly}(\lambda) \end{aligned}$$

   bits to represent, since we know that $\tau < \lambda^{\log \lambda}$ and thus $\log \tau < (\log \lambda)^2$. Evaluated ciphertexts are ciphertexts from the underlying LHE scheme, which, by definition, must be compact.

In the rest of this section, we discuss the efficiency of Construction 3.2 and variations of it.

Remark 3.3 (Efficiency of Construction 3.2). On parameters (n, q, δ, k) and given a LHE scheme with ciphertext space $\mathcal{C}$, Construction 3.2 is a SHE scheme with the following efficiency:

- Each ciphertext output by Enc is a $(k+1)$-sparse vector in $\mathbb{F}_q^{n+1}$. Such a ciphertext can be represented using $(k \log n + 1) \cdot \log q$ bits.
- When evaluating a polynomial of total degree d with M monomials,
  - each call to Expand requires $(k+1)^{2d}$ multiplications and additions in $\mathbb{F}_q$,
  - each call to Mul requires $(k+1)^{2d}$ multiplications and additions in $\mathbb{F}_q$,
  - each call to Add requires $(k+1)^{2d}$ additions in $\mathbb{F}_q$, and
  - the final call to $\mathsf{Compact}$ requires $M \cdot (k+1)^{2d}$ homomorphic additions under LHE encryption.
- Each ciphertext output by $\mathsf{Compact}$ is a value in $\mathcal{C}$, the ciphertext space of the linearly homomorphic encryption scheme.

These compute costs do not scale with the secret dimension n, even though ciphertexts in the intermediate phase of the computation are $(n+1)$-by-$(n+1)$ matrices. This is because we can take advantage of the sparsity in these matrices to prune the computation and perform only those operations that affect the final output [DIJL23, Remark 5.4]. We describe this optimization in more detail in Appendix B.2.

Remark 3.4 (SHE with $\mathrm{negl}(\lambda)$ probability of correctness failures). We can boost our SHE scheme to one that has a negligibly small probability of correctness failures by performing λ instances of Construction 3.2 (with independent randomness) and outputting the majority of the decryptions, in addition to using a LHE scheme with perfect correctness. When evaluating the function class $\mathcal{F}_\tau$, this transformation succeeds as long as the following conditions on the sparsity k and the secret dimension n hold: $k \leq \sqrt{\log \tau} - 1$ and $n = \Omega(\tau^{2/\delta})$. This construction is $\lambda\times$ less efficient.

Remark 3.5 (Performing homomorphic operations in a different order). While Theorem 3.1 describes the class of computations supported by our SHE scheme as consisting of $\log \tau / \log\log \tau$ multiplications followed by τ additions, the construction supports performing additions and multiplications interleaved in any order. Changing the order of operations affects the sparsity growth and the error growth in the scheme's ciphertexts, and thus requires a specialized accounting to determine how many operations can be performed (as per Appendix B.1). As in the GSW scheme [BV14], the error-rate of our ciphertexts grows *asymmetrically* with every Mul.

Remark 3.6 (Using LHE with homomorphism over other moduli). Construction 3.2 natively supports homomorphism over any $\mathbb{F}_q$, where $q \geq 3$ is a prime and $\mathbb{F}_q$ is the LHE scheme's message space. We can decouple the sparse-LPN modulus q from the LHE scheme's message space $\mathbb{Z}_{q'}$ by instead running the linear function evaluation within $\mathsf{Compact}$ "over the integers," as long as q' is sufficiently large compared to n and q (in particular, $q' > (n+1) \cdot q^2$). We give one example of this in Section 4.2.

Remark 3.7 (SHE from sparse LPN and QR?). Standard encryption schemes from the quadratic residuosity assumption (e.g., Goldwasser-Micali [GM84]) are linearly homomorphic over $\mathbb{F}_2$. However, we cannot directly substitute them into Construction 3.2 because the KDM-security proof of sparse LPN, on which our security hinges, only works for a prime modulus $q \geq 3$ (Lemma 2.4). To overcome this barrier, Dao, Ishai, Jain, and Lin sketch a variant of the sparse-LPN assumption that can be proved KDM-secure on modulus $q = 2$: in this variant, every sparse-LPN sample is equally likely to be either k or $(k+1)$-sparse [DIJL23, Remark 4.2]. Under this modified assumption, we can construct SHE using QR or, more broadly, any LHE scheme with message space $\mathbb{F}_2$.

Remark 3.8 (Increased homomorphism from constant-sparse LPN). One way to boost the homomorphism of our SHE schemes would be to rely on the stronger assumption that LPN with *constant* sparsity is hard, which plausibly holds in certain parameter regimes, when the number of samples released is small enough and when sampling the $\mathbf{A}$-matrices from carefully crafted distributions [ADI+17, AK19]. As with Dao, Ishai, Jain, and Lin's homomorphic secret-sharing scheme [DIJL23], this stronger assumption could improve our SHE scheme's homomorphism to support polynomials of total degree $O(\log \lambda)$ with $\mathrm{poly}(\lambda)$ monomials, but also appears to require the encryption algorithm to be stateful to sample the $\mathbf{A}$-matrices from a safe distribution. The critical issue is that the encryptor must not release sparse-LPN samples with too similar $\mathbf{A}$-matrices, which otherwise might occur with non-negligible probability.

Remark 3.9 (Scalar multiplications are "free"). In our schemes, multiplying a ciphertext by a scalar does not incur any growth in the error rate or in the sparsity of a ciphertext, which means that the multiplication-by-scalar operation does not increase the probability of a correctness error. In contrast, lattice-based somewhat homomorphic encryption schemes generally must bound the number and magnitude of scalar multiplications performed on a single ciphertext. (This difference stems from the error distributions used: with LWE, every linear relation has a small amount of error, which grows out of control if multiplied by a too large scalar. With sparse LPN, errors are all-or-nothing and random, so their distribution does not change when multiplied by a scalar.)

4 Optimizing the Bit Length of Ciphertexts

In this section, we turn our attention to the efficiency of our new somewhat homomorphic encryption. We present a sequence of optimizations that gives SHE from sparse LPN and DDH with short ciphertexts, when encrypting a batch of many messages at once and homomorphically evaluating a batch of many polynomials at once. We say that this scheme achieves "batch rate" one.

To capture this notion of "batch encryption" and "batch evaluation," we introduce a slightly more general syntax for somewhat homomorphic encryption in Appendix C.1. We give an optimization that shrinks the size of ciphertexts before homomorphic evaluation in Section 4.1, and one that shrinks the size of ciphertexts after homomorphic evaluation in Section 4.2. Together, these techniques prove the following theorem (Section 4.3):

Theorem 4.1 (SHE with Short Ciphertexts from Sparse LPN and DDH)**.** *On security parameter $\lambda \in \mathbb{N}$, functionality parameter $\tau = \mathrm{poly}(\lambda) \in \mathbb{N}$, constant correctness parameter $c \in \mathbb{N}$, and any prime modulus $q \geq 3$ of size at most $\mathrm{poly}(\lambda)$, assume that $(n, q, \delta', (k+1)/2)$-sparse LPN holds with any error parameter $\delta' = \delta - 1/\log n$, super-constant, odd sparsity $k \leq \sqrt{\log \tau} - 1$, and secret dimension $n \geq \tau^{2/\delta} \cdot \lambda^{c/\delta}$. Then, assuming DDH, there exists a semantically-secure SHE scheme with message space $\mathbb{F}_q$ and with the following properties:*

- Correctness: *decryption succeeds with probability $1 - 2\lambda^{-c}$.*
- Homomorphism: *the function class $\mathcal{F}_\tau$ contains all multivariate polynomials over $\mathbb{F}_q$ with total degree up to $\log \tau / \log \log \tau$, and up to τ monomials.*
- Efficiency: *when homomorphically evaluating a batch of $t = \mathrm{poly}(\lambda)$ polynomials in $\mathbb{F}_q^m \to \mathbb{F}_q$, on a batch of tm inputs,*
  - *The evaluation key consists of $t^2 \cdot \mathrm{poly}(\lambda)$ bits.*

– *All tm input ciphertexts together consist of $tm \log q + m \cdot \mathrm{polylog}(\lambda)$ bits.*

– *All t output ciphertexts together consist of $t \log q + \mathrm{poly}(\lambda)$ bits.*

In this scheme, as the batch size t grows large, the bit length of all tm input ciphertexts together can be arbitrarily close to $tm \cdot \log q$ (i.e., the bit length of all input plaintexts together) and the bit length of all t output ciphertexts together can be arbitrarily close to $t \cdot \log q$ (i.e., the bit length of all output plaintexts together). Finally, by evaluating sufficiently many batches of polynomials, we can also amortize away the cost of transmitting the one-time evaluation key.

Remark 4.2 (SHE with Short Ciphertexts from Sparse LPN and DCR). An alternate construction of SHE, which can have rate-one ciphertexts when evaluating a single polynomial over a sufficiently large modulus, is possible from sparse LPN and DCR: this requires (a) extending the KDM-security proof (Lemma 2.4) to allow for sparse LPN over a modulus that is a power of a product of two λ-bit primes, (b) instantiating Construction 3.2 with the Damgård-Jurik LHE scheme [DJ01] that has rate one, and (c) applying the technique from Section 4.1.

4.1 Shrinking the Bit Length of Fresh Ciphertexts

First, we shrink the size of fresh ciphertexts in our scheme down to *one* element in $\mathbb{F}_q$, instead of a $(k+1)$-sparse vector in $\mathbb{F}_q^{n+1}$ — eliminating the dependence on the sparsity k and the secret dimension n in our ciphertexts' bit lengths. To do so, we apply a standard insight from the lattice literature [PW08, Lemma 6.2][PVW08, Lemma 7.3] [Pie12]: the $\mathbf{a}$-component in sparse-LPN ciphertexts is a random sparse vector (or matrix) that does *not* depend on the message being encrypted. As a result, we can take this $\mathbf{a}$-component to be either pseudorandom or fixed across many encryptions, at only a polynomial loss in security. In more detail:

In the random-oracle model. Given a random oracle, we can specify the $\mathbf{a}$-component of each ciphertext with a short seed for a pseudorandom generator.

Without a random oracle. We can fix and re-use the $\mathbf{a}$-matrix in sparse LPN to build polynomially many ciphertexts, as long as each ciphertext uses a random, independently sampled secret key $\mathbf{s}$ and Bernoulli error e. This optimization lets us communicate the $\mathbf{a}$-vector only once, and amortize this cost over many ciphertexts. We formalize this technique in Lemma C.2, proved via a hybrid argument in Appendix C.2.

Remark 4.3 (Fixing the $\mathbf{A}$-matrix in GSW-style ciphertexts). By further extending Lemma C.2 to handle sparse diagonal LPN, we could apply the same technique to shrink each of the $(n+1)$ sparse-LPN ciphertexts included in the evaluation key from a $(k+1)$-sparse matrix in $\mathbb{F}_q^{(n+1)\times(n+1)}$ down to a vector in $\mathbb{F}_q^{n+1}$. Intuitively, this optimization works because our SHE scheme reveals the $\mathbf{A}$-matrix in the clear as part of each ciphertext. Thus, an adversary can on its own build polynomially many sparse-LPN samples with the same $\mathbf{A}$-matrix but independent secrets and errors. By the scheme's security, it must be that obtaining these sparse-LPN samples with the same $\mathbf{A}$-matrix (but different secrets and errors) does not noticeably help in breaking the original scheme.

4.2 Shrinking the Bit Length of Evaluated Ciphertexts

Next, we shrink the size of ciphertexts *after* homomorphic evaluation. To do so, we build a secret-key linearly homomorphic encryption scheme from DDH that (1) supports an arbitrarily small message space $\mathbb{Z}_q$ and (2) has the following efficiency: when homomorphically evaluating a batch

of many, distinct affine functions on ciphertexts, each output ciphertext consists of roughly $\log q$ bits. Looking ahead, each of these properties of our new LHE scheme is crucial in our construction of SHE with short ciphertexts (Theorem 4.1): first, to evaluate polynomials over $\mathbb{F}_q$ on any prime modulus $q \geq 3$, we need LHE with message space $\mathbb{F}_q$. (Otherwise, we would need to embed the computation into the larger field over which our LHE scheme is homomorphic, e.g., the El-Gamal message space, causing big losses in rate.) Second, we need LHE with short output ciphertexts when homomorphically evaluating a batch of many *distinct* affine functions, because running $\mathsf{Compact}$ on distinct sparse-LPN ciphertexts requires evaluating distinct affine functions (each determined by the ciphertexts' $\mathbf{A}$-component).

Lemma 4.4 (LHE from DDH over $\mathbb{Z}_q$)**.** *On security parameter $\lambda \in \mathbb{N}$, assume that DDH is hard in a group with elements of bit length λ_{DDH}. Then, given any functionality parameter $\tau \in \mathbb{N}$, modulus $q \in \mathbb{N}$, and packing parameter $t \in \mathbb{N}$ that are each at most $\mathrm{poly}(\lambda)$ and given any constant correctness parameter $c \in \mathbb{N}$, there exists a semantically-secure LHE scheme from DDH with message space $\mathbb{Z}_q$ and the following properties:*

- Correctness: *Decryption succeeds with probability $1 - \lambda^{-c}$.*
- Efficiency: *The evaluation key consists of $\mathrm{poly}(\lambda)$ bits. Each fresh ciphertext produced by $\mathsf{LHE.Enc}$ consists of $t \cdot (t+1) \cdot \lambda_{\mathsf{DDH}}$ bits. When homomorphically evaluating a batch of t affine functions in $\mathbb{Z}_q^\tau \to \mathbb{Z}_q$, the t output ciphertexts together consist of $t \log q + \lambda_{\mathsf{DDH}}$ bits.*

We present our linearly homomorphic encryption scheme in Construction C.3; our scheme is a variant of standard El-Gamal encryption [ElG85] with two twists. First, since El-Gamal is linearly homomorphic over an exponentially large message space (i.e., $\mathbb{Z}_p$ on a λ-bit prime p), whereas we require homomorphism over the much smaller space $\mathbb{Z}_q$, we think of homomorphic operations as performed "over the integers." When evaluating any affine function $f : \mathbb{Z}_q^\tau \to \mathbb{Z}_q$, the maximal value that f can output over the integers is bounded from above by $B = q^2 \cdot (\tau + 1)$. So, whenever $p > B$ (which happens naturally as p is exponential in λ, whereas B is at most polynomial in λ), lifting f to $\mathbb{Z}_p$ (i.e., lifting each of its coefficients and inputs from $\mathbb{Z}_q$ to $\mathbb{Z}_p$) and evaluating it within the El-Gamal scheme's message space produces the same result as computing f over the integers. Roughly speaking, our encryption scheme evaluates f over $\mathbb{Z}_p$ and takes the result mod q, which recovers the evaluation of f over $\mathbb{Z}_q$.

Second, inspired by existing LHE schemes from LWE [dCHI+22] and from DDH [DGI+19, BBD+20, BBDP22], we design a "distributed discrete logarithm" protocol that lets us shrink each DDH-ciphertext down to just $\log q$ bits, along with a preamble of λ_{DDH} bits that can be amortized over arbitrarily many ciphertexts. To achieve this, we encrypt vectors of t values in $\mathbb{Z}_q$ at once. Then, given τ such ciphertexts encrypting the vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\tau)}$, each in $\mathbb{Z}_q^t$, and given t affine functions $f_1, \ldots, f_t$, each mapping $\mathbb{Z}_q^\tau \to \mathbb{Z}_q$, we give a procedure that produces a batch of ciphertexts encrypting the t evaluations $v_1 = f_1(x_1^{(1)}, \ldots, x_1^{(\tau)}), \ldots, v_t = f_t(x_t^{(1)}, \ldots, x_t^{(\tau)})$, each in $\mathbb{Z}_q$.

Our $\mathsf{LHE.Eval}$ algorithm works as follows: given an order-p generator g of a DDH-group $\mathbb{G}$ and a secret-key $(z_1, \ldots, z_t) \in \mathbb{Z}_p^t$, we encrypt the message vector $(x_1^{(i)}, \ldots, x_t^{(i)}) \in \mathbb{Z}_q^t$, for $i \in [\tau]$, hidden

along the diagonal of a ciphertext matrix:

$$\mathsf{ct}^{(i)} = \begin{pmatrix} g^{r_1} & g^{r_1 z_1 + x_1^{(i)}} & g^{r_1 z_2} & \dots & g^{r_1 z_t} \\ g^{r_2} & g^{r_2 z_1} & g^{r_2 z_2 + x_2^{(i)}} & \dots & g^{r_2 z_t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g^{r_t} & g^{r_t z_1} & g^{r_t z_2} & \dots & g^{r_t z_t + x_t^{(i)}} \end{pmatrix}.$$

Now, given τ such ciphertext matrices $\mathsf{ct}^{(1)}, \dots, \mathsf{ct}^{(\tau)}$, along with t affine functions $f_1, \dots, f_t : \mathbb{Z}_q^\tau \to \mathbb{Z}_q$, our $\mathsf{LHE.Eval}$ algorithm:

1. Exponentiates the jth row of $\mathsf{ct}^{(i)}$ by the coefficient on the variable "x_i" in function f_j. This step effectively multiplies the encrypted values along each ciphertext's diagonal by the matching coefficient in each function, while keeping the "random preambles" (e.g., g^{r_1}) identical across each row.

2. Computes the matrix $\mathsf{ct}' \in \mathbb{G}^{t \times (t+1)}$ that consists of the element-wise product of all τ ciphertext matrices, and multiplies $g^{f_j(0,\dots,0)}$ into the jth entry of the diagonal. This step effectively produces a matrix with the evaluation of the jth function stored in the jth entry of the diagonal, while keeping the random preambles identical across each row.

3. Computes the vector $\mathsf{ct}'' \in \mathbb{G}^{t+1}$, whose j^{th} entry is the product of all entries in the jth column of ct'. This step effectively produces a vector with the evaluation of the jth function stored in the $(j+1)^{\text{th}}$ entry, and where each entry has the same random preamble.

At the end of this procedure, the ciphertext ct'' has the following form:

$$\mathsf{ct}'' = \begin{pmatrix} g^r & g^{rz_1 + qe_1 + v_1} & g^{rz_2 + qe_2 + v_2} & \dots & g^{rz_t + qe_t + v_t} \end{pmatrix},$$

where $v_1, \dots, v_t \in \mathbb{Z}_q$ are the outputs of the t affine functions over $\mathbb{Z}_q$ applied to the encrypted values, $r \in \mathbb{Z}_p$ is joint randomness, and the terms $e_1, \dots, e_t \in \{0, \dots, B/q\}$ are accumulated by performing $\mathbb{Z}_q$-computations over the integers.

In a final step, we have $\mathsf{LHE.Eval}$ compress the ciphertext ct'' down to just $t \log q + \lambda_{\mathsf{DDH}}$ bits. To do so, we use a PRF and a shared random seed seed, that meets the following properties with high probability:

1. for each $\ell \in [t]$, for *any* extra term $e \in \{0, \dots, B/q\}$, and for *any* output value $\mu \in \mathbb{Z}_q$, it holds that $\mathsf{PRF}(\mathsf{seed}, g^{rz_\ell + qe + \mu}) \neq 0$, and

2. for each $\ell \in [t]$ and for each output value $\mu \in \mathbb{Z}_q$, let $\mathsf{pad}_{\ell,\mu}$ be the smallest integer such that $\mathsf{PRF}(\mathsf{seed}, g^{rz_\ell + q\mathsf{pad}_{\ell,\mu} + \mu}) = 0$. Then, for each $\ell \in [t]$, each of the values of $\mathsf{pad}_{\ell,\mu}$ are distinct and do not fall within ± 1 of each other.

Now, for each $\ell \in [t]$ and for every output value $\mu \in \mathbb{Z}_q$, the $\mathsf{LHE.Eval}$ algorithm can iteratively evaluate PRF on successive group elements (each offset by g^q) starting at the point $g^{rz_\ell + qe_\ell + v_\ell + \mu}$, until it finds a group element where the PRF evaluates to 0. The number of PRF evaluations here must be

$$k_{\ell,\mu} = \mathsf{pad}_{\ell, v_\ell + \mu \bmod q} - e_\ell \pm 1.$$

Finally, for each $\ell \in [t]$, $\mathsf{LHE.Eval}$ sorts the values $(k_{\ell,0}, \ldots, k_{\ell,q-1})$, and computes the rank of $k_{\ell,0}$ in this list — which is just an integer in $\mathbb{Z}_q$. $\mathsf{LHE.Eval}$ outputs a compressed ciphertext consisting of the term g^r and each of these t ranks in $\mathbb{Z}_q$.

Given such a compressed ciphertext, along with the secret key $(z_1, \ldots, z_t)$ and the seed, the $\mathsf{LHE.Dec}$ algorithm can recover each $\mathsf{pad}_{\ell,\mu}$ for each message $\mu \in \mathbb{Z}_q$ and index $\ell \in [t]$: it suffices to compute $(g^r)^{z_\ell} \cdot g^\mu$, and to evaluate the PRF on successive group elements (offset by g^q) until hitting one that evaluates to 0. By sorting the values $(\mathsf{pad}_{\ell,0}, \ldots, \mathsf{pad}_{\ell,q-1})$ and learning their relative ranks, $\mathsf{LHE.Dec}$ can exactly recover each v_ℓ. We formally show that this LHE scheme, described in Construction C.3, is correct, secure, and runs in polynomial time in Appendix C.3.

4.3 Proof Sketch for Theorem 4.1

We describe the somewhat homomorphic encryption scheme that proves Theorem 4.1 in Appendix C.4. Here, we give a high-level overview of the scheme.

Our scheme is governed by two new parameters: $t_{\mathsf{Enc}} \in \mathbb{N}$, which determines how many messages can be "packed" into a single input ciphertext, and $t_{\mathsf{Eval}} \in \mathbb{N}$, which determines how many messages can be "packed" into a single output ciphertext. Our scheme uses t_{Enc} sparse-LPN secrets $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(t_{\mathsf{Enc}})} \in \mathbb{F}_q^n$. For each of these t_{Enc} sparse-LPN secrets, we publish an evaluation key (as described in Section 3) that lets us switch a Regev-style encryption under key $\mathbf{t}^{(i)}$ into a GSW-style encryption under a common sparse-LPN secret $\mathbf{s} \in \mathbb{F}_q^n$. In addition, we publish a redundant encryption of each entry of the sparse-LPN secret $\mathbf{s}$ using our packed-El-Gamal LHE scheme, with packing parameter t_{Eval}.

Then, to encrypt a batch of t_{Enc} messages at once, our encryption scheme can sample a single $\mathbf{a}$-vector, and encrypt each message under the same $\mathbf{a}$-vector using a different one of the $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(t_{\mathsf{Enc}})}$ secret keys (per the optimization in Section 4.1). This produces a "packed" ciphertext encrypting all t_{Enc} messages at once, as a $(k + t_{\mathsf{Enc}})$-sparse vector in $\mathbb{F}_q^{n+t_{\mathsf{Enc}}}$. Our homomorphic evaluation routine can unpack these ciphertexts, run the Expand algorithm on each of them to transform them into a GSW-style encryption under secret key $\mathbf{s}$, and perform homomorphic additions and multiplications as usual. Finally, our compaction routine uses the packed-El-Gamal LHE scheme (from Section 4.2) to produce a "packed" output ciphertext that encrypts a batch of t_{Eval} polynomial evaluations at once. Given the parameter $t \in \mathbb{N}$ in the theorem statement, setting $t_{\mathsf{Enc}} := t$ and $t_{\mathsf{Eval}} := t$ proves Theorem 4.1.

5 Open Question: Can We Bootstrap?

We leave open the question of whether it is possible to bootstrap our somewhat homomorphic schemes to build fully homomorphic encryption. Doing so would give the first direct construction of fully homomorphic encryption without lattice-based assumptions. However, this task appears challenging.

One natural approach, following Gentry's technique [Gen09], would be to instantiate our somewhat homomorphic encryption scheme with a LHE scheme whose decryption circuit falls in the function class $\mathcal{F}_\tau$ — that is, we need LHE whose decryption can be written as a multivariate polynomial of total degree $O(\log \lambda / \log \log \lambda)$ with $\mathrm{poly}(\lambda)$ monomials. If such a LHE scheme were to exist, a tempting direction would be to "bootstrap" our SHE schemes in almost the standard way, using a ladder of encryptions of sparse-LPN secret keys under the LHE scheme and encryptions

of the LHE secret keys under the sparse-LPN scheme. Then, bootstrapping our schemes could take the following form: (a) evaluating a polynomial $f \in \mathcal{F}_\tau$ on our sparse-LPN ciphertexts, (b) performing ciphertext compaction, (c) homomorphically decrypting the resulting LHE ciphertext under our SHE scheme, using a sparse-LPN encryption of its secret key, and (d) repeating, perhaps with extra machinery to handle sparse-LPN error growth. The security of such a transformation would be implied by the hardness of sparse LPN and the semantic security of the LHE scheme.

However, this approach runs into trouble: first, we do not know of such a linearly homomorphic scheme. A symmetric encryption scheme of the type described above — i.e., with a decryption algorithm that can be written as a multivariate polynomial with total degree $O(\log \lambda / \log \log \lambda)$ with $\text{poly}(\lambda)$ monomials — exists assuming the pseudorandomness of Goldreich's function [Gol00] (or, more generally, the existence of local pseudorandom generators), but these schemes are not linearly homomorphic. One may be able to rule out the existence of linearly homomorphic encryption schemes with such a decryption procedure, which we leave as an open problem.

Second, even a bounded-depth version of the bootstrapping procedure described above would essentially give us a gadget that generates many sparse-LPN samples from few of them, without increasing the sparsity by much. This would render such a scheme vulnerable to known attacks on sparse LPN when the number of released samples is large (e.g., super-polynomial). The take-away is that our scheme does not appear "bootstrappable" in this natural way. While this rules out one approach to drawing more homomorphism from our schemes, a compelling open question remains whether other approaches exist.

Acknowledgements. We thank Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin for conversations about constant-sparse LPN. Conversations with Yuval always have unreasonably high information content: in this case, we thank him for discussions about private information retrieval and for insightful suggestions on how to present our results.

This work was funded in part by gifts from Apple, Capital One, Facebook, Google, Mozilla, NASDAQ, and MIT's FinTech@CSAIL Initiative, along with support from NSF under Award CNS-2054869. Alexandra Henzinger was supported by the National Science Foundation Graduate Research Fellowship under Grant No. 2141064. Vinod Vaikuntanathan was supported by NSF CNS-2154149 and a Simons Investigator Award.

References

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *STOC*, 2010. 3, 4, 7, 11, 16

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *S&P*, 2018. 3

[ADI+17] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO*, 2017. 7, 16, 21

[AIK06a] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC0. *SIAM Journal on Computing*, 36(4), 2006. 4

[AIK06b] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in NC0. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, 2006. 3, 4, 7, 16

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, 2015. 13

[AK19] Benny Applebaum and Eliran Kachlon. Sampling graphs without forbidden subgraphs and unbalanced expanders with negligible error. In *FOCS*, 2019. 21

[AL16] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In *STOC*, 2016. 4

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *FOCS*, 2003. 3, 4, 7, 11, 16

[AOW15] Sarah R Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *FOCS*, 2015. 4

[AS16] Prabhanjan Ananth and Amit Sahai. Functional encryption for turing machines. In *TCC*, 2016. 16

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, 2014. 3, 12

[BBD+20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In *TCC*, 2020. 11, 15, 23

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In *CRYPTO*, 2022. 11, 15, 23, 56

[BBTV25] Kiril Bangachev, Guy Bresler, Stefan Tiegel, and Vinod Vaikuntanathan. Near-optimal time-sparsity trade-offs for solving noisy linear equations. In *STOC*, 2025. 16

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *ACM SIGSAC*, 2018. 16

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC*, 2019. 12

[Ben87] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. Yale University, 1987. 12

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, 2001. 12

[BFKL93] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO*, 1993. 4, 15

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In *PKC*, 2013. 3, 12

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO*, 2016. 11, 13

[BGI+18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In *ITCS*, 2018. 13

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC*, 2005. 3, 12

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3), 2014. 3, 6, 12

[BHHO08] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *CRYPTO*, 2008. 17

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT*, 2019. 13

[BL11] Andrej Bogdanov and Chin Ho Lee. Homomorphic encryption from codes. *arXiv preprint arXiv:1111.4301*, 2011. 13

[BLLN13] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *IMACC*, 2013. 3, 12

[BLVW19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In *EUROCRYPT*, 2019. 11, 16

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *International Algorithmic Number Theory Symposium*, 1998. 5, 56

[BPTG14] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. *Cryptology ePrint Archive*, 2014. 3

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, 2012. 3, 12

[Bra13] Zvika Brakerski. When homomorphism becomes a liability. In *TCC*, 2013. 13

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, 2011. 3, 12

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, 2014. 3, 6, 12, 19, 20, 41

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, 2015. 13

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *ASIACRYPT*, 2016. 3, 12

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1), 2020. 6, 10, 12

[CGHK22] Henry Corrigan-Gibbs, Alexandra Henzinger, and Dmitry Kogan. Single-server private information retrieval with sublinear amortized time. In *EUROCRYPT*, 2022. 3

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *FOCS*, 1995. 3

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *ASIACRYPT*, 2017. 3, 12

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *CCS*, 2017. 3

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *TCC*, 2015. 3, 13, 16

[CM01] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC 0. In *MFCS*, 2001. 4

[dCHI+22] Leo de Castro, Carmit Hazay, Yuval Ishai, Vinod Vaikuntanathan, and Muthu Venkitasubramaniam. Asymptotically quasi-optimal cryptography. In *CRYPTO*, 2022. 6, 11, 23

[DGBL+17] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3), 2017. 3

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO*, 2019. 3, 11, 13, 15, 23, 56

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. 22(6), 1976. 5

[DIJL23] Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In *CRYPTO*, 2023. 3, 4, 5, 7, 11, 13, 16, 17, 20, 21, 34, 35, 46

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC*, 2001. 5, 12, 15, 18, 22

[DJ24] Quang Dao and Aayush Jain. Lossy cryptography from code-based assumptions. In *CRYPTO*, 2024. 11, 16

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *EUROCRYPT*, 2015. 3, 12

[DMQN12] Nico Döttling, Jörn Müller-Quade, and Anderson CA Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In *ASIACRYPT*, 2012. 16

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, 2012. 3

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory*, 31(4), 1985. 5, 12, 15, 18, 23

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *STOC*, 2002. 4

[FGJI17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from Paillier encryption. In *ProvSec*, 2017. 13

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009. 3, 6, 12, 25

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *TCC*, 2019. 12

[GHPS13] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P Smart. Field switching in BGV-style homomorphic encryption. *Journal of Computer Security*, 21(5), 2013. 3, 12

[GHS12] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, 2012. 3, 12

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, 2010. 12

[GKL93] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM Journal on Computing*, 22(6), 1993. 4

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 1984. 3, 5, 12, 20

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *IACR Cryp- tol. ePrint Arch*, 2000. 4, 26

[Gol01] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001. 14

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013. 3, 5, 7, 8, 12, 13, 36

[Hal17] Shai Halevi. Homomorphic encryption. In *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. 2017. 14

[HHCP19] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Logistic regression on homomorphic encrypted data at scale. In *AAAI*, volume 33, 2019. 3

[HLC+22] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. *Neurips*, 35, 2022. 3

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. NTRU: A ring-based public key cryptosystem. In *International algorithmic number theory symposium*. Springer, 1998. 3

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC*, 2008. 3, 4, 7, 16

[IKR23] Yuval Ishai, Eyal Kushnir, and Ron D Rothblum. Combinatorially homomorphic encryption. In *TCC*, 2023. 3

[IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *TCC*, 2007. 3, 13

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021. 3, 11, 13, 16

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over Fp, DLIN, and PRGs in NC 0. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2022. 3, 11, 13, 16

[JLS24] Aayush Jain, Huijia Lin, and Sagnik Saha. A systematic study of sparse LWE. In *CRYPTO*, 2024. 16

[JVC18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security*, 2018. 3

[KMOW17] Pravesh K Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In *STOC*, 2017. 4

[KMP14] Eike Kiltz, Daniel Masny, and Krzysztof Pietrzak. Simple chosen-ciphertext security from low-noise LPN. In *PKC*, 2014. 16

[KNY17] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. *Journal of Cryptology*, 30(2), 2017. 16

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, 1997. 3

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, 2012. 3, 12

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In *STOC*, 2023. 3

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, 2010. 3, 12

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *PKC*. 2016. 13

[LT22] Zeyu Liu and Eran Tromer. Oblivious message retrieval. In *CRYPTO*, 2022. 3

[MBGH11] Carlos Aguilar Melchor, Slim Bettaieb, Philippe Gaborit, and Javier Herranz. Improving additive and multiplicative homomorphic encryption schemes based on worst-case hardness assumptions. *Cryptology ePrint Archive*, 2011. 12

[MGH10] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d-operand multiplications. In *CRYPTO*, 2010. 3, 12

[Mic19] Daniele Micciancio. Fully homomorphic encryption from the ground up. `https://cseweb.ucsd.edu/~daniele/papers/FHEeurocrypt19-slides.pdf`, 2019. 7, 13

[MW22] Samir Jordan Menon and David J. Wu. Spiral: Fast, high-rate single-server PIR via FHE composition. In *S&P*, 2022. 3, 10

[OSI05] Rafail Ostrovsky and William E Skeith III. Private searching on streaming data. In *CRYPTO*, 2005. 3

[OSY21] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of Paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT*, 2021. 13

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999. 5, 12, 15, 18

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science*, 2012. 22

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, 2008. 6, 11, 22

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, 2008. 22

[Rab79] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. 1979. 12

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11), 1978. 3, 12

[RCK+21] Brandon Reagen, Woo-Seok Choi, Yeongil Ko, Vincent T Lee, Hsien-Hsin S Lee, Gu-Yeon Wei, and David Brooks. Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In *HPCA*, 2021. 3

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 2009. 3, 7

[RS21] Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In *CRYPTO*, 2021. 13

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. In *TCC*, 2024. 3, 4, 11, 13, 16

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, 2014. 16

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, 2010. 3, 12

[WH12] David Wu and Jacob Haven. Using homomorphic encryption for large scale statistical analysis. *FHE-SI-Report, Univ. Stanford, Tech. Rep. TR-dwu4*, 2012. 3

[WW24] Brent Waters and David J Wu. Adaptively-sound succinct arguments for NP from indistinguishability obfuscation. *Cryptology ePrint Archive*, 2024. 16

[YZ16] Yu Yu and Jiang Zhang. Cryptography with auxiliary input and trapdoor from constant-noise LPN. In *CRYPTO*, 2016. 11, 16

Supplementary Material

A Additional Material on Sparse LPN

We give a proof of Lemma 2.4, which was first proved by Dao, Ishai, Jain, and Lin [DIJL23]. We repeat the proof here since our KDM distribution is slightly (but not meaningfully) different. (In particular, we have the set $\mathsf{Diag}(\mathcal{S}_{k,n+1,n,q})$ correspond to matrices in $\mathbb{F}_q^{(n+1)\times n}$ that have k non-zero entries in every row, rather than having k non-zero entries in every row but the last one, which has $(k+1)/2$ non-zero entries.)

Proof. Let $\delta' = \delta - 1/\log n$, which gives that $n^{\delta'} = n^{\delta}/2$. Pick any odd $k \geq 3$, and let $k' = (k+1)/2 \in \mathbb{Z}$. Take any $m = \mathrm{poly}(\lambda)$ and consider any m messages $\mu_1, \ldots, \mu_m \in \mathbb{F}_q$. Let $m' = m \cdot (n+1) \cdot 2n^3$. Assume for the sake of contradiction that, for these messages, the lemma statement is false — that is, there exists a polynomial-time algorithm $\mathcal{A}$ that distinguishes between the two distributions in Lemma 2.4. We use algorithm $\mathcal{A}$ to break the (n, q, δ', k')-sparse LPN assumption.

To prove the reduction, we build an algorithm $\mathcal{B}$ that is given as input $(\mathbf{A}, \mathbf{b})$, where $\mathbf{A} \in \mathbb{F}_q^{m' \times n}$ and $\mathbf{b} \in \mathbb{F}_q^{m'}$. The algorithm $\mathcal{B}$ proceeds as follows:

1. For $i \in [m]$, for $j \in [n+1]$:
   - Let $\ell = i \cdot (n+1) + j \in \mathbb{Z}$.
   - If $j \leq n$,
     - Find a pair of distinct indices $k_1, k_2 \in [2n^3]$ such that, if $\mathbf{v}_1 \in \mathbb{F}_q^n$ denotes the $\left(\ell \cdot 2n^3 + k_1\right)^{\text{th}}$ row of $\mathbf{A}$ and $\mathbf{v}_2 \in \mathbb{F}_q^n$ denotes the $(\ell \cdot 2n^3 + k_2)^{\text{th}}$ row of $\mathbf{A}$, then the only entry in which the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are *both* non-zero is the jth one. If no such indices exist, fail.
     - Let $e_1, e_2 \in \mathbb{F}_q$ be the entries in the jth position in vectors $\mathbf{v}_1, \mathbf{v}_2$.
     - Sample $r \xleftarrow{R} \mathbb{F}_q$ such that $r \neq 0$. Then, sample random, non-zero $c_1, c_2 \in \mathbb{F}_q$ such that $c_1 \cdot e_1 + c_2 \cdot e_2 = r + \mu_i$. (This is possible because $\mathbb{F}_q$ is a finite field, and $q \geq 3$.)
     - Define the vector $\bar{\mathbf{a}}_{i,j} := c_1 \cdot \mathbf{v}_1 + c_2 \cdot \mathbf{v}_2 - \mu_i \cdot \mathbf{e}_j$ (where $\mathbf{e}_j$ denotes the jth unit vector in $\mathbb{F}_q^n$, which is "0" everywhere except for "1" at the jth entry).
     - Let $u_1, u_2 \in \mathbb{F}_q$ denote the $(\ell \cdot 2n^3 + k_1)^{\text{th}}$ and $(\ell \cdot 2n^3 + k_2)^{\text{th}}$ entries of vector $\mathbf{b}$ respectively. Define the scalar $\bar{b}_{i,j} := c_1 \cdot u_1 + c_2 \cdot u_2 \in \mathbb{F}_q$.
   - If $j = n+1$,
     - Sample $x \xleftarrow{R} [n]$. Find a pair of distinct indices $k_1, k_2 \in [2n^3]$ such that, if $\mathbf{v}_1 \in \mathbb{F}_q^n$ denotes the $(\ell \cdot 2n^3 + k_1)^{\text{th}}$ row of $\mathbf{A}$ and $\mathbf{v}_2 \in \mathbb{F}_q^n$ denotes the $(\ell \cdot 2n^3 + k_2)^{\text{th}}$ row of $\mathbf{A}$, then the only entry in which the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are *both* non-zero is the xth one. If no such indices exist, fail.
     - Let $e_1, e_2 \in \mathbb{F}_q$ be the entries in the xth position in vectors $\mathbf{v}_1, \mathbf{v}_2$.
     - Sample $r \xleftarrow{R} \mathbb{F}_q$ such that $r \neq 0$. Then, sample random, non-zero $c_1, c_2 \in \mathbb{F}_q$ such that $c_1 \cdot e_1 + c_2 \cdot e_2 = r$. (This is possible because $\mathbb{F}_q$ is a finite field, and $q \geq 3$.)
     - Then, define the vector $\bar{\mathbf{a}}_{i,j} := c_1 \cdot \mathbf{v}_1 + c_2 \cdot \mathbf{v}_2$.

– Also, let $u_1, u_2 \in \mathbb{F}_q$ denote the $(\ell \cdot 2n^3 + k_1)^{\text{th}}$ and $(\ell \cdot 2n^3 + k_2)^{\text{th}}$ entries of vector $\mathbf{b}$ respectively. Define the scalar $\bar{b}_{i,j} := c_1 \cdot u_1 + c_2 \cdot u_2 + \mu_i \in \mathbb{F}_q$.

2. Output the result from calling algorithm $\mathcal{A}$ on the list of matrices $\left(\bar{\mathbf{A}}_1, \bar{\mathbf{b}}_1, \ldots, \bar{\mathbf{A}}_m, \bar{\mathbf{b}}_m\right)$, where for $i \in [m]$ we define:

$$\bar{\mathbf{A}}_i := \begin{bmatrix} \bar{\mathbf{a}}_{i,1} \\ \vdots \\ \bar{\mathbf{a}}_{i,n+1} \end{bmatrix} \in \mathbb{F}_q^{(n+1)\times n} \qquad \text{and} \qquad \bar{\mathbf{b}}_i := \begin{bmatrix} \bar{b}_{i,1} \\ \vdots \\ \bar{b}_{i,n+1} \end{bmatrix} \in \mathbb{F}_q^{n+1}$$

By construction, we see that the algorithm $\mathcal{B}$ runs in polynomial time. Next, we analyze the probability that the algorithm $\mathcal{B}$ fails, when the input matrix $\mathbf{A} \in \mathbb{F}_q^{m' \times n}$ is a random k'-sparse matrix. By [DIJL23, Claim 4.1], the probability that any one iteration of the loop fails is $\mathrm{negl}(n)$. So, by a union bound, the probability that algorithm $\mathcal{B}$ fails, when the input matrix $\mathbf{A} \in \mathbb{F}_q^{m' \times n}$ is a random k'-sparse matrix, is also $\mathrm{negl}(n)$.

Finally, conditioned on the event that $\mathcal{B}$ does not fail, we see that for each $i \in [m]$:

- if $(\mathbf{A}, \mathbf{b})$ is a sparse-LPN sample with dimension n, modulus q, sparsity k', and error parameter δ', then $(\bar{\mathbf{A}}_i, \bar{\mathbf{b}}_i)$ is distributed exactly like a sample in our KDM distribution with sparsity k and noise parameter $\leq \delta$ (i.e., the left-hand side in the equation of Lemma 2.4).
- if $(\mathbf{A}, \mathbf{b})$ is distributed such that (1) $\mathbf{A}$ is a random k'-sparse matrix in $\mathcal{S}_{k',m',n,q}$ and (2) $\mathbf{b}$ is a uniformly random vector in $\mathbb{F}_q^{m'}$, then $(\bar{\mathbf{A}}_i, \bar{\mathbf{b}}_i)$ is distributed such that (1) $\bar{\mathbf{A}}_i$ is a random matrix in $\mathsf{Diag}(\mathcal{S}_{k,n+1,n,q})$ and (2) $\bar{\mathbf{b}}_i$ is a uniformly random vector in $\mathbb{F}_q^{n+1}$ (i.e., like the right-hand side in the equation of Lemma 2.4).

So, by our assumption that algorithm $\mathcal{A}$ can break KDM security with non-negligible probability, it must be that $\mathcal{B}$ can distinguish sparse-LPN samples from random ones with non-negligible probability. This contradicts the (n, q, δ', k')-sparse LPN distribution. □

B Additional Material on Somewhat Homomorphic Encryption

B.1 Proof of Theorem 3.1

On security parameter $\lambda \in \mathbb{N}$, functionality parameter $\tau = \mathrm{poly}(\lambda) \in \mathbb{N}$, and constant correctness parameter $c \in \mathbb{N}$, let (n, q, δ, k, Ψ) be the parameters of Construction 3.2 defined in the theorem statement. In this section, we prove that Construction 3.2 is a somewhat homomorphic encryption scheme for the function class $\mathcal{F}_\tau$ that satisfies correctness with probability $1 - \lambda^{-c} - \epsilon_{\mathsf{LHE}}(\lambda)$, semantic security, and compactness.

Proof. We prove correctness, security, and compactness separately.

Correctness. Let $(\mathsf{sk}, \mathsf{ek})$ be a key pair output by running $\mathsf{Gen}(1^\lambda, 1^\tau)$. We parse secret key sk as $(\mathbf{s} \in \mathbb{F}_q^n, \mathbf{t} \in \mathbb{F}_q^n, \mathsf{sk}_{\mathsf{LHE}})$ and evaluation key ek as $(\mathsf{ek}_{\mathsf{LHE}}, \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1})$. For the remainder of this argument, we will reason about Construction 3.2 using this key pair.

To analyze the scheme, we examine the distribution of ciphertexts output by Expand, Add, and Mul, taken over the encryption algorithm's randomness. We say that a distribution $\mathcal{D}$ over ciphertexts is (t, ϵ)-*good* with respect to a message $\mu \in \mathbb{F}_q$ and the fixed key $\begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix}^\intercal$, if it meets the following properties:

1. *Sparsity* t: every ciphertext in the support of $\mathcal{D}$ is a matrix in $\mathbb{F}_q^{(n+1)\times(n+1)}$ that has at most t non-zero entries in each of its rows.

2. *Error-rate* ϵ: the following equation holds:

$$\forall i \in [n+1], \quad \Pr_{\mathbf{C}\leftarrow\mathcal{D}}\left[e_i \neq 0 \;\middle|\; \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n+1} \end{bmatrix} := \mathbf{C} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} - \mu \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \right] \leq \epsilon. \qquad (*)$$

The $(*)$-equation here implies that a ciphertext matrix $\mathbf{C}$ sampled from the distribution $\mathcal{D}$ will have the "extended" secret-key vector $\begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix}^\intercal$ as an approximate eigenvector, with approximate eigenvalue μ — exactly as with GSW encryption [GSW13]. However, unlike with GSW, the eigenvector here is corrupted with sparse (rather than low-norm) noise.

We will show that, given any message $\mu \in \mathbb{F}_q$, the composition of algorithms $\mathsf{Expand}(\mathsf{ek}, \mathsf{Enc}(\mathsf{sk}, \cdot))$ outputs a matrix sampled from a distribution that is $((k+1)^2, (k+2)\cdot n^{-\delta})$-good with respect to message μ and secret-key $\begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix}^\intercal$. Moreover, for any two distributions $\mathcal{D}_{\mathsf{left}}$ and $\mathcal{D}_{\mathsf{right}}$ that are each (t, ϵ)-good with respect to messages μ_{left} and μ_{right}, the distribution

$$\mathcal{D}_{\mathsf{Add}} = \{\mathsf{Add}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}}) : \mathbf{C}_{\mathsf{left}} \leftarrow \mathcal{D}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}} \leftarrow \mathcal{D}_{\mathsf{right}}\}$$

is $(t_{\mathsf{Add}}, \epsilon_{\mathsf{Add}})$-good with respect to $\mu_{\mathsf{left}} + \mu_{\mathsf{right}}$. Similarly, the distribution

$$\mathcal{D}_{\mathsf{Mul}} = \{\mathsf{Mul}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}}) : \mathbf{C}_{\mathsf{left}} \leftarrow \mathcal{D}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}} \leftarrow \mathcal{D}_{\mathsf{right}}\}$$

is $(t_{\mathsf{Mul}}, \epsilon_{\mathsf{Mul}})$-good with respect to $\mu_{\mathsf{left}} \cdot \mu_{\mathsf{right}}$. Here, t_{Add} and t_{Mul} are somewhat bigger than t, and ϵ_{Add} and ϵ_{Mul} are somewhat bigger than ϵ, but they all remain bounded. In other words, the $(*)$-invariant holds after expanding ciphertexts output by the encryption algorithm, after homomorphic additions, and after homomorphic multiplications.

We perform this analysis in a sequence of claims:

Claim B.1 (GSW Encryption)**.** For any $\mu \in \mathbb{F}_q$, the algorithm $\mathsf{GSWEnc}(\mathbf{s}, \mu)$ produces a ciphertext sampled from a $(k+1, n^{-\delta})$-good distribution with respect to μ.

Proof. Sparsity analysis. The GSWEnc algorithm samples a matrix $\mathbf{A}$ from the set $\mathsf{Diag}(\mathcal{S}_{k,n+1,n,q})$. By definition, the $\mathbf{A}$-matrix here contains exactly k non-zero entries in each of its rows. Then, GSWEnc outputs a ciphertext matrix $\mathbf{C}$ that is the concatenation of $\mathbf{A}$ with a column vector. So, the sparsity of $\mathbf{C}$ is at most $k+1$.

Error-rate analysis. To show that the $(*)$-invariant holds after GSW encryption, we observe that the ciphertext matrix $\mathbf{C}$ output by $\mathsf{GSWEnc}(\mathbf{s}, \mu)$ is constructed as

$$\mathbf{C} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_1{}^\intercal\mathbf{s} + e_1 - s_1 \cdot \mu \\ \mathbf{a}_2 & \mathbf{a}_2{}^\intercal\mathbf{s} + e_2 - s_2 \cdot \mu \\ \vdots & \vdots \\ \mathbf{a}_n & \mathbf{a}_n{}^\intercal\mathbf{s} + e_n - s_n \cdot \mu \\ \mathbf{a}_{n+1} & \mathbf{a}_{n+1}{}^\intercal\mathbf{s} + e_{n+1} + \mu \end{bmatrix},$$

where vectors $\mathbf{a}_1, \ldots, \mathbf{a}_{n+1}$ are the rows of the $\mathbf{A}$-matrix, scalars $s_1, \ldots, s_n$ are the entries of the secret vector $\mathbf{s}$, and each scalar $e_1, \ldots, e_{n+1}$ is sampled from the error distribution $\mathsf{RandBern}_{n^{-\delta},q}$. Here, we see that:

$$\mathbf{C} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_1{}^{\mathsf{T}}\mathbf{s} + e_1 - s_1 \cdot \mu \\ \mathbf{a}_2 & \mathbf{a}_2{}^{\mathsf{T}}\mathbf{s} + e_2 - s_2 \cdot \mu \\ \vdots & \vdots \\ \mathbf{a}_n & \mathbf{a}_n{}^{\mathsf{T}}\mathbf{s} + e_n - s_n \cdot \mu \\ \mathbf{a}_{n+1} & \mathbf{a}_{n+1}{}^{\mathsf{T}}\mathbf{s} + e_{n+1} + \mu \end{bmatrix} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = \mu \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \\ e_{n+1} \end{bmatrix}$$

This implies that, for all $i \in [n+1]$:

$$\Pr\left[\tilde{e}_i \neq 0 \;\middle|\; \begin{bmatrix} \tilde{e}_1 \\ \tilde{e}_2 \\ \vdots \\ \tilde{e}_{n+1} \end{bmatrix} := \mathbf{C} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} - \mu \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix}\right] = \Pr\left[e_i \neq 0\right] = n^{-\delta}.$$

That is, the algorithm $\mathsf{GSWEnc}(\mathbf{s}, \mu)$ outputs a ciphertext sampled from a distribution that is $(k+1, n^{-\delta})$-good with respect to μ. □

Claim B.2 (Regev Encryption and Ciphertext Expansion)**.** For any $\mu \in \mathbb{F}_q$, the composition of algorithms $\mathsf{Expand}(\mathsf{ek}, \mathsf{Enc}(\mathsf{sk}, \mu))$ produces a ciphertext sampled from a $((k+1)^2, (k+2) \cdot n^{-\delta})$-good distribution with respect to μ.

Proof. Sparsity analysis. Consider any ciphertext $\mathbf{C}$ output by the following process:

$$\begin{aligned} \mathbf{c} &\leftarrow \mathsf{Enc}(\mathsf{sk}, \mu), \\ \mathbf{C} &\leftarrow \mathsf{Expand}(\mathsf{ek}, \mathbf{c}). \end{aligned}$$

By construction, the vector $\mathbf{c}$ has at most $(k+1)$ non-zero entries. Per Claim B.1, each row of the matrices $\mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}$ given as part of the evaluation key ek contains at most $(k+1)$ non-zero entries. As a result, since the matrix $\mathbf{C}$ is constructed as $\sum_{i \in [n+1]} c_i \cdot \mathbf{C}_{\mathsf{ek},i}$, it must be that the matrix $\mathbf{C}$ has sparsity at most $(k+1)^2$.

Error-rate analysis. Let $\tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & || & 1 \end{bmatrix}^{\mathsf{T}}$. Per Claim B.1, for each $i \in [n+1]$, the ciphertext $\mathbf{C}_{\mathsf{ek},i}$ is sampled from a distribution that is $((k+1), n^{-\delta})$-good with respect to the message $\tilde{t}_i$ and the secret-key $\begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix}^{\mathsf{T}}$. Now, we observe that, by construction:

$$\begin{aligned} \mathbf{C} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} &= \left(\sum_{i \in [n+1]} c_i \cdot \mathbf{C}_{\mathsf{ek},i} \right) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \\ &= \sum_{i \in [n+1]} c_i \cdot \left(\mathbf{C}_{\mathsf{ek},i} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \right) \\ &= \sum_{i \in [n+1]} c_i \cdot \left(\tilde{t}_i \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}^{(i)} \right), \end{aligned}$$

where we know that each entry of vector $\mathbf{e}^{(i)}$ is non-zero with probability at most $n^{-\delta}$ (by Claim B.1). We also know that $\mathbf{c} = \begin{bmatrix}\mathbf{a} & || & b\end{bmatrix}$, where vector $\mathbf{a}$ is k-sparse, scalar b is equal to $\langle \mathbf{a}, \mathbf{t}\rangle + e + \mu$, and error e is sampled from $\mathsf{RandBern}_{n^{-\delta},q}$. Then, we can re-write the above equation:

$$\begin{aligned}
\mathbf{C} \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} &= \sum_{i\in[n+1]} c_i \cdot \left(\tilde{t}_i \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \mathbf{e}^{(i)}\right) \\
&= \left(\sum_{i\in[n]} a_i \cdot \left(-t_i \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \mathbf{e}^{(i)}\right)\right) + b \cdot \left(\begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \mathbf{e}^{(n+1)}\right) \\
&= \left(-\langle \mathbf{a}, \mathbf{t}\rangle \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \sum_{i\in[n]} a_i \cdot \mathbf{e}^{(i)}\right) + (\langle \mathbf{a}, \mathbf{t}\rangle + e + \mu) \cdot \left(\begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \mathbf{e}^{(n+1)}\right) \\
&= \sum_{i\in[n]} a_i \cdot \mathbf{e}^{(i)} + \mu \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + e \cdot \left(\begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \mathbf{e}^{(n+1)}\right) + (\langle \mathbf{a}, \mathbf{t}\rangle + \mu) \cdot \mathbf{e}^{(n+1)} \\
&= \mu \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \underbrace{\sum_{i\in[n]} a_i \cdot \mathbf{e}^{(i)} + e \cdot \left(\begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} + \mathbf{e}^{(n+1)}\right) + (\langle \mathbf{a}, \mathbf{t}\rangle + \mu) \cdot \mathbf{e}^{(n+1)}}_{\text{error } \mathbf{e}_{\mathsf{new}}}.
\end{aligned}$$

By a union bound, each entry of error vector $\mathbf{e}_{\mathsf{new}}$ is non-zero with probability at most $(k+2)\cdot n^{-\delta}$. Equivalently, we can write that, for all $i \in [n+1]$,

$$\Pr\left[(\mathbf{e}_{\mathsf{new}})_i \neq 0 \;\middle|\; \mathbf{e}_{\mathsf{new}} := \mathsf{Expand}(\mathsf{ek}, \mathsf{Enc}(\mathsf{sk}, \mu)) \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix} - \mu \cdot \begin{bmatrix}-\mathbf{s}\\ 1\end{bmatrix}\right] \leq (k+2) \cdot n^{-\delta}.$$

That is, the output distribution is $((k+1)^2, (k+2)\cdot n^{-\delta})$-good with respect to the message μ. □

Claim B.3 (Homomorphic addition)**.** Given any distribution $\mathcal{D}_{\mathsf{left}}$ that is $(k_{\mathsf{left}}, \epsilon_{\mathsf{left}})$-good with respect to message μ_{left} and any distribution $\mathcal{D}_{\mathsf{right}}$ that is $(k_{\mathsf{right}}, \epsilon_{\mathsf{right}})$-good with respect to message μ_{right}, the distribution

$$\mathcal{D}_{\mathsf{Add}} = \{\mathsf{Add}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}}) : \mathbf{C}_{\mathsf{left}} \leftarrow \mathcal{D}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}} \leftarrow \mathcal{D}_{\mathsf{right}}\}$$

is $(k_{\mathsf{new}}, \epsilon_{\mathsf{new}})$-good with respect to message $(\mu_{\mathsf{left}} + \mu_{\mathsf{right}}) \in \mathbb{F}_q$, with sparsity $k_{\mathsf{new}} \leq k_{\mathsf{left}} + k_{\mathsf{right}}$ and error-rate $\epsilon_{\mathsf{new}} \leq \epsilon_{\mathsf{left}} + \epsilon_{\mathsf{right}}$.

Proof. Sparsity analysis. Let $\mathbf{C}_{\mathsf{left}}$ be any matrix in the support of $\mathcal{D}_{\mathsf{left}}$, let $\mathbf{C}_{\mathsf{right}}$ be any matrix in the support of $\mathcal{D}_{\mathsf{right}}$, and let $\mathbf{C}_{\mathsf{new}} := \mathsf{Add}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}})$. Then, each row in $\mathbf{C}_{\mathsf{left}}$ contains at most k_{left} non-zero entries, and each row in $\mathbf{C}_{\mathsf{right}}$ contains at most k_{right} non-zero entries. So, each row of $\mathbf{C}_{\mathsf{new}}$ contains at most $k_{\mathsf{left}} + k_{\mathsf{right}}$ non-zero entries.

Error-rate analysis. Consider matrices $\mathbf{C}_{\mathsf{left}}$ and $\mathbf{C}_{\mathsf{right}}$ sampled from $\mathcal{D}_{\mathsf{left}}$ and $\mathcal{D}_{\mathsf{right}}$, and let $\mathbf{C}_{\mathsf{new}} :=$

$\mathsf{Add}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}})$. We observe that:

$$\mathbf{C}_{\mathsf{new}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} = (\mathbf{C}_{\mathsf{left}} + \mathbf{C}_{\mathsf{right}}) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \tag{3}$$

$$= \mathbf{C}_{\mathsf{left}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{C}_{\mathsf{right}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \tag{4}$$

$$= \mu_{\mathsf{left}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}_{\mathsf{left}} + \mu_{\mathsf{right}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \mathbf{e}_{\mathsf{right}} \tag{5}$$

$$= \underbrace{(\mu_{\mathsf{left}} + \mu_{\mathsf{right}})}_{\text{message } \mu_{\mathsf{new}}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \underbrace{(\mathbf{e}_{\mathsf{left}} + \mathbf{e}_{\mathsf{right}})}_{\text{invariant error } \mathbf{e}_{\mathsf{new}}} . \tag{6}$$

Here, Equation (5) follows by the $(*)$-invariant, which tells us that the probability that each entry in error vectors $\mathbf{e}_{\mathsf{left}}$ and $\mathbf{e}_{\mathsf{right}}$ is non-zero is at most ϵ_{left} and $\epsilon_{\mathsf{right}}$, respectively. Since $\mathbf{e}_{\mathsf{new}} = \mathbf{e}_{\mathsf{left}} + \mathbf{e}_{\mathsf{right}}$, by a union bound, the probability of any one entry in the new invariant error, $\mathbf{e}_{\mathsf{new}}$, being non-zero is at most $\epsilon_{\mathsf{left}} + \epsilon_{\mathsf{right}}$.

Equivalently, we can write that, for all $i \in [n+1]$,

$$\Pr_{\substack{\mathbf{C}_{\mathsf{left}} \leftarrow \mathcal{D}_{\mathsf{left}} \\ \mathbf{C}_{\mathsf{right}} \leftarrow \mathcal{D}_{\mathsf{right}}}} \left[\tilde{e}_i \neq 0 \;\middle|\; \begin{bmatrix} \tilde{e}_1 \\ \tilde{e}_2 \\ \vdots \\ \tilde{e}_{n+1} \end{bmatrix} := \mathsf{Add}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}}) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} - \mu_{\mathsf{new}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \right] \leq \epsilon_{\mathsf{left}} + \epsilon_{\mathsf{right}}.$$

That is, the distribution $\mathcal{D}_{\mathsf{Add}}$ is $(k_{\mathsf{new}}, \epsilon_{\mathsf{new}})$-good with respect to the new message $\mu_{\mathsf{new}} = \mu_{\mathsf{left}} + \mu_{\mathsf{right}} \in \mathbb{F}_q$, for $k_{\mathsf{new}} \leq k_{\mathsf{left}} + k_{\mathsf{right}}$ and $\epsilon_{\mathsf{new}} \leq \epsilon_{\mathsf{left}} + \epsilon_{\mathsf{right}}$. □

Claim B.4 (Homomorphic multiplication)**.** Given any distribution $\mathcal{D}_{\mathsf{left}}$ that is $(k_{\mathsf{left}}, \epsilon_{\mathsf{left}})$-good with respect to message μ_{left} and any distribution $\mathcal{D}_{\mathsf{right}}$ that is $(k_{\mathsf{right}}, \epsilon_{\mathsf{right}})$-good with respect to message μ_{right}, the distribution

$$\mathcal{D}_{\mathsf{Mul}} = \{\mathsf{Mul}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}}) : \mathbf{C}_{\mathsf{left}} \leftarrow \mathcal{D}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}} \leftarrow \mathcal{D}_{\mathsf{right}}\}$$

is $(k_{\mathsf{new}}, \epsilon_{\mathsf{new}})$-good with respect to message $(\mu_{\mathsf{left}} \cdot \mu_{\mathsf{right}}) \in \mathbb{F}_q$, with sparsity $k_{\mathsf{new}} \leq k_{\mathsf{left}} \cdot k_{\mathsf{right}}$ and error-rate $\epsilon_{\mathsf{new}} \leq k_{\mathsf{left}} \cdot \epsilon_{\mathsf{right}} + \epsilon_{\mathsf{left}}$.

Proof. Sparsity analysis. Let $\mathbf{C}_{\mathsf{left}}$ be any matrix in the support of $\mathcal{D}_{\mathsf{left}}$, let $\mathbf{C}_{\mathsf{right}}$ be any matrix in the support of $\mathcal{D}_{\mathsf{right}}$, and let $\mathbf{C}_{\mathsf{new}} := \mathsf{Mul}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}}) = \mathbf{C}_{\mathsf{left}} \cdot \mathbf{C}_{\mathsf{right}}$. For any $j \in [n+1]$, let vector $\mathbf{u} \in \mathbb{F}_q^{n+1}$ denote the jth row in matrix $\mathbf{C}_{\mathsf{left}}$. Additionally, for any $i \in [n+1]$, let vector $\mathbf{v}^{(i)} \in \mathbb{F}_q^{n+1}$ denote the ith row in matrix $\mathbf{C}_{\mathsf{right}}$. Finally, let vector $\mathbf{w} \in \mathbb{F}_q^{n+1}$ denote the jth row in matrix $\mathbf{C}_{\mathsf{new}}$. Then, by construction, the row vector $\mathbf{w}$ is computed as follows:

$$\mathbf{w} = \sum_{\ell=1}^{n+1} u_\ell \cdot \mathbf{v}^{(\ell)} \tag{7}$$

We know that the vector $\mathbf{u}$ contains at most k_{left} non-zero entries, and that each vector $\mathbf{v}^{(\ell)}$ for $\ell \in [n+1]$ contains at most k_{right} non-zero entries. So, by Equation (7), the row vector $\mathbf{w}$ can contain at most $k_{\mathsf{left}} \cdot k_{\mathsf{right}}$ non-zero entries. Thus, the resulting ciphertext $\mathbf{C}_{\mathsf{new}}$ must have sparsity at most $k_{\mathsf{new}} \leq k_{\mathsf{left}} \cdot k_{\mathsf{right}}$.

Error-rate analysis. Consider matrices $\mathbf{C}_{\mathsf{left}}$ and $\mathbf{C}_{\mathsf{right}}$ sampled from $\mathcal{D}_{\mathsf{left}}$ and $\mathcal{D}_{\mathsf{right}}$, and let $\mathbf{C}_{\mathsf{new}} := \mathsf{Mul}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}})$. We observe that:

$$\begin{aligned}
\mathbf{C}_{\mathsf{new}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} &= \mathbf{C}_{\mathsf{left}} \cdot \mathbf{C}_{\mathsf{right}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} && (8)\\
&= \mathbf{C}_{\mathsf{left}} \cdot \left(\begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \cdot \mu_{\mathsf{right}} + \mathbf{e}_{\mathsf{right}} \right) && (9)\\
&= \mathbf{C}_{\mathsf{left}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \cdot \mu_{\mathsf{right}} + \mathbf{C}_{\mathsf{left}} \cdot \mathbf{e}_{\mathsf{right}} && (10)\\
&= \left(\begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \cdot \mu_{\mathsf{left}} + \mathbf{e}_{\mathsf{left}} \right) \cdot \mu_{\mathsf{right}} + \mathbf{C}_{\mathsf{left}} \cdot \mathbf{e}_{\mathsf{right}} && (11)\\
&= \underbrace{\mu_{\mathsf{left}}\mu_{\mathsf{right}}}_{\text{message } \mu_{\mathsf{new}}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} + \underbrace{\left(\mathbf{e}_{\mathsf{left}} \cdot \mu_{\mathsf{right}} + \mathbf{C}_{\mathsf{left}} \cdot \mathbf{e}_{\mathsf{right}}\right)}_{\text{invariant error } \mathbf{e}_{\mathsf{new}}}. && (12)
\end{aligned}$$

Here, Equations (9) and (11) each follow by the $(*)$-invariant, which tells us that the probability of any one entry in vectors $\mathbf{e}_{\mathsf{left}}$ and $\mathbf{e}_{\mathsf{right}}$ being non-zero is at most ϵ_{left} and $\epsilon_{\mathsf{right}}$, respectively. Since $\mathbf{e}_{\mathsf{new}} = \mu_{\mathsf{right}} \cdot \mathbf{e}_{\mathsf{left}} + \mathbf{C}_{\mathsf{left}} \cdot \mathbf{e}_{\mathsf{right}}$, where each row of $\mathbf{C}_{\mathsf{left}}$ contains at most k_{left} non-zero entries, by a union bound, the probability of any one entry in $\mathbf{e}_{\mathsf{new}}$ being non-zero is at most $\epsilon_{\mathsf{left}} + k_{\mathsf{left}} \cdot \epsilon_{\mathsf{right}}$.

Equivalently, we can write that, for all $i \in [n+1]$,

$$\Pr_{\substack{\mathbf{C}_{\mathsf{left}} \leftarrow \mathcal{D}_{\mathsf{left}} \\ \mathbf{C}_{\mathsf{right}} \leftarrow \mathcal{D}_{\mathsf{right}}}} \left[\tilde{e}_i \neq 0 \;\middle|\; \begin{bmatrix} \tilde{e}_1 \\ \tilde{e}_2 \\ \vdots \\ \tilde{e}_{n+1} \end{bmatrix} := \mathsf{Mul}(\mathbf{C}_{\mathsf{left}}, \mathbf{C}_{\mathsf{right}}) \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} - \mu_{\mathsf{new}} \cdot \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \right] \leq \epsilon_{\mathsf{left}} + k_{\mathsf{left}} \cdot \epsilon_{\mathsf{right}}.$$

That is, the distribution $\mathcal{D}_{\mathsf{Mul}}$ is $(k_{\mathsf{new}}, \epsilon_{\mathsf{new}})$-good with respect to the message $\mu_{\mathsf{new}} = \mu_{\mathsf{left}} \cdot \mu_{\mathsf{right}} \in \mathbb{F}_q$, for $k_{\mathsf{new}} \leq k_{\mathsf{left}} \cdot k_{\mathsf{right}}$ and $\epsilon_{\mathsf{new}} \leq \epsilon_{\mathsf{left}} + k_{\mathsf{left}} \cdot \epsilon_{\mathsf{right}}$. □

Finally, we observe that the $(*)$-invariant is exactly used by the $\mathsf{Compact}$ and Dec algorithms to squash and decrypt ciphertexts. Here, we must now consider the randomness over both the Gen and Enc algorithms (that is, we no longer work with a fixed key pair $(\mathsf{sk}, \mathsf{ek})$).

Claim B.5 (Compaction and Decryption)**.** Sample a key pair $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda, 1^\tau)$. Given any distribution $\mathcal{D}$ that is (t, ϵ)-good with respect to a message $\mu \in \mathbb{F}_q$ and the secret key $\mathbf{s}$ (given in sk), it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{Compact}(\mathsf{ek}, \mathbf{C})) \neq \mu \mid \mathbf{C} \leftarrow \mathcal{D}\right] \leq \epsilon + \epsilon_{\mathsf{LHE}}(\lambda).$$

Proof. Consider a ciphertext $\mathbf{C} \leftarrow \mathcal{D}$, and let $\mathbf{c} \in \mathbb{F}_q^{n+1}$ denote the last row of ciphertext matrix $\mathbf{C}$. Looking at just the last row of $\mathbf{C}$, because $\mathcal{D}$ is (t, ϵ)-good, we know that $\Pr\left[\begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix}^\intercal \cdot \mathbf{c} \neq \mu\right] \leq \epsilon$.

Then, the algorithm $\mathsf{Compact}(\mathsf{ek}, \mathbf{C})$ outputs the result of the affine function

$$f_{\mathbf{c}}(x_1, \ldots, x_{n+1}) = \sum_{i \in [n+1]} c_i \cdot x_i = \mathbf{x}^\intercal \cdot \mathbf{c}$$

applied to the LHE-encryption of the secret-key vector $\tilde{\mathbf{s}} = \begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix}^\intercal$. Then, the $\mathsf{Dec}(\mathsf{sk}, \cdot)$ algorithm decrypts this LHE ciphertext. So, because our LHE scheme has $\epsilon_{\mathsf{LHE}}(\lambda)$ probability of correctness failures, taking the probability over the randomness of Gen and Enc, by a union bound we see that:

$$\begin{aligned} \Pr\left[\mathsf{Dec}\left(\mathsf{sk}, \mathsf{Compact}(\mathsf{ek}, \mathbf{C})\right) \neq \mu\right] &\leq \Pr\left[f_{\mathbf{c}}(\tilde{s}_1, \ldots, \tilde{s}_{n+1}) \neq \mu\right] + \epsilon_{\mathsf{LHE}}(\lambda) \\ &= \Pr\left[\begin{bmatrix} -\mathbf{s} & 1 \end{bmatrix}^\intercal \cdot \mathbf{c} \neq \mu\right] + \epsilon_{\mathsf{LHE}}(\lambda) \\ &\leq \epsilon + \epsilon_{\mathsf{LHE}}(\lambda). \end{aligned}$$

□

An inductive argument on Claims B.2 to B.5 shows that we can chain homomorphic operations computed with Add and Mul on ciphertexts encrypted with $\mathsf{Enc}(\mathsf{sk}, \cdot)$ and expanded with $\mathsf{Expand}(\mathsf{ek}, \cdot)$. As long as the error growth is not too large, the algorithms $\mathsf{Compact}(\mathsf{ek}, \cdot)$ and $\mathsf{Dec}(\mathsf{sk}, \cdot)$ will recover the correct output of the computation. For every Add, the sparsity and the error-rate of the resulting ciphertexts grow *additively* (Claim B.3). For every Mul, the sparsity grows *multiplicatively* and — exactly as with GSW encryption [BV14] — the error-rate grows *asymmetrically*: it scales linearly with the sparsity of the left operand (Claim B.4).

To evaluate degree-d products, we perform multiplications in a "straight line," such that each call to Mul takes as input a fresh ciphertext with low sparsity as its left operand. With this evaluation strategy, computing a degree-d product on "fresh" ciphertexts (with sparsity $(k+1)^2$ and error-rate $(k+2) \cdot n^{-\delta}$, per Claim B.2) produces an encryption of their product with sparsity $(k+1)^{2d}$ and error rate

$$(k+2) \cdot n^{-\delta} \cdot \left(\sum_{j=0}^{d-1} (k+1)^{2j} \right) = n^{-\delta} \cdot \frac{(k+1)^{2d} - 1}{k} \leq n^{-\delta} \cdot (k+1)^{2d}.$$

Then, performing M additions after this degree-d product gives an encryption of the result with sparsity $M \cdot (k+1)^{2d}$ and error rate at most $M \cdot (k+1)^{2d} \cdot n^{-\delta}$.

To homomorphically evaluate polynomials in $\mathcal{F}_\tau$ with correctness error $(\lambda^{-c} + \epsilon_{\mathsf{LHE}}(\lambda))$, it then suffices to set the secret dimension n to be sufficiently large. To achieve this, when computing products of total degree d and then adding up M such terms, we set the parameters so that

$$M \cdot (k+1)^{2d} \cdot n^{-\delta} \leq \lambda^{-c}. \qquad (**)$$

Taking $M = \tau$ and $d = \log \tau / \log \log \tau$, we get that $\tau \cdot (k+1)^{2\frac{\log \tau}{\log \log \tau}} \leq n^\delta \cdot \lambda^{-c}$. When $k \leq \sqrt{\log \tau} - 1$, this equation is satisfied whenever $n \geq \tau^{2/\delta} \cdot \lambda^{c/\delta}$, as required by the theorem statement. Applying Claim B.5 proves the final result.

Security. We now show that Construction 3.2 satisfies semantic security. Consider any number of messages $m = \mathrm{poly}(\lambda)$. Let $\delta' := \delta - 1/\log n$, as in the theorem statement. We need to show that, given the evaluation key ek, any m ciphertexts output by Enc look computationally indistinguishable, regardless of the underlying messages being encrypted. The evaluation key output by Gen consists of:

1. an evaluation key $\mathsf{ek}_{\mathsf{LHE}}$ for the underlying LHE scheme,

2. $(n+1)$ LHE encryptions of the entries of $\tilde{\mathbf{s}} = \begin{bmatrix}-\mathbf{s} & 1\end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1}$, and

3. $(n+1)$ GSW-style sparse-LPN encryptions of the entries of $\tilde{\mathbf{t}} = \begin{bmatrix}-\mathbf{t} & 1\end{bmatrix} \in \mathbb{F}_q^{n+1}$, under secret key $\mathbf{s}$.

To prove security, we will show that, assuming $(n, q, \delta', (k+1)/2)$-sparse LPN and under the LHE scheme's semantic security, the encryption of any m messages looks computationally indistinguishable from the encryption of m zeros.

We show this via a hybrid argument with five steps:

1. First, by the semantic security of the underlying LHE scheme, we can swap the LHE encryptions of the entries of secret vector $\tilde{\mathbf{s}}$ with LHE encryptions of the all-zeros vector.

2. Second, by the semantic security of ciphertexts produced by GSWEnc (which in turn follows from the KDM-security of sparse LPN), we can swap the sparse-LPN encryptions of the entries of $\tilde{\mathbf{t}}$, under secret key $\mathbf{s}$, with sparse-LPN encryptions of the all-zeros vector under $\mathbf{s}$.

3. Third, by the sparse-LPN assumption, we can swap the m ciphertexts output by Enc, encrypting $\mu_1, \ldots, \mu_m$ under secret-key $\mathbf{t}$, with m encryptions of zero under secret-key $\mathbf{t}$.

4. Fourth, by the semantic security of ciphertexts produced by GSWEnc (which follows from the KDM-security of sparse LPN), we can swap the sparse-LPN encryptions of the all-zeros vector under secret key $\mathbf{s}$ back to the sparse-LPN encryptions of the entries of $\tilde{\mathbf{t}}$, under $\mathbf{s}$.

5. Fifth, by the semantic security of the underlying LHE scheme, we can swap the LHE encryptions of the all-zeros vector back to LHE encryptions of the entries of secret key $\tilde{\mathbf{s}}$.

We formalize this argument with the following five claims.

Claim B.6. $\mathcal{D}_0 \overset{c}{\approx} \mathcal{D}_1$, where we define:

$$\mathcal{D}_0 = \left\{ \begin{array}{c|rl} \mathsf{ek}_{\mathsf{LHE}}, & \mathbf{s} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix}-\mathbf{s} & || & 1\end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, & \mathbf{t} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix}-\mathbf{t} & || & 1\end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, & \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathbf{c}_1, \ldots, \mathbf{c}_m & \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \tilde{s}_i), \forall\, i \in [n+1] \\ & \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i), \forall\, i \in [n+1] \\ & \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, \mu_j), \forall\, j \in [m] \end{array} \right\}$$

$$\mathcal{D}_1 = \left\{ \begin{array}{c|rl} \mathsf{ek}_{\mathsf{LHE}}, & \mathbf{s} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix}-\mathbf{s} & || & 1\end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, & \mathbf{t} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix}-\mathbf{t} & || & 1\end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, & \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathbf{c}_1, \ldots, \mathbf{c}_m & \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0), \forall\, i \in [n+1] \\ & \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i), \forall\, i \in [n+1] \\ & \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, \mu_j), \forall\, j \in [m] \end{array} \right\}.$$

Proof. For the sake of contradiction, let there be a polynomial-time algorithm $\mathcal{A}$ that distinguishes between the two distributions with advantage $\epsilon_{\mathsf{adv}} \geq 1/\operatorname{poly}(\lambda)$. That is, when given a sample from distribution $\mathcal{D}_0$, $\mathcal{A}$ outputs "1" with probability p_{left}; when given a sample from distribution $\mathcal{D}_1$, $\mathcal{A}$ outputs "1" with probability p_{right}; and it must be that $|p_{\text{left}} - p_{\text{right}}| \geq \epsilon_{\mathsf{adv}}$.

Consider the sequence of $n+2$ distributions $\mathcal{D}_{0,0}, \ldots, \mathcal{D}_{0,n+1}$ defined as, $\forall\, k \in \{0, 1, \ldots, n+1\}$:

$$\mathcal{D}_{0,k} = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0)\,, \forall\, i \in [k] \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \tilde{s}_i)\,, \forall\, i \in \{k+1, \ldots, n+1\} \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i)\,, \forall\, i \in [n+1] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, \mu_j)\,, \forall\, j \in [m] \end{array} \right\}.$$

Let p_k denote the probability of algorithm $\mathcal{A}$ outputting "1" given an input from distribution $\mathcal{D}_{0,k}$. Since distribution $\mathcal{D}_{0,0}$ is equal to $\mathcal{D}_0$, we have $p_{\text{left}} = p_0$. Similarly, since distribution $\mathcal{D}_{0,n+1}$ is equal to $\mathcal{D}_1$, we have $p_{\text{right}} = p_{n+1}$. As a result, $|p_0 - p_{n+1}| \geq \epsilon_{\mathsf{adv}}$, so there must exist some index $i \in [n+1]$ such that $|p_i - p_{i-1}| \geq \epsilon_{\mathsf{adv}}/(n+1)$. That is, the algorithm $\mathcal{A}$ can distinguish between distributions $\mathcal{D}_{0,i}$ and $\mathcal{D}_{0,i-1}$ with non-negligible advantage.

Since distributions $\mathcal{D}_{0,i}$ and $\mathcal{D}_{0,i-1}$ differ only in the ith LHE encryption in the evaluation key being an encryption of either 0 or $\tilde{s}_i$, this means that algorithm $\mathcal{A}$ can break the semantic security of the LHE scheme. This is a contradiction. □

Claim B.7. $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$, where we define:

$$\mathcal{D}_2 = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0)\,, \forall\, i \in [n+1] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, 0)\,, \forall\, i \in [n+1] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, \mu_j)\,, \forall\, j \in [m] \end{array} \right\}.$$

Proof. For the sake of contradiction, let there be a polynomial-time algorithm $\mathcal{A}$ that distinguishes between the two distributions with advantage $\epsilon_{\mathsf{adv}} \geq 1/\operatorname{poly}(\lambda)$. That is, when given a sample from distribution $\mathcal{D}_1$, $\mathcal{A}$ outputs "1" with probability p_{left}; when given a sample from distribution $\mathcal{D}_2$, $\mathcal{A}$ outputs "1" with probability p_{right}; and it must be that $|p_{\text{left}} - p_{\text{right}}| \geq \epsilon_{\mathsf{adv}}$.

Now, consider the $n+2$ distributions $\mathcal{D}_{1,0}, \ldots, \mathcal{D}_{1,n+1}$ defined as, $\forall\, k \in \{0, 1, \ldots, n+1\}$:

$$\mathcal{D}_{1,k} = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0)\,, \forall\, i \in [n+1] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, 0)\,, \forall\, i \in [k] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i)\,, \forall\, i \in \{k+1, \ldots, n+1\} \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, \mu_j)\,, \forall\, j \in [m] \end{array} \right\}.$$

Let p_k denote the probability of algorithm $\mathcal{A}$ outputting "1" given an input from distribution $\mathcal{D}_{1,k}$. Since distribution $\mathcal{D}_{1,0}$ is equal to $\mathcal{D}_1$, we have $p_{\text{left}} = p_0$. Similarly, since distribution $\mathcal{D}_{1,n+1}$ is equal to $\mathcal{D}_2$, we have $p_{\text{right}} = p_{n+1}$. As a result, $|p_0 - p_{n+1}| \geq \epsilon_{\mathsf{adv}}$, so there must exist some index $i \in [n+1]$ such that $|p_i - p_{i-1}| \geq \epsilon_{\mathsf{adv}}/(n+1)$. That is, the algorithm $\mathcal{A}$ can distinguish between distributions $\mathcal{D}_{1,i}$ and $\mathcal{D}_{1,i-1}$ with non-negligible advantage.

The distributions $\mathcal{D}_{1,i}$ and $\mathcal{D}_{1,i-1}$ differ only in the ith sparse-LPN encryption in the evaluation key being an encryption of either 0 or $\tilde{t}_i$. By Lemma 2.4, assuming $(n, q, \delta', (k+1)/2)$-sparse LPN, these two sparse-LPN encryptions (of either 0 or $\tilde{t}_i$) are both computationally indistinguishable from the distribution

$$\left\{(\mathbf{A}, \mathbf{u}) : \mathbf{A} \xleftarrow{R} \mathsf{Diag}(\mathcal{S}_{k,n+1,n,q}),\ \mathbf{u} \xleftarrow{R} \mathbb{F}_q^{n+1}\right\}.$$

This is a contradiction. □

Claim B.8. $\mathcal{D}_2 \overset{c}{\approx} \mathcal{D}_3$, where we define:

$$\mathcal{D}_3 = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0), \forall\, i \in [n+1] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, 0), \forall\, i \in [n+1] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, 0), \forall\, j \in [m] \end{array} \right\}.$$

Proof. For the sake of contradiction, let there be a polynomial-time algorithm $\mathcal{A}$ that distinguishes between the two distributions with advantage $\epsilon_{\mathsf{adv}} \geq 1/\operatorname{poly}(\lambda)$. That is, when given a sample from distribution $\mathcal{D}_2$, $\mathcal{A}$ outputs "1" with probability p_{left}; when given a sample from distribution $\mathcal{D}_3$, $\mathcal{A}$ outputs "1" with probability p_{right}; and it must be that $|p_{\text{left}} - p_{\text{right}}| \geq \epsilon_{\mathsf{adv}}$.

Now, consider the sequence of $m+1$ distributions $\mathcal{D}_{2,0}, \ldots, \mathcal{D}_{2,m}$ defined as, $\forall\, k \in \{0, 1, \ldots, m\}$:

$$\mathcal{D}_{2,k} = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{R} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0), \forall\, i \in [n+1] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, 0), \forall\, i \in [n+1] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, 0), \forall\, j \in [k] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, \mu_j), \forall\, j \in \{k+1, \ldots, m\} \end{array} \right\}.$$

Let p_k denote the probability of algorithm $\mathcal{A}$ outputting "1" given an input from distribution $\mathcal{D}_{2,k}$. Since distribution $\mathcal{D}_{2,0}$ is equal to $\mathcal{D}_2$, we have $p_{\text{left}} = p_0$. Similarly, since distribution $\mathcal{D}_{2,m}$ is equal to $\mathcal{D}_3$, we have $p_{\text{right}} = p_m$. As a result, $|p_0 - p_m| \geq \epsilon_{\mathsf{adv}}$, so there must exist some index $i \in [m]$ such that $|p_i - p_{i-1}| \geq \epsilon_{\mathsf{adv}}/m$. That is, the algorithm $\mathcal{A}$ can distinguish between distributions $\mathcal{D}_{2,i}$ and $\mathcal{D}_{2,i-1}$ with non-negligible advantage.

The distributions $\mathcal{D}_{2,i}$ and $\mathcal{D}_{2,i-1}$ differ only in the ith sparse-LPN ciphertext being an encryption of either 0 or μ_i. Assuming $(n, q, \delta', (k+1)/2)$-sparse LPN, these two sparse-LPN encryptions (of either 0 or μ_i) are both computationally indistinguishable from the distribution

$$\left\{(\mathbf{a}, u) : \mathbf{a} \xleftarrow{R} \mathcal{S}_{k,1,n,q},\ u \xleftarrow{R} \mathbb{F}_q\right\}.$$

(This is because $(n, q, \delta', (k+1)/2)$-sparse LPN directly implies (n, q, δ, k)-sparse LPN[3], for prime modulus $q \geq 3$.) This is a contradiction. □

[3]The case of $j = n+1$ in the reduction in Appendix A gives a proof of this statement.

Claim B.9. $\mathcal{D}_3 \overset{c}{\approx} \mathcal{D}_4$, where we define:

$$\mathcal{D}_4 = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0), \forall\, i \in [n+1] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i), \forall\, i \in [n+1] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, 0), \forall\, j \in [m] \end{array} \right\}.$$

Proof. For the sake of contradiction, let there be a polynomial-time algorithm $\mathcal{A}$ that distinguishes between the two distributions with advantage $\epsilon_{\mathsf{adv}} \geq 1/\operatorname{poly}(\lambda)$. That is, when given a sample from distribution $\mathcal{D}_3$, $\mathcal{A}$ outputs "1" with probability p_{left}; when given a sample from distribution $\mathcal{D}_4$, $\mathcal{A}$ outputs "1" with probability p_{right}; and it must be that $|p_{\text{left}} - p_{\text{right}}| \geq \epsilon_{\mathsf{adv}}$.

Now, consider the $n+2$ distributions $\mathcal{D}_{3,0}, \ldots, \mathcal{D}_{3,n+1}$ defined as, $\forall\, k \in \{0, 1, \ldots, n+1\}$:

$$\mathcal{D}_{3,k} = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0), \forall\, i \in [n+1] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i), \forall\, i \in [k] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, 0), \forall\, i \in \{k+1, \ldots, n+1\} \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, 0), \forall\, j \in [m] \end{array} \right\}.$$

Let p_k denote the probability of algorithm $\mathcal{A}$ outputting "1" given an input from distribution $\mathcal{D}_{3,k}$. Since distribution $\mathcal{D}_{3,0}$ is equal to $\mathcal{D}_3$, we have $p_{\text{left}} = p_0$. Similarly, since distribution $\mathcal{D}_{3,n+1}$ is equal to $\mathcal{D}_4$, we have $p_{\text{right}} = p_{n+1}$. As a result, $|p_0 - p_{n+1}| \geq \epsilon_{\mathsf{adv}}$, so there must exist some index $i \in [n+1]$ such that $|p_i - p_{i-1}| \geq \epsilon_{\mathsf{adv}}/(n+1)$. That is, the algorithm $\mathcal{A}$ can distinguish between distributions $\mathcal{D}_{3,i}$ and $\mathcal{D}_{3,i-1}$ with non-negligible advantage.

The distributions $\mathcal{D}_{3,i}$ and $\mathcal{D}_{3,i-1}$ differ only in the ith sparse-LPN encryption in the evaluation key being an encryption of either 0 or $\tilde{t}_i$. By Lemma 2.4, assuming $(n, q, \delta', (k+1)/2)$-sparse LPN, these two sparse-LPN encryptions (of either 0 or $\tilde{t}_i$) are both computationally indistinguishable from the distribution

$$\left\{ (\mathbf{A}, \mathbf{u}) : \mathbf{A} \xleftarrow{\mathrm{R}} \mathsf{Diag}(\mathcal{S}_{k,n+1,n,q}),\ \mathbf{u} \xleftarrow{\mathrm{R}} \mathbb{F}_q^{n+1} \right\}.$$

This is a contradiction. □

Claim B.10. $\mathcal{D}_4 \overset{c}{\approx} \mathcal{D}_5$, where we define:

$$\mathcal{D}_5 = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{\mathrm{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & \| & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \tilde{s}_i), \forall\, i \in [n+1] \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i), \forall\, i \in [n+1] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, 0), \forall\, j \in [m] \end{array} \right\}.$$

Proof. For the sake of contradiction, let there be a polynomial-time algorithm $\mathcal{A}$ that distinguishes between the two distributions with advantage $\epsilon_{\mathsf{adv}} \geq 1/\operatorname{poly}(\lambda)$. That is, when given a sample from

distribution $\mathcal{D}_4$, $\mathcal{A}$ outputs "1" with probability p_{left}; when given a sample from distribution $\mathcal{D}_5$, $\mathcal{A}$ outputs "1" with probability p_{right}; and it must be that $|p_{\text{left}} - p_{\text{right}}| \geq \epsilon_{\mathsf{adv}}$.

Now, consider the $n+2$ distributions $\mathcal{D}_{4,0}, \ldots, \mathcal{D}_{4,n+1}$ defined as, $\forall\, k \in \{0, 1, \ldots, n+1\}$:

$$\mathcal{D}_{4,k} = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}, \\ \mathbf{C}_{\mathsf{ek},1}, \ldots, \mathbf{C}_{\mathsf{ek},n+1}, \\ \mathbf{c}_1, \ldots, \mathbf{c}_m \end{array} \middle| \begin{array}{rl} \mathbf{s} & \xleftarrow{\text{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{s}} := \begin{bmatrix} -\mathbf{s} & || & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathbf{t} & \xleftarrow{\text{R}} \mathbb{F}_q^n \text{ and } \tilde{\mathbf{t}} := \begin{bmatrix} -\mathbf{t} & || & 1 \end{bmatrix}^\intercal \in \mathbb{F}_q^{n+1} \\ \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^{n+1}) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \tilde{s}_i)\,, \forall\, i \in [k] \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, 0)\,, \forall\, i \in \{k+1, \ldots, n+1\} \\ \mathbf{C}_{\mathsf{ek},i} & \leftarrow \mathsf{GSWEnc}(\mathbf{s}, \tilde{t}_i)\,, \forall\, i \in [n+1] \\ \mathbf{c}_j & \leftarrow \mathsf{Enc}(\mathbf{t}, 0)\,, \forall\, j \in [m] \end{array} \right\}.$$

Let p_k denote the probability of algorithm $\mathcal{A}$ outputting "1" given an input from distribution $\mathcal{D}_{4,k}$. Since distribution $\mathcal{D}_{4,0}$ is equal to $\mathcal{D}_4$, we have $p_{\text{left}} = p_0$. Similarly, since distribution $\mathcal{D}_{4,n+1}$ is equal to $\mathcal{D}_5$, we have $p_{\text{right}} = p_{n+1}$. As a result, $|p_0 - p_{n+1}| \geq \epsilon_{\mathsf{adv}}$, so there must exist some index $i \in [n+1]$ such that $|p_i - p_{i-1}| \geq \epsilon_{\mathsf{adv}}/(n+1)$. That is, the algorithm $\mathcal{A}$ can distinguish between distributions $\mathcal{D}_{4,i}$ and $\mathcal{D}_{4,i-1}$ with non-negligible advantage.

Since distributions $\mathcal{D}_{4,i}$ and $\mathcal{D}_{4,i-1}$ differ only in the ith LHE encryption in the evaluation key being an encryption of either 0 or $\tilde{s}_i$, this means that algorithm $\mathcal{A}$ can break the semantic security of the LHE scheme. This is a contradiction. □

Compactness. Finally, we examine the size of ciphertexts in Construction 3.2.

Size of fresh ciphertexts. Each ciphertext output by Enc is a $(k+1)$-sparse vector in $\mathbb{F}_q^{n+1}$. So, it can be represented in $\leq (k+1) \cdot \log n \cdot \log q = \text{poly}(\lambda)$ bits.

Size of evaluated ciphertexts. Each ciphertext output by $\mathsf{Compact}$ is a ciphertext of the underlying LHE scheme, produced by $\mathsf{LHE.Eval}$. Here, $\mathsf{LHE.Eval}$ is given:

- an evaluation key $\mathsf{ek}_{\mathsf{LHE}}$, output by $\mathsf{LHE.Gen}(1^\lambda, 1^{n+1})$,
- an affine function $f : \mathbb{F}_q^{n+1} \to \mathbb{F}_q$, and
- $n+1$ LHE ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_{n+1}$, encrypted by calling $\mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \cdot)$.

By definition, the $(n+1)$-variate affine function f is in the function class supported by the LHE scheme with the given parameters (see Section 2.2). Then, since the LHE scheme must satisfy compactness, evaluated ciphertexts in Construction 3.2 must also be compact. □

B.2 Proof of Remark 3.3

Remark 3.3 describes the efficiency of the SHE scheme in Construction 3.2.

Proof. In Construction 3.2, we can take advantage of the sparsity in our ciphertexts by pruning the computation required by each Add and Mul so as to perform only those operations that affect the final output of a computation. Pruning the computation in this way does not affect the scheme's correctness, error growth, or security, but improves the cost of operating on (and storing) intermediate ciphertexts. An analogous observation was previously applied in the context of homomorphic secret sharing by Dao et al. [DIJL23, Remark 5.4].

In more detail, each ciphertext before compaction is a matrix in $\mathbb{F}_q^{(n+1)\times(n+1)}$. However, the $\mathsf{Compact}$ algorithm only inspects the last row of the ciphertext matrix $\mathbf{C}$ it is given. So, we can materialize only those rows of the intermediate ciphertexts that affect the last row of matrix $\mathbf{C}$.

Consider a degree-d multiplication circuit computing

$$\mathbf{C}_{\mathsf{new}} \leftarrow \mathbf{C}_d \cdot \mathbf{C}_{d-1} \cdot \ldots \cdot \mathbf{C}_2 \cdot \mathbf{C}_1,$$

where each input ciphertext has sparsity $k_{\mathsf{init}} \in \mathbb{N}$. Per Appendix B.1, we perform this computation as $\mathbf{C}_{\mathsf{new}} \leftarrow \mathbf{C}_d \cdot (\mathbf{C}_{d-1} \cdot (\ldots \cdot (\mathbf{C}_2 \cdot \mathbf{C}_1)))$, in a straight line from right to left to minimize error growth.

In the last call to Mul, we can materialize only the *last row* of the output ciphertext $\mathbf{C}_{\mathsf{new}}$. This lets us compute over only the last row of ciphertext $\mathbf{C}_d$, given as the left operand to Mul. Since the ciphertext $(\mathbf{C}_{d-1} \cdot (\ldots \cdot (\mathbf{C}_2 \cdot \mathbf{C}_1)))$ has sparsity $k_{\mathsf{right}} = k_{\mathsf{init}}^{d-1}$, the cost of this multiplication of a row vector with a ciphertext matrix, exactly as in Equation (7), is $k_{\mathsf{init}} \cdot k_{\mathsf{right}} = k_{\mathsf{init}}^d$ operations in $\mathbb{F}_q$.

In the second-to-last call to Mul, operating on $\mathbf{C}_{d-1}$ and $\mathbf{C}_{d-2} \cdot (\ldots \cdot (\mathbf{C}_2 \cdot \mathbf{C}_1))$, we similarly compute only the k_{init} rows of the output that are needed by the final call to Mul—namely, those corresponding to non-zero entries in the last row of $\mathbf{C}_d$. Given that the ciphertext $\mathbf{C}_{d-2} \cdot (\ldots \cdot (\mathbf{C}_2 \cdot \mathbf{C}_1))$ has sparsity $k'_{\mathsf{right}} = k_{\mathsf{init}}^{d-2}$, this multiplication of a k_{init}-by-$(n+1)$ dimensional matrix with another ciphertext matrix incurs $k_{\mathsf{init}} \cdot k_{\mathsf{init}} \cdot k'_{\mathsf{right}} = k_{\mathsf{init}}^d$ operations in $\mathbb{F}_q$.

We push this optimization further, all the way to inspecting only the k_{init}^{d-1} rows of $\mathbf{C}_1$ needed to compute the final output. For any $i \in [d]$, the intermediate ciphertext output by the ith call to Mul will be a $k_{\mathsf{init}}^{d-i-1}$-by-$(n+1)$ dimensional matrix with sparsity k_{init}^{i+1}, which we can compute in k_{init}^d operations. Taken together, all $d-1$ calls to Mul require $(d-1) \cdot k_{\mathsf{init}}^d$ operations in $\mathbb{F}_q$, independent of the secret dimension n.

Incorporating homomorphic additions into this framework is seamless: to perform any number of Adds between the ith and the $(i+1)^{\text{th}}$ level of multiplications, it suffices to add those rows of the ciphertext matrices that the subsequent operations will touch. Since, after i of the d total calls to Mul, each ciphertext is a $k_{\mathsf{init}}^{d-i-1}$-by-$(n+1)$ dimensional matrix with sparsity k_{init}^{i+1}, calls to Add each require $k_{\mathsf{init}}^{d-i-1} \cdot k_{\mathsf{init}}^{i+1} = k_{\mathsf{init}}^d$ operations in $\mathbb{F}_q$—again, independent of n. These additions worsen ciphertext sparsity, which affects the cost of future operations. If all calls to Mul are completed and only Adds remain, we can immediately run the $\mathsf{Compact}$ algorithm and perform these additions on the LHE ciphertexts.

Finally, the number of LHE operations required by $\mathsf{Compact}$ is linear in the sparsity of the ciphertext it is given as input. That is, performing degree-d multiplications, followed by M total additions, on ciphertexts with initial sparsity k_{init} requires $(d-1) \cdot M \cdot k_{\mathsf{init}}^d$ operations in $\mathbb{F}_q$, followed by $M \cdot k_{\mathsf{init}}^d$ LHE additions. By Claim B.2, on LPN sparsity parameter k, we have $k_{\mathsf{init}} = (k+1)^2$. In the initial call to Expand, we can materialize only the needed rows of each matrix $\mathbf{C}_i$ (for $i \in [d]$), of which there are at most k_{init}^{d-1} (for the case of $i = 1$). So, on LPN sparsity parameter k, each call to Expand requires at most $(k+1)^2 \cdot (k+1)^{2(d-1)} = (k+1)^{2d}$ operations. □

C Additional Material on Optimizations and Batching

C.1 Syntax for Batch Somewhat Homomorphic Encryption

To allow for batch encryption and batch evaluation in our SHE schemes, we extend their syntax to include two "packing parameters" $t_{\mathsf{Enc}}, t_{\mathsf{Eval}} \in \mathbb{N}$. Then, the encryption algorithm can produce

"packed" ciphertexts encrypting any number $t_0 \leq t_{\mathsf{Enc}}$ of values, and the evaluation algorithm can evaluate any number $t_1 \leq t_{\mathsf{Eval}}$ of functions on such packed ciphertexts, producing a packed encryption of their outputs.

Definition C.1 (Somewhat Homomorphic Encryption with "Batch Encryption" and "Batch Evaluation")**.** Given a key space $\mathcal{K}$, a message space $\mathcal{M}$, and a ciphertext space $\mathcal{C}$, a *somewhat homomorphic encryption scheme with batch encryption and batch evaluation* (batch-SHE) is a tuple of four polynomial-time algorithms:

- $\mathsf{BatchGen}(1^\lambda, 1^\tau, 1^{t_{\mathsf{Enc}}}, 1^{t_{\mathsf{Eval}}}) \to (\mathsf{sk}, \mathsf{ek})$, a randomized algorithm that takes as input a security parameter $\lambda \in \mathbb{N}$, a functionality parameter $\tau \in \mathbb{N}$, and two packing parameters $t_{\mathsf{Enc}}, t_{\mathsf{Eval}} \in \mathbb{N}$, and outputs a secret key $\mathsf{sk} \in \mathcal{K}$ and an evaluation key $\mathsf{ek} \in \mathcal{K}$.
- $\mathsf{BatchEnc}(\mathsf{sk}, \mathbf{v}) \to \mathsf{ct}$, a randomized algorithm that takes as input a secret key $\mathsf{sk} \in \mathcal{K}$ and a message vector $\mathbf{v} \in \mathcal{M}^t$ for any $1 \leq t \leq t_{\mathsf{Enc}}$, and outputs a "packed" ciphertext $\mathsf{ct} \in \mathcal{C}$.
- $\mathsf{BatchEval}(\mathsf{ek}, f_1, \ldots, f_t, \mathsf{ct}_1, \ldots, \mathsf{ct}_m) \to \mathsf{ct}_{\mathsf{out}}$, a deterministic algorithm that takes as input an evaluation key $\mathsf{ek} \in \mathcal{K}$, t functions $f_1, \ldots, f_t : \mathcal{M}^\ell \to \mathcal{M}$ for any $1 \leq t \leq t_{\mathsf{Eval}}$, and any $m \in \mathbb{N}$ "packed" ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m \in \mathcal{C}$ and outputs a "packed" ciphertext $\mathsf{ct}_{\mathsf{out}} \in \mathcal{C}$.
- $\mathsf{BatchDec}(\mathsf{sk}, \mathsf{ct}) \to \mathbf{v}$, a deterministic algorithm that takes as input a secret key $\mathsf{sk} \in \mathcal{K}$ and a ciphertext $\mathsf{ct} \in \mathcal{C}$ and outputs a message vector $\mathbf{v} \in \mathcal{M}^*$.

We define a batch-SHE scheme relative to a correctness failure probability $\epsilon = \epsilon(\lambda) \in [0, 1]$ and a function class $\mathcal{F}_\tau \subseteq \{f : \mathcal{M}^* \to \mathcal{M}\}$, which comprises the computations that can be homomorphically performed on ciphertexts. Given such a failure probability ϵ and a function class $\mathcal{F}_\tau$, we require a batch-SHE scheme to satisfy correctness, semantic security, and compactness.

Correctness. For all parameters $\lambda \in \mathbb{N}$, $\tau \in \mathbb{N}$, $t_{\mathsf{Enc}} \in \mathbb{N}$, and $t_{\mathsf{Eval}} \in \mathbb{N}$, for any $t_0 \in [t_{\mathsf{Enc}}]$ and $t_1 \in [t_{\mathsf{Eval}}]$ and $m = \mathrm{poly}(\lambda)$, for all t_1 functions $f_1, \ldots, f_{t_1} : \mathcal{M}^{m \cdot t_0} \to \mathcal{M}$ in the class $\mathcal{F}_\tau$, and for all $t_1 \cdot m$ vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(t_1 \cdot m)} \in \mathcal{M}^{t_0}$, let $\epsilon = \epsilon(\lambda) \in [0, 1]$ be the correctness failure probability. For each $i \in [t_1]$, we require that the following quantity is at most ϵ:

$$\Pr\left[\, \mathsf{BatchDec}(\mathsf{sk}, \mathsf{ct})_i \neq y_i \;\middle|\; \begin{array}{rl} \mathsf{sk}, \mathsf{ek} & \leftarrow \mathsf{BatchGen}(1^\lambda, 1^\tau, 1^{t_{\mathsf{Enc}}}, 1^{t_{\mathsf{Eval}}}) \\ \mathsf{ct}_i & \leftarrow \mathsf{BatchEnc}(\mathsf{sk}, v_1^{(i)}, \ldots, v_{t_0}^{(i)}) \text{ for } i \in [t_1 m] \\ \mathsf{ct} & \leftarrow \mathsf{BatchEval}(\mathsf{ek}, f_1, \ldots, f_{t_1}, \mathsf{ct}_1, \ldots, \mathsf{ct}_{t_1 m}) \end{array} \right],$$

where we write $y_i = f_i(v_1^{((i-1)m+1)}, \ldots, v_{t_0}^{((i-1)m+1)}, \ldots, v_1^{(im)}, \ldots, v_{t_0}^{(im)})$.

Semantic security. For all parameters $\lambda \in \mathbb{N}$, $\tau \in \mathbb{N}$, $t_{\mathsf{Enc}} = \mathrm{poly}(\lambda) \in \mathbb{N}$, and $t_{\mathsf{Eval}} = \mathrm{poly}(\lambda) \in \mathbb{N}$, for any $t_0 \in [t_{\mathsf{Enc}}]$ and any number of messages $m = \mathrm{poly}(\lambda) \in \mathbb{N}$, and for any sets of $2m$ vectors $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}, \mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)} \in \mathcal{M}^{t_0}$, their encryptions are computationally indistinguishable:

$$\left\{ \begin{array}{c} \mathsf{ek}, \mathsf{BatchEnc}(\mathsf{sk}, u_1^{(1)}, \ldots, u_{t_0}^{(1)}), \\ \ldots, \mathsf{BatchEnc}(\mathsf{sk}, u_1^{(m)}, \ldots, u_{t_0}^{(m)}) \end{array} \;\middle|\; \mathsf{sk}, \mathsf{ek} \leftarrow \mathsf{BatchGen}(1^\lambda, 1^\tau, 1^{t_{\mathsf{Enc}}}, 1^{t_{\mathsf{Eval}}}) \right\}$$
$$\overset{c}{\approx} \left\{ \begin{array}{c} \mathsf{ek}, \mathsf{BatchEnc}(\mathsf{sk}, v_1^{(1)}, \ldots, v_{t_0}^{(1)}), \\ \ldots, \mathsf{BatchEnc}(\mathsf{sk}, v_1^{(m)}, \ldots, v_{t_0}^{(m)}) \end{array} \;\middle|\; \mathsf{sk}, \mathsf{ek} \leftarrow \mathsf{BatchGen}(1^\lambda, 1^\tau, 1^{t_{\mathsf{Enc}}}, 1^{t_{\mathsf{Eval}}}) \right\}.$$

Compactness. There exists a polynomial $p(\cdot)$ such that, for every $\tau \in \mathbb{N}$, there exists a constant $c \in \mathbb{N}$ where, for all parameters $\lambda > c$, $t_{\mathsf{Enc}} \in \mathbb{N}$, and $t_{\mathsf{Eval}} \in \mathbb{N}$, for any $t_0 \in [t_{\mathsf{Enc}}]$, $t_1 \in [t_{\mathsf{Eval}}]$, and $m = \mathrm{poly}(\lambda)$, for all t_1 functions $f_1, \ldots, f_{t_1} : \mathcal{M}^{m \cdot t_0} \to \mathcal{M}$ in the class $\mathcal{F}_\tau$, and for all $t_1 \cdot m$ vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(t_1 \cdot m)} \in \mathcal{M}^{t_0}$, let:

$$\begin{aligned} \mathsf{sk}, \mathsf{ek} &\leftarrow \mathsf{BatchGen}(1^\lambda, 1^\tau, 1^{t_{\mathsf{Enc}}}, 1^{t_{\mathsf{Eval}}}) \\ \mathsf{ct}_i &\leftarrow \mathsf{BatchEnc}(\mathsf{sk}, v_1^{(i)}, \ldots, v_{t_0}^{(i)}) \text{ for } i \in [t_1 \cdot m] \\ \mathsf{ct}_{\mathsf{out}} &\leftarrow \mathsf{BatchEval}(\mathsf{ek}, f_1, \ldots, f_{t_1}, \mathsf{ct}_1, \ldots, \mathsf{ct}_{t_1 \cdot m}). \end{aligned}$$

Then, the bit length of each of the ciphertexts ct_i for $i \in [t_1 \cdot m]$ is at most $t_0 \cdot p(\lambda)$, independent of τ. The bit length of the ciphertext $\mathsf{ct}_{\mathsf{out}}$ is at most $t_1 \cdot p(\lambda)$, independent of τ.

C.2 Additional Material for Section 4.1

We state and prove Lemma C.2.

Lemma C.2 (Sparse LPN with a reused $\mathbf{A}$-matrix)**.** *On security parameter $\lambda \in \mathbb{N}$, under (n, q, δ, k)-sparse LPN, for any $m = \mathrm{poly}(\lambda)$ and $\ell = \mathrm{poly}(\lambda) \in \mathbb{N}$, the following distributions are computationally indistinguishable:*

$$\left\{ (\mathbf{A}, \mathbf{AS} + \mathbf{E}) : \begin{array}{r} \mathbf{A} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,m,n,q} \\ \mathbf{S} \xleftarrow{\mathrm{R}} \mathbb{F}_q^{n \times \ell} \\ \mathbf{E} \leftarrow \mathsf{RandBern}_{n^{-\delta},q}^{m \times \ell} \end{array} \right\} \overset{c}{\approx} \left\{ (\mathbf{A}, \mathbf{U}) : \begin{array}{r} \mathbf{A} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,m,n,q} \\ \mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{F}_q^{m \times \ell} \end{array} \right\}.$$

Proof. Fix any $m, \ell = \mathrm{poly}(\lambda)$. For the sake of contradiction, let there be a polynomial-time algorithm $\mathcal{A}$ that distinguishes between the two distributions with advantage $\epsilon_{\mathsf{adv}} \geq 1/\mathrm{poly}(\lambda)$. That is, when given a sample from the left distribution, $\mathcal{A}$ outputs "1" with probability p_{left}; when given a sample from the right distribution, $\mathcal{A}$ outputs "1" with probability p_{right}; and it must be that $|p_{\text{left}} - p_{\text{right}}| \geq \epsilon_{\mathsf{adv}}$.

Now, consider the sequence of $\ell + 1$ distributions $\mathcal{D}_0, \ldots, \mathcal{D}_\ell$ defined as:

$$\forall i \in \{0, 1, \ldots, \ell\}, \quad \mathcal{D}_i = \left\{ \left(\mathbf{A}, \begin{bmatrix} \mathbf{AS} + \mathbf{E} & ||\mathbf{U} \end{bmatrix}\right) : \begin{array}{r} \mathbf{A} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,m,n,q} \\ \mathbf{S} \xleftarrow{\mathrm{R}} \mathbb{F}_q^{n \times (\ell - i)} \\ \mathbf{E} \leftarrow \mathsf{RandBern}_{n^{-\delta},q}^{m \times (\ell - i)} \\ \mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{F}_q^{m \times i} \end{array} \right\}.$$

Let p_i denote the probability of algorithm $\mathcal{A}$ outputting "1" given an input from distribution $\mathcal{D}_i$. Since distribution $\mathcal{D}_0$ is equal to the left distribution above, we have $p_{\text{left}} = p_0$. Similarly, since distribution $\mathcal{D}_\ell$ is equal to the right distribution above, we have $p_{\text{right}} = p_\ell$. As a result, $|p_0 - p_\ell| \geq \epsilon_{\mathsf{adv}}$, so there must exist some index $i \in [\ell]$ such that $|p_i - p_{i-1}| \geq \epsilon_{\mathsf{adv}}/\ell$.

Since distributions $\mathcal{D}_i$ and $\mathcal{D}_{i-1}$ differ only in the ith column of the second component being either a sparse-LPN sample or truly random, this means that algorithm $\mathcal{A}$ can break the (n, q, δ, k)-sparse LPN assumption with probability $\epsilon_{\mathsf{adv}}/\ell = O(1/\mathrm{poly}(\lambda))$. This is a contradiction. □

Construction C.3 (Packed El-Gamal over $\mathbb{Z}_q$ with ciphertext compression)**.** Parameterized by plaintext modulus $q \in \mathbb{N}$, packing parameter $t \in \mathbb{N}$, and correctness parameter $c \in \mathbb{N}$. Let the algorithm $\mathsf{Sort} : \mathbb{Z}_q^q \to \mathbb{Z}_q^q$ output a sorted copy of the list of values it is given. Instantiate PRF using a pseudorandom function from DDH.

$\mathsf{LHE.Gen}(1^\lambda, 1^\tau) \to (\mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}})$

- Let $B := q^2 \cdot (\tau + 1)$ and $T := 6\lambda^c \cdot q^2 \cdot t \cdot (B + 1)$. Let g be a generator of a group $\mathbb{G}$ in which DDH is hard, and where $\mathsf{order}(g) > B + (q+1)\lambda T$. Let $p := \mathsf{order}(g)$.
- Pick any $\mathsf{PRF} : \mathcal{K} \times \mathbb{G} \to \{0,1\}^{\log(T)}$. Sample $\mathsf{seed} \xleftarrow{\mathrm{R}} \mathcal{K}$. Sample $z_1, \ldots, z_t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$.
- Output $\mathsf{sk}_{\mathsf{LHE}} := (\mathsf{PRF}, \mathsf{seed}, g, p, z_1, \ldots, z_t)$ and $\mathsf{ek}_{\mathsf{LHE}} := (\mathsf{PRF}, \mathsf{seed}, B, T, g)$.

$\mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, (\mu_1, \ldots, \mu_t) \in \mathbb{Z}_q^t) \to \mathsf{ct}$

- Parse $\mathsf{sk}_{\mathsf{LHE}}$ as $(\mathsf{PRF}, \mathsf{seed}, g, p, z_1, \ldots, z_t)$. Sample $r_1, \ldots, r_t \xleftarrow{\mathrm{R}} \mathbb{Z}_p$.
- Output $\mathsf{ct} := \begin{pmatrix} g^{r_1} & g^{r_1 z_1 + \mu_1} & g^{r_1 z_2} & \ldots & g^{r_1 z_t} \\ g^{r_2} & g^{r_2 z_1} & g^{r_2 z_2 + \mu_2} & \ldots & g^{r_2 z_t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g^{r_t} & g^{r_t z_1} & g^{r_t z_2} & \ldots & g^{r_t z_t + \mu_t} \end{pmatrix}$.

$\mathsf{LHE.Eval}(\mathsf{ek}_{\mathsf{LHE}}, f_1 : \mathbb{Z}_q^\tau \to \mathbb{Z}_q, \ldots, f_t : \mathbb{Z}_q^\tau \to \mathbb{Z}_q, \mathsf{ct}_1, \ldots, \mathsf{ct}_\tau) \to \mathsf{ct}'$

- For $j \in [t]$, parse the affine function f_j as $f_j(x_1, \ldots, x_\tau) = c_{j,0} + \sum_{k=1}^{\tau} c_{j,k} \cdot x_k$.
- For $k \in [\tau]$, parse ct_k as the matrix of group elements $\{g_{k,\ell,j}\}_{\ell \in \{0,\ldots,t\}, j \in [t]}$.
- For $\ell \in \{0, \ldots, t\}$, let $h_\ell := g^{c_{\ell,0}} \cdot \prod_{k=1}^{\tau} \prod_{j=1}^{t} g_{k,\ell,j}^{c_{j,k}}$, where we take $c_{0,0} := 0$.
- Output $\mathsf{ct}' := \mathsf{Shrink}(\mathsf{ek}_{\mathsf{LHE}}, h_0, \ldots, h_t)$.

$\mathsf{LHE.Dec}(\mathsf{sk}_{\mathsf{LHE}}, \mathsf{ct}) \to (\mu_1, \ldots, \mu_t) \in \mathbb{Z}_q^t$

- Parse $\mathsf{sk}_{\mathsf{LHE}}$ as $(\mathsf{PRF}, \mathsf{seed}, g, p, z_1, \ldots, z_t)$ and parse ct as $(h, u_1, \ldots, u_t)$.
- Output $(w_1, \ldots, w_t)$, where $w_\ell := \mathsf{Unshrink}(\mathsf{PRF}, \mathsf{seed}, g, h^{z_\ell}, u_\ell)$ for $\ell \in [t]$.

$\mathsf{Shrink}(\mathsf{ek}_{\mathsf{LHE}}, h_0 \in \mathbb{G}, \ldots, h_t \in \mathbb{G}) \to \mathsf{ct}$

- Parse $\mathsf{ek}_{\mathsf{LHE}}$ as $(\mathsf{PRF}, \mathsf{seed}, B, T, g)$.
- If there exists $\ell \in [t]$ such that any of the following 3 conditions hold, output $\perp$:
  1. there exists $b \in \{0, \ldots, B\}$ such that $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{-b}) = 0$.
  2. there exists $\mu \in \mathbb{Z}_q$ such that, for all $b \in [\lambda \cdot T]$, $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot b + \mu}) \neq 0$.
  3. there exist distinct $\mu, \mu' \in \mathbb{Z}_q$ such that $\left|\mathsf{pad}_{\ell,\mu} - \mathsf{pad}_{\ell,\mu'}\right| \leq 1$, where, for all $m \in \mathbb{Z}_q$, we define $\mathsf{pad}_{\ell,m}$ to be the smallest non-negative integer for which $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot \mathsf{pad}_{\ell,m} + m}) = 0$.
- Let $u_\ell \in \mathbb{Z}_q$ be the index of $\mathsf{pad}_{\ell,0}$ in $\mathsf{Sort}(\mathsf{pad}_{\ell,0}, \ldots, \mathsf{pad}_{\ell,q-1})$. Output $(h_0, u_1, \ldots, u_t)$.

$\mathsf{Unshrink}(\mathsf{PRF} : \mathcal{K} \times \mathbb{G} \to \{0,1\}^{\log(T)}, \mathsf{seed} \in \mathcal{K}, g \in \mathbb{G}, h' \in \mathbb{G}, u \in \mathbb{Z}_q) \to \mathbb{Z}_q$

- For $\mu \in \mathbb{Z}_q$, let pad_μ be the smallest non-negative integer for which $\mathsf{PRF}(\mathsf{seed}, h' \cdot g^{q \cdot \mathsf{pad}_\mu + \mu}) = 0$.
- Output $i \in \mathbb{Z}_q$ so that pad_i is the uth element in $\mathsf{Sort}(\mathsf{pad}_0, \ldots, \mathsf{pad}_{q-1})$.

Figure 3: Construction of linearly homomorphic encryption over $\mathbb{Z}_q$ from DDH.

C.3 Additional Material for Section 4.2

We give the construction described informally in Section 4.2 in Construction C.3.

Let (c, q, t) be the parameters of Lemma 4.4. We prove that Construction C.3, instantiated with these parameters, gives a correct and semantically secure LHE scheme with message space $\mathbb{Z}_q$ and correctness failure probability λ^{-c}. The claimed efficiency follows by construction, and directly implies that Construction C.3 is compact.

Proof. We prove correctness and security separately.

Correctness. To prove correctness, we first show that the probability that $\mathsf{LHE.Eval}$ outputs $\perp$ is at most λ^{-c} (Claim C.4). Then, we show that decryption in Construction C.3 always recovers the correct output of the t affine functions $f_1, \ldots, f_t$ evaluated on the encrypted values, conditioned on the event that $\mathsf{LHE.Eval}$ did not output $\perp$ (Claim C.5).

Claim C.4. For all $\lambda \in \mathbb{N}$, $\tau = \mathrm{poly}(\lambda) \in \mathbb{N}$, $q = \mathrm{poly}(\lambda) \in \mathbb{N}$, constant $c \in \mathbb{N}$, and $t = \mathrm{poly}(\lambda) \in \mathbb{N}$, for all τ messages $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\tau)} \in \mathbb{Z}_q^t$, and for all t affine functions $f_1, \ldots, f_t : \mathbb{Z}_q^\tau \to \mathbb{Z}_q$, it holds that:

$$\Pr\left[\mathsf{ct}_{\mathsf{out}} = \perp \;\middle|\; \begin{array}{rcl} \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen} & (1^\lambda, 1^\tau) \\ \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc} & (\mathsf{sk}_{\mathsf{LHE}}, \mathbf{v}^{(i)}),\ \forall\, i \in [\tau] \\ \mathsf{ct}_{\mathsf{out}} & \leftarrow \mathsf{LHE.Eval} & (\mathsf{ek}_{\mathsf{LHE}}, f_1, \ldots, f_t, \\ & & \mathsf{ct}_1, \ldots, \mathsf{ct}_\tau) \end{array}\right] \leq \lambda^{-c}.$$

Proof. Let the evaluation key $\mathsf{ek}_{\mathsf{LHE}}$, the affine functions $f_1, \ldots, f_t$, and the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\tau$ be as defined in the claim statement. Let $h_0, \ldots, h_t \in \mathbb{G}$ be the inputs passed to Shrink by $\mathsf{LHE.Eval}(\mathsf{ek}_{\mathsf{LHE}}, f_1, \ldots, f_t, \mathsf{ct}_1, \ldots, \mathsf{ct}_\tau)$. Let $T := 6\lambda^c \cdot q^2 \cdot t \cdot (B+1)$. The $\mathsf{Shrink}(\mathsf{ek}_{\mathsf{LHE}}, h_0, \ldots, h_t)$ algorithm outputs $\perp$ if and only if one of 3 events occur:

1. **Event** E_1**:** there exists some $\ell \in [t]$ and some $b \in \{0, \ldots, B\}$ such that $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{-b}) = 0$.
2. **Event** E_2**:** there exists some $\ell \in [t]$ and some $\mu \in \mathbb{Z}_q$ such that, for all $b \in [\lambda \cdot T]$, it holds that $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot b + \mu}) \neq 0$.
3. **Event** E_3**:** there exists some $\ell \in [t]$ and two distinct $\mu, \mu' \in \mathbb{Z}_q$ such that $\left|\mathsf{pad}_{\ell,\mu} - \mathsf{pad}_{\ell,\mu'}\right| \leq 1$, where $\mathsf{pad}_{\ell,\mu}$ is the smallest non-negative integer such that $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot \mathsf{pad}_{\ell,\mu} + \mu}) = 0$ and $\mathsf{pad}_{\ell,\mu'}$ is the smallest non-negative integer such that $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot \mathsf{pad}_{\ell,\mu'} + \mu'}) = 0$.

Checking whether these events occur can be done in polynomial time. Since the group order is larger than $B + (q+1)\lambda T$, the events E_1 and E_2 are non-overlapping. We analyze each event's probability.

Analyzing E_1**.** Since PRF is a pseudorandom function with output bitlength $\log(T)$, for each $\ell \in [t]$ and $b \in \{0, \ldots, B\}$, we have that

$$\Pr\left[\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{-b}) = 0\right] \leq \frac{1}{T} + \mathrm{negl}(\lambda).$$

We know that $t = \mathrm{poly}(\lambda)$ and $B = \mathrm{poly}(\lambda)$. So, by a union bound over all $\ell \in [t]$, $b \in \{0, \ldots, B\}$, it holds that

$$\Pr[E_1] \leq \frac{t \cdot (B+1)}{T} + \mathrm{negl}(\lambda) = \frac{1}{6\lambda^c \cdot q^2} + \mathrm{negl}(\lambda).$$

Analyzing E_2. Assume for the sake of contradiction that, with non-negligible probability, there exists some $\ell \in [t]$ and some $\mu \in \mathbb{Z}_q$ such that $\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot b + \mu}) \neq 0$ for all $b \in [\lambda \cdot T]$. We will construct an efficient distinguisher that wins with non-negligible advantage in the PRF security game. Given a generator g for the group $\mathbb{G}$, our distinguisher works as follows:

1. it samples $r_1, \ldots, r_t \xleftarrow{\mathrm{R}} \mathbb{Z}_{\mathsf{order}(g)}$.
2. for $\ell \in [t]$, $\mu \in \mathbb{Z}_q$, it asks for the evaluation of the PRF at point $g^{r_\ell + q \cdot b + \mu}$, for all $b \in [\lambda T]$. If there exists some ℓ and μ such that all λT evaluations are non-zero, it outputs "1". Else, it outputs "0".

Here, we observe that each of the inputs that the distinguisher passes to the PRF are distributed identically to $h_\ell \cdot g^{q \cdot b + \mu}$, for $\ell \in [t]$, $\mu \in \mathbb{Z}_q$, and $b \in [\lambda T]$. So, when the distinguisher is interacting with the PRF, by our assumption it must be that the distinguisher outputs "1" with non-negligible probability. However, when the distinguisher is interacting with a truly random function f_{rand} from the space $\mathcal{F} : \mathbb{G} \to \{0,1\}^{\log(T)}$, it outputs "1" only with neligible probability: namely, for each $\ell \in [t]$, $\mu \in \mathbb{Z}_q$,

$$\begin{aligned}
\Pr_{f_{\mathsf{rand}} \xleftarrow{\mathrm{R}} \mathcal{F}} \left[\forall\, b \in [\lambda T],\ f_{\mathsf{rand}}(g^{r_\ell + q \cdot b + \mu}) \neq 0 \right] &\leq \prod_{b \in [\lambda \cdot T]} \Pr \left[f_{\mathsf{rand}}(g^{r_\ell + q \cdot b + \mu}) \neq 0 \right] \\
&\leq \left(1 - \frac{1}{T}\right)^{\lambda \cdot T} \\
&\leq e^{-\lambda} = \mathrm{negl}(\lambda).
\end{aligned}$$

We can then take a union bound over all choices of ℓ and μ (of which there are $\mathrm{poly}(\lambda)$). So, we have reached a contradiction with the PRF's computational security. As a result, it must be that $\Pr[E_2] \leq \mathrm{negl}(\lambda)$.

Analyzing $E_3 | \neg E_2$. For each $\ell \in [t]$ and each pair of distinct $\mu, \mu' \in \mathbb{Z}_q$, we know that if event E_2 did not occur, it must be that $\mathsf{pad}_{\ell,\mu} \leq \lambda T$ and $\mathsf{pad}_{\ell,\mu'} \leq \lambda T$. Then, as the order of the group generated by g is bigger than $(q+1)\lambda T$, each of the group elements in the sets $\{h_\ell \cdot g^{q \cdot b + \mu}\}_{b \in [\lambda T]}$ and $\{h_\ell \cdot g^{q \cdot b + \mu'}\}_{b \in [\lambda T]}$ must be distinct. Since PRF is a PRF with output bitlength $\log(T)$, we then have that

$$\begin{aligned}
\Pr \left[\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot \mathsf{pad}_{\ell,\mu} + \mu'}) = 0 | \neg E_2 \right] &\leq \frac{1}{T} + \mathrm{negl}(\lambda) \\
\Pr \left[\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot (\mathsf{pad}_{\ell,\mu} - 1) + \mu'}) = 0 | \neg E_2 \right] &\leq \frac{1}{T} + \mathrm{negl}(\lambda) \\
\Pr \left[\mathsf{PRF}(\mathsf{seed}, h_\ell \cdot g^{q \cdot (\mathsf{pad}_{\ell,\mu} + 1) + \mu'}) = 0 | \neg E_2 \right] &\leq \frac{1}{T} + \mathrm{negl}(\lambda).
\end{aligned}$$

As a result,

$$\Pr \left[\left| \mathsf{pad}_{\ell,\mu} - \mathsf{pad}_{\ell,\mu'} \right| \leq 1 | \neg E_2 \right] \leq \frac{3}{T} + \mathrm{negl}(\lambda).$$

We know that $t = \mathrm{poly}(\lambda)$ and that $q = \mathrm{poly}(\lambda)$. So, by a union bound over all $\ell \in [t]$ and all $\binom{q}{2}$ pairs of $\mu, \mu' \in \mathbb{Z}_q$, it holds that

$$\Pr[E_3 | \neg E_2] \leq \frac{3t \cdot q^2}{T} + \mathrm{negl}(\lambda) \leq \frac{1}{2\lambda^c \cdot (B+1)} + \mathrm{negl}(\lambda).$$

By a final union bound and the law of total probability, it must then be that the probability that $\mathsf{LHE.Eval}$ outputs $\perp$ is at most:

$$\begin{aligned}
\Pr[E_1 \cup E_2 \cup E_3] &\leq \Pr[E_1] + \Pr[E_2 \cup E_3] \\
&= \Pr[E_1] + \Pr[E_2 \cup E_3 | E_2] \cdot \Pr[E_2] + \Pr[E_2 \cup E_3 | \neg E_2] \cdot \Pr[\neg E_2] \\
&\leq \Pr[E_1] + \Pr[E_2] + \Pr[E_3 | \neg E_2] \\
&\leq \frac{1}{6\lambda^c \cdot q^2} + \frac{1}{2\lambda^c \cdot (B+1)} + \mathrm{negl}(\lambda) \\
&\leq \frac{1}{\lambda^c}.
\end{aligned}$$

□

Claim C.5. For all $\lambda \in \mathbb{N}$, $\tau = \mathrm{poly}(\lambda) \in \mathbb{N}$, $q = \mathrm{poly}(\lambda) \in \mathbb{N}$, constant $c \in \mathbb{N}$, and $t = \mathrm{poly}(\lambda) \in \mathbb{N}$, for all τ messages $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\tau)} \in \mathbb{Z}_q^t$, and for all t affine functions $f_1, \ldots, f_t : \mathbb{Z}_q^\tau \to \mathbb{Z}_q$, define

$$\mathbf{w} := \left(f_1(v_1^{(1)}, \ldots, v_1^{(\tau)}), \ldots, f_t(v_t^{(1)}, \ldots, v_t^{(\tau)}) \right).$$

Then, it holds that:

$$\Pr\left[\begin{array}{c|rll} \mathsf{LHE.Dec}(\mathsf{sk_{LHE}}, \mathsf{ct_{out}}) & \mathsf{sk_{LHE}}, \mathsf{ek_{LHE}} & \leftarrow \mathsf{LHE.Gen} & (1^\lambda, 1^\tau) \\ \neq \mathbf{w} & \mathsf{ct}_i & \leftarrow \mathsf{LHE.Enc} & (\mathsf{sk_{LHE}}, \mathbf{v}^{(i)}),\ \forall\, i \in [\tau] \\ & \mathsf{ct_{out}} & \leftarrow \mathsf{LHE.Eval} & (\mathsf{ek_{LHE}}, f_1, \ldots, f_t, \\ & & & \mathsf{ct}_1, \ldots, \mathsf{ct}_\tau) \\ & \mathsf{ct_{out}} & \neq \perp & \end{array}\right] = 0.$$

Proof. Let $(\mathsf{sk_{LHE}}, \mathsf{ek_{LHE}}) \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^\tau)$, then parse $\mathsf{sk_{LHE}}$ as the terms $(\mathsf{PRF}, \mathsf{seed}, g, p, z_1, \ldots, z_t)$, and parse $\mathsf{ek_{LHE}}$ as $(\mathsf{PRF}, \mathsf{seed}, B, T, g)$. For the rest of the correctness argument, we will reason about Construction C.3 using this key pair.

For $j \in [t]$, parse the jth affine function as

$$f_j(x_1, \ldots, x_\tau) = c_{j,0} + \sum_{k=1}^{\tau} c_{j,k} \cdot x_k \in \mathbb{Z}_q,$$

and write the corresponding affine function "lifted" to the integers as

$$\hat{f}_j(x_1, \ldots, x_\tau) = c_{j,0} + \sum_{k=1}^{\tau} c_{j,k} \cdot x_k \in \mathbb{Z}.$$

Here, for $\ell \in [t]$, we observe that

$$\hat{f}_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) = c_{\ell,0} + \sum_{k=1}^{\tau} c_{\ell,k} \cdot v_\ell^{(k)} \text{ and} \tag{13}$$

$$f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) = \hat{f}_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) \mod q. \tag{14}$$

Moreover, for all $\ell \in [t]$, it must hold that

$$0 \leq \hat{f}_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) < B. \tag{15}$$

This is because $\hat{f}_\ell$ is the lifted version of f_ℓ, which is an affine function with τ inputs and all coefficients in $\{0,\ldots,q-1\}$. In addition, all of the inputs must be in $\{0,\ldots,q-1\}$. For ease of notation, define $z_0 := 1$ and $c_{0,0} := 0$. Throughout this proof, we describe the functionality of Construction C.3, along with a line-by-line analysis highlighted in yellow.

Encryption. The $\mathsf{LHE.Enc}$ algorithm generates encryptions $\mathsf{ct}_1,\ldots,\mathsf{ct}_\tau$ of the messages $\mathbf{v}^{(1)},\ldots,\mathbf{v}^{(\tau)}$. By construction, for $k\in[\tau]$, $\ell\in\{0,\ldots,t\}$, and $j\in[t]$, we denote the entry in the ℓth column and j^{th} row of ciphertext matrix ct_k as $g^{r_{k,j}\cdot z_\ell + v_\ell^{(k)}\cdot \mathbb{1}_{\ell=j}}$.

Homomorphic evaluation. For $\ell\in\{0,\ldots,t\}$, the $\mathsf{LHE.Eval}$ algorithm computes $h_\ell = g^{c_{\ell,0}}\cdot \prod_{k=1}^{\tau}\prod_{j=1}^{t}\left(g^{r_{k,j}\cdot z_\ell + v_\ell^{(k)}\cdot\mathbb{1}_{\ell=j}}\right)^{c_{j,k}}$.

Re-writing the above expression, we see that, for $\ell\in\{0,\ldots,t\}$,

$$\begin{aligned} h_\ell &= g^{c_{\ell,0}}\cdot g^{\sum_{k=1}^{\tau}\sum_{j=1}^{t}\left(c_{j,k}\cdot r_{k,j}\cdot z_\ell + c_{j,k}\cdot v_\ell^{(k)}\cdot\mathbb{1}_{\ell=j}\right)} \\ &= g^{\left(\sum_{k=1}^{\tau}\sum_{j=1}^{t} c_{j,k}\cdot r_{k,j}\right)\cdot z_\ell + \mathbb{1}_{\ell>0}\cdot\left(\sum_{k=1}^{\tau} c_{\ell,k}\cdot v_\ell^{(k)}\right)+c_{\ell,0}}. \end{aligned}$$

Here, writing $r := \sum_{k=1}^{\tau}\sum_{j=1}^{t} c_{j,k}\cdot r_{k,j}$ and by Equation (13), we get that

$$h_0 = g^{r\cdot z_0} = g^r \tag{16}$$

$$\text{for } \ell\in[t],\quad h_\ell = g^{r\cdot z_\ell + \hat{f}_\ell(v_\ell^{(1)},\ldots,v_\ell^{(\tau)})}. \tag{17}$$

Then, $\mathsf{LHE.Eval}$ outputs $\mathsf{ct}' := \mathsf{Shrink}(\mathsf{ek}_{\mathsf{LHE}}, h_0,\ldots,h_t)$. Here, conditioned on the fact that Shrink does not output $\perp$, it must be that:

(1) for $\ell\in[t]$, $b\in\{0,\ldots,B\}$, it holds that $\mathsf{PRF}(\mathsf{seed}, h_\ell\cdot g^{-b})\neq 0$.

By Equations (15) and (17), this implies that, for $\ell\in[t]$,

$$\mathsf{PRF}(\mathsf{seed}, g^{r\cdot z_\ell})\neq 0 \text{ and} \tag{18}$$

$$\mathsf{PRF}(\mathsf{seed}, g^{r\cdot z_\ell + k})\neq 0 \text{ for } k\in\{0,\ldots,\hat{f}_\ell(v_\ell^{(1)},\ldots,v_\ell^{(\tau)})\} \tag{19}$$

(2) for $\ell\in[t]$, $\mu\in\mathbb{Z}_q$, let $\mathsf{pad}_{\ell,\mu}$ be the smallest non-negative integer such that $\mathsf{PRF}(\mathsf{seed}, h_\ell\cdot g^{q\cdot\mathsf{pad}_{\ell,\mu}+\mu})=0$. We know that $\mathsf{pad}_{\ell,\mu}\le\lambda\cdot T$. In addition, we know that there exist no distinct $\mu,\mu'\in\mathbb{Z}_q$ with $\left|\mathsf{pad}_{\ell,\mu}-\mathsf{pad}_{\ell,\mu'}\right|\le 1$.

Then, for $\ell\in[t]$, Shrink sets $u_\ell\in\mathbb{Z}_q$ to be the position of $\mathsf{pad}_{\ell,0}$ in the sorted list $\mathsf{Sort}(\mathsf{pad}_{\ell,0},\ldots,\mathsf{pad}_{\ell,q-1})$. Finally, Shrink outputs $(h_0,u_1,\ldots,u_t)$.

For $\ell \in [t]$ and $\mu \in \mathbb{Z}_q$, let $\mathsf{offset}_\ell := \left\lfloor \hat{f}_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)})/q \right\rfloor$. Then, by Equations (14) and (17),

$$\begin{aligned} h_\ell \cdot g^{q \cdot \mathsf{pad}_{\ell,\mu} + \mu} &= g^{r \cdot z_\ell + \hat{f}_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + q \cdot \mathsf{pad}_{\ell,\mu} + \mu} \\ &= g^{r \cdot z_\ell + f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + q \cdot (\mathsf{pad}_{\ell,\mu} + \mathsf{offset}_\ell) + \mu} \\ &= g^{r \cdot z_\ell + q \cdot (\mathsf{pad}_{\ell,\mu} + \mathsf{offset}_\ell) + (f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + \mu)}. \end{aligned}$$

So, by construction, $\mathsf{pad}_{\ell,\mu}$ is the smallest non-negative integer such that

$$\mathsf{PRF}(\mathsf{seed}, g^{r \cdot z_\ell + q \cdot (\mathsf{pad}_{\ell,\mu} + \mathsf{offset}_\ell) + (f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + \mu)}) = 0.$$

Now, define $\mathsf{pad}'_{\ell,\mu}$ to be the smallest non-negative integer such that $\mathsf{PRF}(\mathsf{seed}, g^{r \cdot z_\ell + q \cdot \mathsf{pad}'_{\ell,\mu} + \mu}) = 0$. Let $\mu' = f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + \mu$, computed over the integers. Here, it must be that $\mu' < 2q$. Define $e_{\ell,\mu}$ to be 0 if $\mu' < q$, and to be 1 otherwise. That is, $\mu' = (\mu' \mod q) + e_{\ell,\mu} \cdot q$.
Then, by Equation (19), because $\mathsf{pad}_{\ell,\mu} \leq \lambda T$, and because $\mathsf{order}(g) > B + (q+1)\lambda T$, it must be that

$$\begin{aligned} r \cdot z_\ell + q \cdot (\mathsf{pad}_{\ell,\mu} + \mathsf{offset}_\ell) + (f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + \mu) &= c \\ r \cdot z_\ell + q \cdot \mathsf{pad}'_{\ell,\mu' \mod q} + (\mu' \mod q) &= c \end{aligned}$$

for the same value $c \in \mathbb{Z}$ such that $\mathsf{PRF}(\mathsf{seed}, g^c) = 0$. As a result, it must be that

$$r \cdot z_\ell + q \cdot (\mathsf{pad}_{\ell,\mu} + \mathsf{offset}_\ell) + (f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + \mu) = r \cdot z_\ell + q \cdot \mathsf{pad}'_{\ell,\mu' \mod q} + (\mu' \mod q),$$

which implies that

$$\mathsf{pad}'_{\ell, f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)}) + \mu \bmod q} = \mathsf{pad}_{\ell,\mu} + \mathsf{offset}_\ell + e_{\ell,\mu}, \tag{20}$$

where $e_{\ell,\mu} \in \{0,1\}$. By Equation (20) and since there exist no distinct $\mu, \mu' \in \mathbb{Z}_q$ with $\left|\mathsf{pad}_{\ell,\mu} - \mathsf{pad}_{\ell,\mu'}\right| \leq 1$, it must be that u_ℓ is also the position of $\mathsf{pad}'_{\ell, f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)})}$ in $\mathsf{Sort}(\mathsf{pad}'_{\ell,0}, \ldots, \mathsf{pad}'_{\ell,q-1})$.

Decryption. $\mathsf{LHE.Dec}$ outputs t values in $\mathbb{Z}_q$. For $\ell \in [t]$, the ℓth value output by $\mathsf{LHE.Dec}$ corresponds to the output of $\mathsf{Unshrink}(\mathsf{PRF}, \mathsf{seed}, g, h_0^{z_\ell}, u_\ell)$. Here, for $\mu \in \mathbb{Z}_q$, $\mathsf{Unshrink}$ computes $\mathsf{pad}''_{\ell,\mu}$ to be the smallest non-negative integer such that $\mathsf{PRF}(\mathsf{seed}, h_0^{z_\ell} \cdot g^{q \cdot \mathsf{pad}''_{\ell,\mu} + \mu}) = 0$. Then, $\mathsf{Unshrink}$ outputs the value $\mu' \in \mathbb{Z}_q$ such that $\mathsf{pad}''_{\ell,\mu'}$ is the $u_\ell{}^{\text{th}}$ element in $\mathsf{Sort}(\mathsf{pad}''_{\ell,0}, \ldots, \mathsf{pad}''_{\ell,q-1})$.

For each $\ell \in [t]$, $h_0^{z_\ell} = g^{r \cdot z_\ell}$ by Equation (16). So, for each $\ell \in [t]$ and $\mu \in \mathbb{Z}_q$, $\mathsf{pad}''_{\ell,\mu}$ is exactly equal to $\mathsf{pad}'_{\ell,\mu}$. Then, since u_ℓ is the position of $\mathsf{pad}'_{\ell, f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)})}$ in $\mathsf{Sort}(\mathsf{pad}'_{\ell,0}, \ldots, \mathsf{pad}'_{\ell,q-1})$, $\mathsf{LHE.Dec}$ outputs $f_\ell(v_\ell^{(1)}, \ldots, v_\ell^{(\tau)})$, as desired. □

Semantic security. The semantic security of Construction C.3 follows directly from DDH. Let $\mathbf{u} \in \mathbb{Z}_q^t$ be the all-zeros vector. In more detail, for any $m = \mathrm{poly}(\lambda)$, for any m messages $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)} \in$

$\mathbb{Z}_q^t$, we will show that $\mathcal{D} \overset{c}{\approx} \mathcal{D}'$, where:

$$\mathcal{D} = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_m \end{array} \;\middle|\; \begin{array}{rl} \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^\tau) \\ \mathsf{ct}_j & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \mathbf{v}^{(j)}), \forall\, j \in [m] \end{array} \right\} \quad \text{and}$$

$$\mathcal{D}' = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_m \end{array} \;\middle|\; \begin{array}{rl} \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^\tau) \\ \mathsf{ct}_j & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \mathbf{u}), \forall\, j \in [m] \end{array} \right\}.$$

To do so, we use a series of m hybrid arguments. In the ith of these m hybrid arguments, we replace ciphertext ct_i from an encryption of $\mathbf{v}^{(j)}$ to an encryption of $\mathbf{u}$. That is, we define:

$$\mathcal{D}_i = \left\{ \begin{array}{c} \mathsf{ek}_{\mathsf{LHE}}, \\ \mathsf{ct}_1, \ldots, \mathsf{ct}_m \end{array} \;\middle|\; \begin{array}{rl} \mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} & \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^\tau) \\ \mathsf{ct}_j & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \mathbf{v}^{(j)}), \forall\, j \in \{1, \ldots, i\} \\ \mathsf{ct}_j & \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, \mathbf{u}), \forall\, j \in \{i+1, \ldots, m\} \end{array} \right\}$$

By construction, we have that $\mathcal{D} = \mathcal{D}_m$ and $\mathcal{D}' = \mathcal{D}_0$. Now, for any $i \in \{0, \ldots, m-1\}$, the distributions $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ differ only in the ith ciphertext. Since $\mathsf{ek}_{\mathsf{LHE}}$ consists of only a public description of a PRF (computable from the public parameters, λ, and τ), a randomly sampled PRF seed seed (that is independent of $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$), public bounds B and T (computable from the public parameters, λ, and τ), and the group generator g, these two ciphertexts are computationally indistinguishable under DDH, by the same argument as for the semantic security of El-Gamal encryption [Bon98, DGI+19, BBDP22]. So we get that $\mathcal{D}_i \overset{c}{\approx} \mathcal{D}_{i+1}$, implying that $\mathcal{D} \overset{c}{\approx} \mathcal{D}'$. □

C.4 Proof of Theorem 4.1

We describe the batch-SHE scheme (defined in Appendix C.1) from sparse LPN and DDH that proves Theorem 4.1. This scheme works by applying the optimizations from Sections 4.1 and 4.2 to the original SHE scheme in Construction 3.2. Let (n, q, δ, k) be the sparse-LPN parameters defined in the theorem statement and let $c \in \mathbb{N}$ be the correctness parameter. In addition to the usual parameters (λ, τ), the scheme is parameterized by "packing parameters" $t_{\mathsf{Enc}} = \text{poly}(\lambda) \in \mathbb{N}$ and $t_{\mathsf{Eval}} = \text{poly}(\lambda) \in \mathbb{N}$, as per the syntax in Appendix C.1.

Then, for any $t_A \in [t_{\mathsf{Enc}}]$, our scheme can produce a "packed" ciphertext that encrypts t_A values at once. Moreover, for any $t_B \in [t_{\mathsf{Eval}}]$, when homomorphically evaluating t_B multivariate polynomials $p_1, \ldots, p_{t_B} \in \mathcal{F}_\tau$, the scheme can produce "packed" output ciphertexts encrypting all t_B polynomial evaluations at once.

The scheme proceeds as follows:

1. **Key generation.** Using $\mathsf{LHE.Gen}, \mathsf{LHE.Enc}$ from the packed-El-Gamal encryption scheme in Construction C.3 (with plaintext modulus q, packing parameter t_{Eval}, and correctness parameter c) and instantiating GSWEnc as in Construction 3.2 (with parameters n, q, δ, k, and correctness parameter c), we compute:

   $\mathsf{BatchGen}(1^\lambda, 1^\tau, 1^{t_{\mathsf{Enc}}}, 1^{t_{\mathsf{Eval}}}) \to (\mathsf{sk}, \mathsf{ek})$

   - Let $\mathsf{sk}_{\mathsf{LHE}}, \mathsf{ek}_{\mathsf{LHE}} \leftarrow \mathsf{LHE.Gen}(1^\lambda, 1^n)$.
   - Let $\mathbf{s} \xleftarrow{R} \mathbb{F}_q^n$. For $\ell \in [t_{\mathsf{Enc}}]$, let $\mathbf{t}^{(\ell)} \xleftarrow{R} \mathbb{F}_q^n$.
   - For $i \in [n]$, let $\mathsf{ct}_{i,\mathsf{ek}} \leftarrow \mathsf{LHE.Enc}(\mathsf{sk}_{\mathsf{LHE}}, (s_i, \ldots, s_i))$. *(Here, per the syntax of Construction C.3, $(s_i, \ldots, s_i)$ is a vector with repeated entries of dimension t_{Eval}, the packing parameter.)*

- For $i \in [n]$, for $\ell \in [t_{\mathsf{Enc}}]$, let $\mathbf{C}_{\ell,i,\mathsf{ek}} \leftarrow \mathsf{GSWEnc}(\mathbf{s}, -t_i^{(\ell)})$.
- For $\ell \in [t_{\mathsf{Enc}}]$, let $\mathbf{C}_{\ell,n+1,\mathsf{ek}} \leftarrow \mathsf{GSWEnc}(\mathbf{s}, 1)$.
- Output

$$\mathsf{sk} := (\mathsf{sk}_{\mathsf{LHE}}, \mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(t_{\mathsf{Enc}})})$$
$$\mathsf{ek} := \left(\mathsf{ek}_{\mathsf{LHE}}, (\mathsf{ct}_{1,\mathsf{ek}}, \ldots, \mathsf{ct}_{n,\mathsf{ek}}), \{\mathbf{C}_{\ell,1,\mathsf{ek}}, \ldots, \mathbf{C}_{\ell,n+1,\mathsf{ek}}\}_{\ell \in [t_{\mathsf{Enc}}]}\right)$$

2. **Encryption.** Our encryption algorithm takes as input any $t_A \le t_{\mathsf{Enc}}$ values in $\mathbb{F}_q$ at once.

   $\mathsf{BatchEnc}(\mathsf{sk}, (\mu_1, \ldots, \mu_{t_A}) \in \mathbb{F}_q^{t_A}) \to \mathsf{ct} \in \mathbb{F}_q^{n+t_A}$

   - Parse $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(t_A)}$ from sk.
   - Sample $\mathbf{a} \xleftarrow{\mathrm{R}} \mathcal{S}_{k,1,n,q}$. (That is, $\mathbf{a}$ is a random k-sparse vector in $\mathbb{F}_q^n$.)
   - For $\ell \in [t_A]$, sample $e_\ell \leftarrow \mathsf{RandBern}_{n^{-\delta},q}$.
   - Compute the vector

$$\mathbf{b} := \mathbf{a}^\intercal \cdot \begin{bmatrix} | & | & & | \\ \mathbf{t}^{(1)} & \mathbf{t}^{(2)} & \cdots & \mathbf{t}^{(t_A)} \\ | & | & & | \end{bmatrix} + \begin{bmatrix} e_1 & e_2 & \cdots & e_{t_A} \end{bmatrix} + \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_{t_A} \end{bmatrix} \in \mathbb{F}_q^{t_A}.$$

   - Output $\mathsf{ct} := \begin{bmatrix} \mathbf{a} & || & \mathbf{b} \end{bmatrix} \in \mathbb{F}_q^{n+t_A}$.

3. **Homomorphic evaluation.** Let $t_B \le t_{\mathsf{Eval}}$ be the number of polynomials to evaluate.

   For each of the input ciphertexts, we proceed as follows: we decompose an input ciphertext

$$\mathsf{ct} = \begin{bmatrix} \mathbf{a} & || & b_1 & b_2 & \cdots & b_t \end{bmatrix} \in \mathbb{F}_q^{n+t}$$

   into the t smaller ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_t$ as follows:

$$\text{for } \ell \in [t], \quad \mathsf{ct}_\ell := \begin{bmatrix} \mathbf{a} & || & b_\ell \end{bmatrix} \in \mathbb{F}_q^{n+1}.$$

   Then, for $\ell \in [t]$, we run the Expand algorithm from Construction 3.2 on ciphertext ct_ℓ, using the key-switching key $(\mathbf{C}_{\ell,1,\mathsf{ek}}, \ldots, \mathbf{C}_{\ell,n+1,\mathsf{ek}})$ from ek. The output of Expand is a ciphertext matrix in $\mathbb{F}_q^{(n+1)\times(n+1)}$ that holds the value encrypted in the ℓth slot of ct.

   At this point, we can use the Add and Mul algorithms from Construction 3.2 to evaluate each polynomial $p_1, \ldots, p_{t_B}$ on encrypted inputs.

   At the end of this step, we hold one sparse-LPN ciphertext matrix for each of the t_B polynomials $p_1, \ldots, p_{t_B}$. For $\ell \in [t_B]$, denote the last row of the ℓth ciphertext matrix as $\begin{bmatrix} \mathbf{u}^{(\ell)} & || & v^{(\ell)} \end{bmatrix} \in \mathbb{F}_q^n \times \mathbb{F}_q$. To compact these ciphertexts, define the following t_B affine functions:

$$\text{for } \ell \in [t_B], \quad f_\ell(x_1, \ldots, x_n) := v^{(\ell)} - \sum_{i=1}^{n} u_i^{(\ell)} \cdot x_i.$$

   For $\ell \in \{t_B + 1, \ldots, t_{\mathsf{Eval}}\}$, define $f_\ell(x_1, \ldots, x_n) := 0$.

   Finally, we call $\mathsf{LHE.Eval}(\mathsf{ek}_{\mathsf{LHE}}, f_1, \ldots, f_{t_{\mathsf{Eval}}}, \mathsf{ct}_{1,\mathsf{ek}}, \ldots, \mathsf{ct}_{n,\mathsf{ek}})$ from Construction C.3. We parse its result as $(h, u_1, \ldots, u_{t_{\mathsf{Eval}}}) \in \mathbb{G} \times \mathbb{F}_q^{t_{\mathsf{Eval}}}$ and output $(h, u_1, \ldots, u_{t_B})$.

4. **Decryption.** Parse the ciphertext as $(h, u_1, \ldots, u_{t_B})$. Run $\mathsf{LHE.Dec}(\mathsf{sk}_{\mathsf{LHE}}, \cdot)$ from Construction C.3 on the given ciphertext (skipping the call to $\mathsf{Unshrink}$ for all but the first t_B entries of the message) and output its result in $\mathbb{F}_q^{t_B}$.

Analysis. We discuss the scheme's security, correctness, and compactness.

Security. Security follows from the semantic security of the LHE scheme in Construction C.3 (Lemma 4.4, proved in Appendix C.3), the semantic security of the SHE scheme in Construction 3.2 (Theorem 3.1, proved in Appendix B.1), and the hardness of sparse LPN with a re-used $\mathbf{A}$-matrix (Lemma C.2, proved in Appendix C.2).

More formally, the evaluation key output by $\mathsf{BatchGen}$ consists of:

1. an evaluation key $\mathsf{ek}_{\mathsf{LHE}}$ for the underlying LHE scheme,
2. n "redundant" LHE encryptions of the entries of secret key vector $\mathbf{s}$, and
3. for each $\ell \in [t_{\mathsf{Enc}}]$, $(n+1)$ sparse-LPN encryptions of the entries of $\left[-\mathbf{t}^{(\ell)} \quad || \quad 1\right] \in \mathbb{F}_q^{n+1}$, under secret key $\mathbf{s}$.

To prove security, we must show that, under $(n, q, \delta', (k+1)/2)$-sparse LPN, for any $t \in [t_{\mathsf{Enc}}]$ and any $m = \mathrm{poly}(\lambda)$, the evaluation key together with m ciphertexts output by $\mathsf{BatchEnc}$, on any m messages $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)} \in \mathbb{F}_q^t$, are computationally indistinguishable from the evaluation key together with m encryptions of the all-zeros vector.

Again, security follows by a hybrid argument:

- First, by the semantic security of the underlying LHE scheme, we can swap the LHE encryptions of the entries of the secret-key vector $\mathbf{s}$ with LHE encryptions of the all-zeros vector.
- Second, by the semantic security of ciphertexts produced by GSWEnc (which in turn follows from the KDM-security of sparse LPN, as proved in Theorem 3.1), for each $\ell \in [t_{\mathsf{Enc}}]$, we can swap the sparse-LPN encryptions of the entries of $\left[-\mathbf{t}^{(\ell)} \quad || \quad 1\right] \in \mathbb{F}_q^{n+1}$, under secret key $\mathbf{s}$, with sparse-LPN encryptions of the all-zeros vector under $\mathbf{s}$.
- Third, by the sparse-LPN assumption with a re-used $\mathbf{A}$-matrix (Lemma C.2), we can swap the m ciphertexts output by $\mathsf{BatchEnc}$, encrypting $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(m)}$ under secret keys $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(t)}$, with m encryptions of zero under secret keys $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(t)}$.
- Fourth, by the semantic security of ciphertexts produced by GSWEnc (which in turn follows from the KDM-security of sparse LPN, as proved in Theorem 3.1), for each $\ell \in [t_{\mathsf{Enc}}]$, we can swap the sparse-LPN encryptions of the all-zeros vector under secret key $\mathbf{s}$ back to the sparse-LPN encryptions of the entries of $\left[-\mathbf{t}^{(\ell)} \quad || \quad 1\right] \in \mathbb{F}_q^{n+1}$, under $\mathbf{s}$.
- Fifth, by the semantic security of the underlying LHE scheme, we can swap the LHE encryptions of the all-zeros vector back to LHE encryptions of the entries of secret key vector $\mathbf{s}$.

Correctness. Correctness follows from the correctness of the SHE scheme in Section 3 (Theorem 3.1, proved in Appendix B.1) and the correctness of the LHE scheme in Section 4.2 (Lemma 4.4, proved in Appendix C.3).

More formally, consider any key pair $(\mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{BatchGen}(1^\lambda, 1^\tau, 1^{t_{\mathsf{Enc}}}, 1^{t_{\mathsf{Eval}}})$. Then, on any input vector $\mathbf{v} \in \mathbb{F}_q^t$ for any $t \leq t_{\mathsf{Enc}}$, running $\mathsf{BatchEnc}(\mathsf{sk}, \mathbf{v})$ and "decomposing" the resulting ciphertext

in $\mathbb{F}_q^{n+t}$ into t ciphertexts in $\mathbb{F}_q^{n+1}$ (as described above) produces t ciphertexts that respectively encrypt each of the scalars $v_1, \ldots, v_t \in \mathbb{F}_q$ and that are of the same form as those in Construction 3.2. That is, for each $\ell \in [t]$, the message v_ℓ is Regev-encrypted under the secret key $\mathbf{t}^{(\ell)}$, with a k-sparse $\mathbf{a}$-component and error-rate $n^{-\delta}$.

Then, by the proof of Construction 3.2 (given in Appendix B), passing this ciphertext to Expand with the ℓth key-switching key produces a ciphertext matrix, as in the original SHE scheme from Construction 3.2, encrypting the message v_ℓ under secret key $\mathbf{s}$. This ciphertext is sampled from a distribution that is $((k+1)^2, (k+2) \cdot n^{-\delta})$-good with respect to the message v_ℓ (Claim B.2). At this point, as in Construction 3.2, we can perform homomorphic operations using Add and Mul (Claims B.3 and B.4). At the end, we perform "batch compaction" on the t_B resulting ciphertexts (one for each of the t_B polynomials), exactly as in Construction 3.2 instantiated with our Construction C.3 as the underlying LHE scheme.

Again as in the proof of Construction 3.2, if $k \leq \sqrt{\log \tau} - 1$ and $n \geq \tau^{2/\delta} \cdot \lambda^{c/\delta}$, then this encryption scheme can homomorphically evaluate functions in the class $\mathcal{F}_\tau$, where *each* of the t_B evaluations incurs a correctness error with probability λ^{-c}. Then, the compaction step fails with probability $\epsilon_{\mathsf{LHE}}(\lambda)$, where $\epsilon_{\mathsf{LHE}}(\lambda)$ is the decryption failure probability of the underlying LHE scheme. Here, by construction, we have that $\epsilon_{\mathsf{LHE}}(\lambda) = \lambda^{-c}$, which by a union bound gives the final correctness bound.

Efficiency. When our LHE scheme is instantiated with a group with elements of bitlength λ_{DDH}, the evaluation key consists of:

- $\text{poly}(\lambda)$ bits to specify $\mathsf{ek}_{\mathsf{LHE}}$,
- $n \cdot t_{\mathsf{Eval}} \cdot (t_{\mathsf{Eval}} + 1) \cdot \lambda_{\mathsf{DDH}}$ bits to specify the packed-El-Gamal encryptions of each of the n entries of the sparse-LPN secret $\mathbf{s}$, and
- $t_{\mathsf{Enc}} \cdot (n+1)^2 \cdot (k+1) \cdot \log n \log q$ bits to specify the t_{Enc} key-switching keys, each of which consists of $(n+1)$ matrices in $\mathbb{F}_q^{(n+1)\times(n+1)}$ that are $(k+1)$-sparse.

In total, as the sparsity $k = O(\sqrt{\log \lambda})$, $q = \text{poly}(\lambda)$, $n = \text{poly}(\lambda)$, and $\lambda_{\mathsf{DDH}} = \text{poly}(\lambda)$, this comes out to $(t_{\mathsf{Eval}}^2 + t_{\mathsf{Enc}}) \cdot \text{poly}(\lambda)$ bits.

Then, each ciphertext output by $\mathsf{BatchEnc}$ (which encrypts a vector of plaintexts in $\mathbb{F}_q^{t_A}$ for $t_A \leq t_{\mathsf{Enc}}$) consists of a vector in $\mathbb{F}_q^{n+t_A}$, whose first n entries contain exactly k non-zero values. Each such ciphertext can be represented in $t_{\mathsf{Enc}} \log q + k \log n \log q$ bits. So, encrypting t_{Enc} values in $\mathbb{F}_q$ requires $t_{\mathsf{Enc}} \log q + \text{polylog}(\lambda)$ bits. Finally, when homomorphically evaluating $t_B \in [t_{\mathsf{Eval}}]$ polynomials, per the analysis of our LHE scheme (Section 4.2), each ciphertext after homomorphic evaluation consists of $t_B \log q + \lambda_{\mathsf{DDH}}$ bits. As a result, each such ciphertext can be represented in $t_{\mathsf{Eval}} \log q + \lambda_{\mathsf{DDH}}$ bits.

We finally describe how to recover the statement of Theorem 4.1: when homomorphically evaluating t polynomials that are each m variate, on a set of tm inputs, we set the parameters $t_{\mathsf{Enc}} := t$ and $t_{\mathsf{Eval}} := t$. When this is the case:

- The evaluation key has size $t^2 \cdot \text{poly}(\lambda)$ bits,
- Each of m "packed" ciphertexts encrypting t inputs has size $t \log q + \text{polylog}(\lambda)$ bits. So, together, all ciphertexts consist of $tm \log q + m \cdot \text{polylog}(\lambda)$ bits.
- The "packed" ciphertext encrypting all t outputs has size $t \log q + \lambda_{\mathsf{DDH}}$ bits.