A Systematic Study of Sparse LWE

Aayush Jain* Huijia Lin† Sagnik Saha‡

October 8, 2024

Abstract

In this work, we introduce the sparse LWE assumption, an assumption that draws inspiration from both Learning with Errors (Regev JACM 10) and Sparse Learning Parity with Noise (Alekhnovich FOCS 02). Exactly like LWE, this assumption posits indistinguishability of $(\mathbf{A}, \mathbf{sA} + \mathbf{e} \mod p)$ from $(\mathbf{A}, \mathbf{u})$ for a random $\mathbf{u}$ where the secret $\mathbf{s}$, and the error vector $\mathbf{e}$ is generated exactly as in LWE. However, the coefficient matrix $\mathbf{A}$ in sparse LPN is chosen randomly from $\mathbb{Z}_p^{n\times m}$ so that each column has Hamming weight exactly k for some small k. We study the problem in the regime where k is a constant or polylogarithmic. The primary motivation for proposing this assumption is efficiency. Compared to LWE, the samples can be computed and stored with roughly $O(n/k)$ factor improvement in efficiency. Our results can be summarized as:

- **Foundations:** We show several properties of sparse LWE samples, including: 1) The hardness of LWE/LPN with dimension k implies the hardness of sparse LWE/LPN with sparsity k and arbitrary dimension $n \geq k$. 2) When the number of samples $m = \Omega(n \log p)$, length of the shortest vector of a lattice spanned by rows of a random sparse matrix is large, close to that of a random dense matrix of the same dimension (up to a small constant factor). 3) Trapdoors with small polynomial norm exist for random sparse matrices with dimension $n \times m = O(n \log p)$. 4) Efficient algorithms for sampling such matrices together with trapdoors exist when the dimension is $n \times m = \widetilde{\mathcal{O}}(n^2)$.
- **Cryptanalysis:** We examine the suite of algorithms that have been used to break LWE and sparse LPN. While naively many of the attacks that apply to LWE do not exploit sparsity, we consider natural extensions that make use of sparsity. We propose a model to capture all these attacks. Using this model we suggest heuristics on how to identify concrete parameters. Our initial cryptanalysis suggests that concretely sparse LWE with a modest k and slightly bigger dimension than LWE will satisfy similar level of security as LWE with similar parameters.
- **Applications:** We show that the hardness of sparse LWE implies very efficient homomorphic encryption schemes for low degree computations. We obtain the first secret key Linearly Homomorphic Encryption (LHE) schemes with *slightly super-constant*, or even *constant*, overhead, which further has applications to private information retrieval, private set intersection, etc. We also obtain secret key homomorphic encryption for arbitrary constant-degree polynomials with slightly super-constant, or constant, overhead.

We stress that our results are preliminary. However, our results make a strong case for further investigation of sparse LWE.

*Carnegie Mellon University. Email: aayushja@andrew.cmu.edu
†University of Washington. Email: rachel@cs.washington.edu
‡Carnegie Mellon University. Email: sagniks@andrew.cmu.edu

Contents

1 Introduction

The celebrated work by Regev [Reg05] proposed the Learning With Errors (LWE) assumption, postulating that noisy linear samples are indistinguishable from random,

$$(\mathbf{A}, \boldsymbol{b} = \boldsymbol{s}\mathbf{A} + \boldsymbol{e} \bmod p) \quad \approx_c \quad (\mathbf{A}, \boldsymbol{u})$$

where the coefficient matrix $\mathbf{A} \leftarrow \mathbb{Z}_p^{n \times m}$ and the secret vector $\boldsymbol{s} \leftarrow \mathbb{Z}_p^n$ are sampled randomly, the entries of the noise vector $\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{1 \times m}$ are sampled from the discrete Gaussian distribution with width σ smaller than p, and the vector $\boldsymbol{u} \leftarrow \mathbb{Z}_p^{1 \times m}$ is uniformly random.

The LWE assumption is extremely versatile, and has been the basis of a plethora of constructions. This spans the development of core cryptographic tools including key agreement, public key encryption, and signature schemes that are ready to be deployed [ACC+18, AAC+22], schemes and protocols with richer functionalities and nearly practical efficiency e.g., [HHC+23, HDCZ23, LNP22, BLNS23], to advanced primitives pushing the envelop of cryptographic feasibility, such as, Fully Homomorphic Encryption [Gen09, BV11, GSW13, Mah18a, Bra18], Attribute-Based Encryption [GVW13, BGG+14], Succinct Arguments for P [CJJ22] and many others [GKW17, WZ17, GKW18, BCM+18, Mah18b]. The hardness of the LWE problem has been extensively studied. It was shown that the hardness of LWE is implied by the worst-case hardness of certain lattice problems such as GapSVP [Reg05, Pei09, LM09, BLP+13], which is widely believed to be subexponentially hard for a range of parameters.

Many variants of the LWE problems have been explored, encompassing 1) the use of structured secrets, such as binary random secrets and entropic secrets, 2) the use of structured noise, e.g., binary random noise, uniform random noise of bounded magnitude, and rounded version known as Learning With Rounding (LWR) [BPR12], and 3) the use of structured matrices, such as those implicit in Ring LWE (RLWE) [LPR10] or middle-point LWE [RSSS17]. All these variants have significantly extended the landscape of functionalities, efficiency and security levels, and techniques supported by the LWE family of assumptions.

The Sparse Learning With Errors Problems This work proposes a new variant called the k-Sparse LWE (k-sLWE) problem. We conjecture the hardness of the LWE problem when the coefficient matrices $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$ are **random k-sparse matrices** (as opposed to random dense matrices), where each column has exactly k non-zero entries for some $k \leq n$ and every non-zero entry is sampled uniformly and randomly from $\mathbb{Z}_p$. Sparsity is the new feature – the smaller the sparsity parameter k is, the more efficient it is to store and compute the samples. Each sparse LWE sample $(\mathbf{a}_i, \mathbf{s}\mathbf{a}_i + e_i)$ contains only $k+1$ elements in $\mathbb{Z}_p$ and takes $\widetilde{O}(k \log p)$ time to compute, whereas a standard LWE sample contains $n+1$ elements and takes $\widetilde{O}(n \log p)$ time to compute. At the same time, k is crucial for the hardness of the problem. A variety of settings of k is interesting; k can be constant, logarithmic, polylogarithmic, or sublinear in the dimension n. We found that k being logarithmic or polylogarithmic is the most suitable for cryptographic applications, ensuring efficiency and strong security. While the assumption with constant k and appropriately bounded number of samples might also be useful for deriving interesting theoretical results, constant k will not provide almost exponential security when the number of samples is superlinear in the dimension n.

Sparse LWE is a natural question to study. The effect of sparsity in the coefficient matrix has long been studied in the context of the Learning Parity with Noise (LPN) problem, a close cousin of LWE, which uses *sparse* random noise, instead of *small* (but dense) Gaussian noise as in LWE. Sparse LPN further uses random k-sparse coefficient matrices as described above. Variants of the

sparse LPN assumption in the binary field $\mathbb{F}_2$ have been proposed and studied for at least a couple of decades in average-case complexity and are intimately connected to constraint satisfaction problems (see works such as [Gol00, CM01, Fei02, Ale03, MST03, AOW15, AL16, KMOW17]). The work of [ADI+17a] generalized the assumption to large fields $\mathbb{F}_q$. These assumptions have been used in a number of applications, for example [Ale03, AIK06, IKOS08, ABW10, CRR21, DIJL23]. Among them, the work of [ABW10] constructed a public-key encryption scheme from k-sparse LPN for a constant $k \geq 3$ and bounded polynomial number of samples, while the recent work of [DIJL23] constructed multi-party Homomorphic Secret Sharing and multi-party computation protocols with sublinear communication from k-sparse LPN for any polylogarithmic k and unbounded polynomial number of samples. The related assumption of local PRGs [Gol00] has also been used in a number of works including the recent construction of a program obfuscation scheme [JLS21].

It is somewhat surprising that the sparse variant of LWE has not been investigated, to the best of our knowledge. That is the purpose of this work. Sparse LWE is related to constraint satisfaction problems like sparse LPN and Goldreich's one-way function and PRG, and at the same time draws connection to lattices. It is curious and interesting to ask:

How hard is the sparse LWE problem?
What properties do sparse LWE samples have?
What can the hardness of sparse LWE enable cryptographically?

We give initial answers to these questions, and ask new questions. Our preliminary results include:

- *Foundations.* We show several properties of sparse LWE samples, including: 1) the hardness of LWE with dimension k implies the hardness of sparse LWE with sparsity k and arbitrary dimension $n \geq k$. Our reduction is oblivious to the error model and hence the results also apply to sparse LPN.

  2) When the number of samples $m = \Omega(n \log p)$, the length of the shortest vector of the lattice spanned by the rows of a random sparse matrix is large, close to that of a random dense matrix of the same dimensions (up to a small constant factor).

  3) Trapdoors with small polynomial norm exist for random sparse matrices with dimension $n \times m = O(n \log p)$. An interesting consequence of the first claim from [GPV08a] is that when $m = O(n \log p)$, the distribution $(\mathbf{A}, \mathbf{Ar} \mod p)$ for a vector $\mathbf{r} \in \mathbb{Z}_p^m$ is statistically close to random in $\mathbb{Z}_p^n$ provided $\mathbf{r}$ is chosen from a polynomially wide discrete Gaussian over $\mathbb{Z}^m$. This might be useful for the design of public-key primitives.

  4) While we don't know how to sample random sparse matrices with trapdoors when $m = O(n \log p)$, we can design an efficient algorithm for sampling such matrices together with trapdoors when the dimension is $n \times m = O(n^2 \log n \log p)$.

- *Cryptanalysis.* Examining existing attacks on the LWE problem, the structure of sparse matrices does not naively speed up combinatorial [BKW03], lattice reduction [LLL82, Sch94], or algebraic attacks [AG11]. Moreover, many attacks that would typically apply to sparse LPN become harder to apply to sparse LWE due to the fact that the noise is dense (though small). We also explore attacks that directly leverage sparsity, and show that they are ineffective when the sparsity parameter k is polylogarithmic and number of samples m is (unbounded) polynomial. Overall, our initial cryptanalysis optimistically suggests that when all other parameters are equal, namely the modulus p, the noise width σ, and polynomial number of

samples m, polylogarithmic-sparse LWE with slightly larger dimension $n' = \Theta(n)$, is comparable to LWE with dimension n. We also give some concrete parameters based on our initial cryptanalysis.

- *Applications.* The hardness of sparse LWE implies very efficient homomorphic encryption schemes for low degree computations. In particular, we obtain the first secret key Linearly Homomorphic Encryption (LHE) schemes with *slightly super-constant*, or even *constant*, overhead, which further has applications to private information retrieval, private set intersection, etc. We also obtain secret key homomorphic encryption for arbitrary constant-degree polynomials with slightly super-constant, or constant, overhead. In comparison, to the best of our knowledge, previous homomorphic encryption schemes have (large) polylogarithmic overhead [GHS12]. In particular, our LHE scheme is concretely efficient.

We emphasize that our cryptanalysis is preliminary, and more thorough study must be conducted before we can rest our confidence on the hardness of sparse LWE. Nevertheless, in order to assess the potential benefits of sparse LWE and provide motivation for further study, we make tentative suggestions for concrete parameters to facilitate comparison of efficiency.

Comparison of sparse LWE with LWE and RLWE The main motivation behind introduction of sparse LWE is efficiency. Storing and computing sparse LWE samples are roughly $\widetilde{O}(n/k)$ times more efficient compared to plain LWE. Game-changing efficiency improvements have been afforded by Ring LWE [LPR10] due to its nice properties (such as SIMD compatibility [SV14]). However, there are few settings where rings may not be ideal, for instance, when we want to access data as integer elements rather than as ring elements. A good example of this is private-information retrieval applications [HHC+23]. We believe that in those settings sparse LWE might form an appealing alternative to ring LWE.

Open Problems Our work leaves a great supply of open problems from various angles. On the complexity side, we ask if there are new attacks on sparse LWE? Can we show worst-case to average-case reductions or leakage resilience properties similar to LWE? Similarly, it is completely unclear if sparse LWE with small secrets is hard. On the concrete attacks and algorithms side, a systematic study of concrete parameters is warranted. On the applications side, it would be good to study more applications of sparse LWE beyond what we mentioned.

2 Technical Outline

We now describe our results with more context. The sLWE assumption is parameterized by a sparsity parameter k, dimension n, sample complexity m, noise parameter σ and modulus p. It posits that $(\mathbf{A}, \mathbf{sA} + \mathbf{e} \mod p)$ is indistinguishable from $(\mathbf{A}, \mathbf{u})$ where $\mathbf{A}$ is a random matrix in $\mathbb{Z}_p^{n\times m}$ such that each column is k-sparse, $\mathbf{s} \leftarrow \mathbb{Z}_p^{1\times n}$ is randomly chosen, $\mathbf{e} \in \mathbb{Z}^{1\times m}$ is generated from discrete Gaussian with width σ and $\mathbf{u} \in \mathbb{Z}_p^{1\times m}$ is randomly chosen. Our results can be divided into three categories: Foundations, Cryptanalysis and Applications.

2.1 Foundations

We start by investigating some basic properties of sparse LWE. A very natural question to ask is how much is the minimum distance of the lattice $\mathcal{L} = \mathcal{L}(\mathbf{A})$ given by $\mathcal{L} = \{\mathbf{xA} + p\mathbb{Z}^m : \mathbf{x} \in \mathbb{Z}^{1\times n}\}$ as a function of k, n, m and p. This should be as large as possible, and certainly much larger

than the norm of the error vector, so that given m sLWE samples the planted secret $\mathbf{s}$ is uniquely determined. In section 4.1, we show that when $m = \Omega(n \log p)$ as long as $k > 1$ and p is polynomial in n, the minimum distance is $\Omega(p)$. In particular, once the number of samples exceeds $8n \log n$, the minimum distance properties of $\mathcal{L}$ behave roughly similarly as in regular LWE samples.

An immediate consequence of this from the Transference theorems of Banaszczyk [Ban93] (and as argued in corollary 3.1) is that the there exist m linearly independent vectors of bounded polynomial norm in the lattice $\mathcal{L}^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0}\}$ once $m = \Omega(n \log p)$. Thus, in principle when $m = \Omega(n \log p)$, $\mathbf{A}$ satisfies nice properties similar to a dense matrix corresponding to LWE distribution. For instance, one immediate consequence from the work of [GPV08a] is that $\mathbf{A}$ can be used to extract from Gaussian sources. In other words, if $\mathbf{r}$ is chosen from a wide enough (polynomial width) discrete Gaussian, the induced distribution on $\mathbf{A}\mathbf{r}$ is statistically close to a random vector in $\mathbb{Z}_p^n$ even if $\mathbf{A}$ is public.

We ask when can we sample a random sparse matrix $\mathbf{A}$ along with such a trapdoor. While we leave sampling random sparse matrices $\mathbf{A}$ along with a short trapdoor for $m = O(n \log p)$ as a great open question, in section 6, we show that it is indeed possible to sample such sparse matrices along with lattice trapdoors for $m = O(n^2 \log n \log p)$. Such linearly independent short vectors in $\mathcal{L}^{\perp}(\mathbf{A})$ can be used to sample from a discrete Gaussian distribution on vectors in $\mathcal{L}^{\perp}(\mathbf{A})$ using lattice sampling methods from prior work such as [GPV08a, MP13]. The current approach for sampling fully random (dense) matrices along with trapdoors works by first sampling a matrix $\mathbf{A}$ at random and then setting $\mathbf{V} = \mathbf{A}\mathbf{R} + \mathbf{G}$ for a random small norm matrix $\mathbf{R}$ and the gadget matrix $\mathbf{G}$. This gives a small norm matrix $\mathbf{X}$ so that $[\mathbf{A}|\mathbf{V}]\mathbf{X} = \mathbf{G}$. Micciancio and Peikert [MP13] show how to convert $\mathbf{X}$ to a trapdoor for $[\mathbf{A}|\mathbf{V}]$. This approach does not work directly for us because unlike the case with dense matrices, $\mathbf{V} = \mathbf{A}\mathbf{R} + \mathbf{G}$ is not sparse anymore. The main technical highlight of our algorithm is a novel combinatorial approach to get around this issue and sample random sparse matrices along with trapdoors.

We then ask if we can show asymptotic polynomial time hardness of sparse LWE. It would be nice to show a result of the form: "if LWE is hard, then sparse LWE is also hard". Such results are not known even in the setting of LPN/sparse LPN which has received a lot of study. While we do not have a fully general result here, in section 5, we show quite an interesting reduction.

Specifically, sparse LWE with sparsity parameter k, dimension $n > k$, sample complexity m, prime modulus p and the noise parameter σ is harder than LWE with dimension k, sample complexity m, modulus p and the noise parameter σ. Our reduction is oblivious to the error model and also applies to the case of sparse LPN/LPN. Assuming subexponential hardness for LWE, this establishes polynomial time hardness for sparse LWE provided k is polylogarithmic. We leave showing such hardness results for constant k to future work.

2.2 Cryptanalysis

We perform some preliminary cryptanalysis of this new assumption and also suggest some concrete parameters based on that. We stand on the tall shoulders of a body of great work done in analyzing security of sparse LPN [Gol00, CM01, Fei02, Ale03, MST03, AIK06, IKOS08, ABW10, AOW15, AL16, KMOW17] as well as LWE [ACF+15, APS15, Pla18, ADRSD15, ACF+14].

In sLWE sparsity creates some interesting effects. For example, when $m \geq k\binom{n}{k} + 1$ we are guaranteed to find at least k+1 samples supported on the same set of k variables, which might be enough to distinguish a random vector from one close to the lattice. More generally, one can potentially use an algorithm that breaks LWE for some small dimension whenever there exists a set of t samples supported on less than t distinct variables, with t being sufficiently small. However, we believe that once k is large enough, we can release polynomially many samples without

compromising security. Most of our cryptanalysis deals with the case of $k = O(\log n)$ so that we do not have to worry about this effect.

We now discuss a few different types of attacks against sparse LWE. Some of our attacks are inspired by existing algorithms for sparse LPN and LWE attacks, while others leverage the sparsity of $\mathbf{A}$ more directly.

Sparse LPN Style Attacks We first observe that many of the attacks that apply to sparse LPN do not apply in this case because our error vectors are dense as opposed to being sparse. In the main attack that still might apply how it does for sparse LPN, one aims to find sparse vectors $\mathbf{x} \in \mathbb{Z}_p^m$ so that $\mathbf{Ax} = \mathbf{0}$. Such a t-sparse $\mathbf{x}$ can break sparse LPN if the error probability is less than $\frac{1}{t}$. We show that if we can find a t-sparse vector $\mathbf{x}$ like this for very small t, it can lead to attacks against sparse LWE in certain cases. The idea is that one can multiply $\mathbf{sA} + \mathbf{e}$ with $\mathbf{x}$ to yield $\mathbf{ex}$. Observe that $\mathbf{ex}$ only touches upon t coordinates of small norm error vector $\mathbf{e}$. If t is very small and the error bound B is also very small so that $B^t \ll p$ one could potentially use brute force to iterate over all the relevant choices of error to learn with high probability the error coordinates corresponding to the t non-zero locations of $\mathbf{x}$. To rule out such attacks we propose that $B^t \gg p$. In section 7.1, we show that when $k = \log n$ or above, t is linear in n as long as the sample complexity $m = \mathsf{poly}(n)$. Hence if $p \ll B^t$, this attack will either not apply or take exponential time. We believe that this attack is easy to circumvent by choosing parameters carefully as it applies less obviously to sparse LWE than sparse LPN. For typical settings we would work with polynomial sized modulus and polynomial norm errors and so t only needs be moderately big. This can be easily ensured by setting k to be bigger than some large enough constant.

Attacks Inspired by LWE We have a great body of work on LWE algorithms consisting of algebraic [AG11], combinatorial [BKW03, ACF+15, Lyu05], lattice-reduction / geometric attacks [LLL82, Sch87, SE94] and reductions for LWE [Reg05, Pei09, BLP+13]. We observe that naively these algorithms do not exploit the sparsity structure of the coefficients directly that well. It is however plausible that some algorithms fare marginally better with sparse matrices – we leave practical investigation of this as an open question for future work to explore. We observe that the natural way for LWE based attacks to exploit sparsity would be to identify what we call a *dense minor*.

Given m samples of sparse LWE of dimension n and sparsity k, one could try to identify a special set of L variables for $L < n$ so that there are lots of LWE samples supported only over those L variables. Indeed, then one might be able to meaningfully use an LWE solver on a smaller dimension L. For instance if $L = n/2$ and say there are more than $L \log p$ samples supported over those L variables, this means that one might be able to use an LWE solver that works over much smaller dimension $n/2$ giving rise to exponential improvement over general LWE of dimension n in the case of sparse LWE.

Nevertheless, in section 7.2 we show that when $k = \log n$, if the number of samples is bounded by some polynomial, with high probability there won't exist any set of up to L variables with $\geq L + 1$ samples for some $L = \Theta(n)$.

We believe that this is a very natural way of leveraging sparsity. Typical attacks that require a large number of samples generate these samples by taking small linear combinations of the given samples. Such combinations would increase the sparsity rapidly (as the locality pattern would form an expander) and therefore it is not clear how to leverage sparsity meaningfully using this approach. Despite our serious efforts of exploiting sparsity more cleverly, we could not come up with a method that does it better than finding a dense minor. Moreover, our condition is really

mild. In our *dense-minor* model, we look for minors of size L with just $L+1$ samples (one more than the dimension). Most attacks would need $L \log p$ samples or cL samples for some constant $c > 1$ for optimal running time. At $L + 1$ samples the secret may not even be information theoretically determined.

Concrete Parameters from Dense Minor We suggest that our dense-minor attack gives a nice way to capture all natural attacks that exploit sparsity. Then, relying on influential work on LWE parameter estimation [APS15], we can suggest a nice method to estimate concrete parameters. Namely, since concrete parameters are well understood thanks to highly impactful prior work [Pla18] for LWE, based on the dense-minor model we can use it as a benchmark for finding parameters for sparse LWE. For example, for our required number of samples m, and the required level of security (say 128 bits), one could use the Lattice Estimator [ACD+18] and learn dimension n, error and the modulus parameters.

We compute a new dimension n' by setting k appropriately (say around $\log n$) and then computing the expected size of its dense minor. We want to set n' and k so that the dense minor is at least of size n given m samples. We provide a script that computes the dimension n as a function of n', m and k based on the dense-minor model. By simply running this script for various values of n', we can find an appropriate dimension such that dense minors of size $< n$ are highly unlikely to exist. In section 7.4, we show concrete examples of this for improving SimplePIR [HHC+23]. We take their number of required samples, error probabilities and modulus and work out the dimension n' such that sparse LWE with dimension n' and m samples provides the same level of security as their parameter settings (according to our conjecture) with a sparsity k that is much smaller. Indeed, for a reasonable $k \leq 50$, we get very comparable dimensions n' close to n for all settings of sample complexity considered in [HHC+23]. Typical dimensions for 128 bit security in SimplePIR was suggested around $n = 1024$. This suggests around a factor of $n/k \approx 20$ improvement in the efficiency in certain aspects of the scheme (query generation, setup and update time). Due to this, we believe that the sparse LWE assumption has great promise and must be analyzed further.

Asymptotically, we conjecture when $k \approx \log n$, for polynomial number of samples m, sparse LWE with dimension $n' = \alpha \cdot n$ for some constant $\alpha > 1$ is concretely similarly hard as LWE with dimension n (assuming the sample complexity, error and modulus remain the same).

Exploiting Special Structure We note that this work does not touch upon structured assumptions for sparse LWE such as circular sparse LWE or small secret LWE. We have observed that some of these variants behave differently for sLWE as compared to LWE. We leave investigating such assumptions to future work.

2.3 Applications

The hardness of sparse LWE implies very efficient homomorphic encryption schemes for low degree computations. In particular, we obtain the first secret key Linearly Homomorphic Encryption (LHE) schemes with constant $O(1)$ or superconstant $\omega(1)$ storage and evaluation overheads, and secret key homomorphic encryption for arbitrary constant-degree polynomials with the same level of overheads. The schemes are extremely simple. For LHE, we simply consider the vanilla secret key encryption scheme where ciphertexts have form

$$\mathsf{ct}_i^\mathsf{T} = (\mathbf{a}_i^\mathsf{T}, b_i = \mathbf{sa} + m_i + e \cdot p \bmod q) ,$$

for which homomorphically evaluating a linear function specified by the coefficient vector $\mathbf{l} \in \mathbb{Z}_p^\ell$ simply amounts to computing $\mathsf{ct} = \mathsf{CT} \cdot \mathbf{l} \bmod q$. The efficiency stems from the fact that as sparse LWE samples, all vectors $\boldsymbol{a}_i$ are k sparse. Therefore, each ciphertext can be stored in a succinct form using $(k \log n + (k+1) \log q)$ bits. If the message space $\mathbb{Z}_p$ satisfies $\log p = \Omega(\log q + \log n)$, the rate of encryption is $O(k)$, which can be set to a constant if the total number of ciphertexts is a bounded polynomial and superconstant for unbounded number of ciphertexts. To analyze the evaluation overhead, we observe that the cost of one step addition $\mathsf{ct} + \mathsf{ct}_i \cdot l_i$ where ct_i is a fresh $(k+1)$-sparse ciphertext is linear in k in the Random Access Memory model: one just need to perform $(k+1)$ random accesses into ct, costing $(k+1) \log n$ time (the address length is $\log n$), followed by $(k+1)$ additions mod q. Comparing with the cost of performing one multiplication and one addition mod p in the clear, the overhead is $O(k)$ when p is sufficiently large as described above. We illustrate that our efficient LHE can lead to performance improvement in the practical Private Information Retrieval scheme SimplePIR based on LWE [HHC+23], reducing both the server database update time and the client query time by a multiplicative factor of n/k.

The same phenomenon extends to evaluating constant degree polynomials. The key to small overheads is keeping the sparsity of the ciphertexts small. We show that with some optimization over the Brakerski-Gentry-Vaikuntanathan scheme [BGV12a], for computing a degree 2^D monomial, we can bound the sparsity of the ciphertexts by $k^{O(2^D)} D^{2^D}$, which is a constant when k is a constant. Using the BGV scheme directly would lead to sparsity $k^{O(2^D)} (\log q)^{2^D}$, which is a large polylog. We avoid this by using a gadget vector $\boldsymbol{g}$ that contains powers of a large base, namely p, instead of 2. Another issue is that the BGV modulus reduction performs rounding on the ciphertexts and requires the secret to be short, in order to bound the error introduced by rounding. When using sparse LWE, we would like to rely on large secrets, since small secret sparse LWE is *not* equivalent to sLWE. Instead, we perform flipped modulus reduction – rounding the secret instead of the ciphertexts.

We describe our LHE and constant-degree HE schemes based on sparse LWE, in Section 8.1 and Section 8.2 respectively. We also give comparisons with other lattice-based schemes proposed in the literature. In short, to the best of our knowledge, no prior HE scheme achieves constant or super-constant overhead, even for the simplest linear evaluation. Their overheads usually grows at least linearly with λ, with the exception of [GHS12] that achieves fully homomorphic encryption with (large) polylogarithmic overheads. They are also limited to quasipolynomially large modulus, whereas our scheme can handle any message space $\mathbb{Z}_p$ that is not too small $\log p = \Omega(\log n)$.

3 Preliminaries

Let $\mathbb{N} = \{1, 2, \dots\}$ be the natural numbers, and define $[a, b] := \{a, a+1, \dots, b\}$, $[n] := [1, n]$. Our logarithms are in base 2. For a finite set S, we write $x \leftarrow S$ to denote uniformly sampling x from S. We denote the security parameter by λ; our parameters depend on λ, e.g. $n = n(\lambda)$, and we often drop the explicit dependence. We abbreviate PPT for probabilistic polynomial-time. Our adversaries are non-uniform PPT ensembles $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$. We write $\mathsf{negl}(\lambda)$ to denote negligible functions in λ. Two ensembles of distributions $\{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{D}'_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if for any non-uniform PPT adversary $\mathcal{A}$ there exists a negligible function negl such that $\mathcal{A}$ can distinguish between the two distributions with probability at most $\mathsf{negl}(\lambda)$. For any binary outcome adversary $\mathcal{A}$ we define its distinguishing advantage in distinguishing $\{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{D}'_\lambda\}_{\lambda \in \mathbb{N}}$ as:

$$\left| \Pr_{x \leftarrow \mathcal{D}_\lambda} [\mathcal{A}_\lambda(1^\lambda, x) = 1] - \Pr_{x \leftarrow \mathcal{D}'_\lambda} [\mathcal{A}_\lambda(1^\lambda, x) = 1] \right|$$

We will use matrices and vectors throughout. Both of them will be represented in boldface letters. Matrices are represented using capital bold letters (such as $\mathbf{A}, \mathbf{B}$) and vectors using lowercase bold letters (such as $\mathbf{a}, \mathbf{b}$). To refer to i^{th} index of any vector $\mathbf{v}$, we denote it by $\mathbf{v}[i]$. For matrices, we denote that by $\mathbf{A}[i, j]$.

3.1 Lattice Preliminaries

A *lattice* $\mathcal{L}$ is a discrete subgroup of $\mathbb{R}^m$, or equivalently the set

$$\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \left\{ \sum_{i=1}^{n} x_i \boldsymbol{b}_i \ : \ x_i \in \mathbb{Z} \right\}$$

of all integer combinations of n linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$. Such $\boldsymbol{b}_i$'s form a *basis* of $\mathcal{L}$.

The lattice $\mathcal{L}$ is said to be *full-rank* if $n = m$. We denote by $\lambda_1(\mathcal{L})$ the first minimum of $\mathcal{L}$, defined as the length of a shortest non-zero vector of $\mathcal{L}$.

Definition 3.1 (Successive Minima)**.** *Given a lattice $\mathcal{L}$ and an integer $1 \leq k \leq n$, define the k-th successive minimum $\lambda_k(\mathcal{L})$ as the smallest possible length of a set of k linearly independent vectors in $\mathcal{L}$.*

Equivalently, $\lambda_k(\mathcal{L})$ is the smallest value r such that a ball of radius r (centered on the origin) contains k linearly independent lattice points.

For a rank n lattice $\mathcal{L} \subset \mathbb{R}^n$, the *dual lattice*, denoted $\mathcal{L}^*$, is defined as the set of all points in $\mathsf{span}(\mathcal{L})$ that have integer inner products with all lattice points,

$$\mathcal{L}^* = \{\boldsymbol{w} \in \mathsf{span}(\mathcal{L}) : \forall \boldsymbol{y} \in \mathcal{L}, \langle \boldsymbol{w}, \boldsymbol{y} \rangle \in \mathbb{Z}\} \ .$$

Similarly, for a lattice basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, we define the dual basis $\boldsymbol{B}^* = (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*)$ to be the unique set of vectors in $\mathsf{span}(\mathcal{L})$ satisfying $\langle \boldsymbol{b}_i^*, \boldsymbol{b}_j \rangle = 1$ if $i = j$, and 0, otherwise. It is easy to show that $\mathcal{L}^*$ is itself a rank n lattice and $\boldsymbol{B}^*$ is a basis of $\mathcal{L}^*$. Given a lattice $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, we denote $\|\boldsymbol{B}\|_2 = \max_i \|\boldsymbol{b}_i\|$.

An important property of the dual lattice is the following transference theorem, shown in Theorem 2.1 of [Ban93].

Lemma 3.1 (Transference Theorem)**.** *For any n-rank lattice $\mathcal{L}$, $1 \leq \lambda_1(\mathcal{L}) \cdot \lambda_n(\mathcal{L}^*) \leq n$.*

Discrete Gaussian and Related Distributions For any $s > 0$, define

$$\rho_s(\boldsymbol{x}) = \exp(-\pi \|\boldsymbol{x}\|^2 / s^2)$$

for all $\boldsymbol{x} \in \mathbb{R}^n$. We write ρ for ρ_1. For a discrete set S, we extend ρ to sets by $\rho_s(S) = \sum_{\boldsymbol{x} \in S} \rho_s(\boldsymbol{x})$. Given a lattice $\mathcal{L}$, the *discrete Gaussian* $\mathcal{D}_{\mathcal{L},s}$ is the distribution over $\mathcal{L}$ such that the probability of a vector $\boldsymbol{y} \in \mathcal{L}$ is proportional to $\rho_s(\boldsymbol{y})$:

$$\Pr_{X \leftarrow \mathcal{D}_{\mathcal{L},s}}[X = \boldsymbol{y}] = \frac{\rho_s(\boldsymbol{y})}{\rho_s(\mathcal{L})}.$$

Using standard sub-Gaussian tail-bounds, one can show we can show the following claim.

Claim 3.1. *Let $m \in \mathbb{N}$, $\sigma > 0$, then it holds that:*

$$\Pr_{\boldsymbol{e} \leftarrow \mathcal{D}_{\mathbb{Z}^m,\sigma}}[\|\boldsymbol{e}\| > m\sigma] < \exp(-\widetilde{\Omega}(m)).$$

We define the truncated discrete Gaussian as a distribution statistically close to discrete Gaussian where one samples the vectors from discrete Gaussian conditioned on their norm being upper bounded by $m\sigma$.

Trapdoor Sampling We will need the following definition of a lattice trapdoor [GPV08b, MP13, Vai20]. For $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$, we define the rank m lattice

$$\mathcal{L}^{\perp}(\mathbf{A}) = \{\boldsymbol{z} \in \mathbb{Z}^m \ : \ \boldsymbol{A}\boldsymbol{z} = \mathbf{0} \pmod q\} \ .$$

A lattice trapdoor for $\boldsymbol{A}$ is a set of short linearly independent vectors in $\mathcal{L}^{\perp}(\boldsymbol{A})$.

Definition 3.2. *A matrix $\boldsymbol{T} \in \mathbb{Z}^{m \times m}$ is a β-good lattice trapdoor for a matrix $\boldsymbol{A} \in \mathbb{Z}_p^{n \times m}$ if*

1. $\boldsymbol{A}\boldsymbol{T} = \mathbf{0} \pmod p$.
2. *For each column vector $\boldsymbol{t}_i$ of $\boldsymbol{T}$, $\|\boldsymbol{t}_i\|_\infty \leq \beta$.*
3. *$\boldsymbol{T}$ has rank m over $\mathbb{R}$.*

Theorem 3.1. *[GPV08b, MP13, Vai20] There is an efficient algorithm that, on input $1^n, p, m \geq 10n \log p$, outputs a matrix $\boldsymbol{A}$ distributed statistically close to uniformly on $\mathbb{Z}_p^{n \times m}$, and a $O(m)$-good lattice trapdoor $\boldsymbol{T}$ for $\boldsymbol{A}$.*

We also observe that due to the transference theorems for any matrix $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$ a lower bound on $\lambda_1(\mathcal{L})$ implies an upper bound on $\lambda_m(\mathcal{L}^{\perp})$.

Corollary 3.1. *(of Transference Theorem 3.1) Given any $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$, if $\lambda_1(\mathcal{L}) \geq \beta$, then $\lambda_m(\mathcal{L}^{\perp}) \leq m \cdot \frac{p}{\beta}$. Here $\mathcal{L} = \mathcal{L}(\mathbf{A})$ and $\mathcal{L}^{\perp} = \mathcal{L}^{\perp}(\mathbf{A})$.*

Proof. Recall that the transference theorem states:

$$1 \leq \lambda_1(\mathcal{L}) \cdot \lambda_m(\mathcal{L}^*) \leq m,$$

where $\mathcal{L}^*$ is the dual of $\mathcal{L}$. Observe that $\mathcal{L}^{\perp} = p\mathcal{L}^*$ and so $\lambda_m(\mathcal{L}^*) = \frac{\lambda_m(\mathcal{L}^{\perp})}{p}$. The claim follows from using the right inequality using the lower bound on $\lambda_1(\mathcal{L}) \geq \beta$.

□

4 Sparse Learning with Error

We now formally define our sparse LWE assumption. Our assumption is exactly identical to LWE with a crucial difference that each coefficient vector is exactly k-sparse for some k that for the most part should be thought of as a small function of the dimension n (ideally a constant or polylogarithmic in n).

Definition 4.1 (Coefficient Distribution $\mathcal{D}_{\mathsf{coeff},n,k,p}$)**.** *For a parameter $n \in \mathbb{N}$, and a $k = k(n)$, a modulus p, we define the distribution $\mathcal{D}_{\mathsf{coeff},n,k,p}$ that first samples a uniformly random set or a "locality pattern" $S \subset [n]$ so that S has exactly size k (we abuse notation to denote this by saying $S \leftarrow \binom{n}{k}$). Then, the distribution samples a random vector as follows. Sample $\mathbf{a} \in \mathbb{Z}_p^n$ so that each coordinate $\mathbf{a}[i]$ is set to 0 if $i \notin S$. Otherwise $\mathbf{a}[i]$ is chosen at random from $\mathbb{Z}_p$.*

Definition 4.2 (Locality Set)**.** *For a modulus p, and any vector $\mathbf{v} \in \mathbb{Z}_p^n$ we denote by the locality set of $\mathbf{v}$ (denoted as $\mathsf{LocSet}(\mathbf{v})$) as a set of indices $S \subset [n]$ so that $\mathbf{v}[i]$ is non-zero for every $i \in S$.*

Definition 4.3 (Sparse LWE Distinguishing Problem)**.** *For an integer dimension n, sparsity parameter k, a sample complexity m, modulus p, and noise parameter $\sigma = \sigma(n)$ we define the $\mathsf{sLWE}_{n,k,m,p,\sigma}$ problem as a distinguishing problem. In the first distribution, we generate m sparse LWE samples as follows:*

- *Sample m coefficient vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m$ where each $\mathbf{a}_i \leftarrow \mathcal{D}_{\mathsf{coeff},n,k,p}$.*
- *Sample a secret vector $\mathbf{s} \leftarrow \mathbb{Z}_p^n$.*
- *Sample m noise values from $\{e_i \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}\}_{i\in[m]}$.*
- *The first distribution consists of $\{(\mathbf{a}_i, b_i = \langle \mathbf{s}, \mathbf{a}_i\rangle + e_i \mod p)\}$*

The second distribution is exactly the same except $\{b_i\}_{i\in[m]}$ are randomly sampled from $\mathbb{Z}_p$.

We also define the LWE distinguishing problem to be identical to a sparse LWE distinguishing problem as above except that the vectors $\mathbf{a}_i$ are chosen randomly from $\mathbb{Z}_p^n$ as opposed to being sampled as sparse vectors.

Definition 4.4 (Sparse LWE Assumption)**.** *Let $k = k(n) \in [0, n]$, $p = p(n) : \mathbb{N} \to \mathbb{N}$, $\sigma = \sigma(n) : \mathbb{N} \to \mathbb{N}$, $m = m(n) : \mathbb{N} \to \mathbb{N}$ be efficiently computable functions (computable in time polynomial in n).*

We say that the sparse LWE assumption with respect to sparsity $k(n)$, modulus $p(n)$, noise parameter $\sigma = \sigma(n)$ and sample complexity $m(n)$ holds, denoted as the $\mathsf{sLWE}_{n,k,m,p,\sigma}$ assumption, if for every non-uniform p.p.t adversary $\mathcal{A}$, the distinguishing advantage in winning the $\mathsf{sLWE}_{n,k,m,p,\sigma}$ distinguishing problem (Definition 4.3) is bounded by some negligible function $\mathsf{negl}(n)$.

We say that the sparse LWE assumption with respect to sparsity $k(n)$, modulus $p(n)$, noise parameter $\sigma = \sigma(n)$ and unbounded polynomial number of samples holds, denoted as the $\mathsf{sLWE}_{n,k,p,\sigma}$ assumption, if the above condition holds for every non-uniform p.p.t adversary $\mathcal{A}$, and every polynomial $m(n)$.

We say that the subexponential $\mathsf{sLWE}_{n,k,m,p,\sigma}$ (respectively $\mathsf{sLWE}_{n,k,p,\sigma}$) assumption holds if there exists a constant $\epsilon \in (0, 1)$, such that, every non-uniform probabilistic $\mathsf{poly}(2^{n^\epsilon})$-time adversary $\mathcal{A}$ (and respectively every polynomial $m(n)$), the distinguishing advantage in winning the $\mathsf{sLWE}_{n,k,m,p,\sigma}$ distinguishing problem is bounded by $1/\mathsf{poly}(2^{n^\epsilon})$.

Remark 4.1. For notational brevity, we will also denote sparse LWE samples as $(\mathbf{A}, \mathbf{sA}+\mathbf{e} \mod p)$ where the coefficient vectors $\{\mathbf{a}_i\}$ form the columns of the matrix $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$. $\mathbf{s}$ is interpreted to live in $\mathbb{Z}_p^{1\times m}$ and the error vector $\mathbf{e}$ is in $\mathbb{Z}^{1\times m}$.

4.1 Minimum Distance of the Sparse LWE Lattice

We now analyze bounds on $\lambda_1(\mathcal{L})$ for the lattice:

$$\mathcal{L}(\mathbf{A}) = \{\mathbf{xA} + p\mathbb{Z}^m : \mathbf{x} \in \mathbb{Z}_p^{1\times n}\}.$$

We compute these bounds as a function of dimension n, sparsity parameter $k = k(n)$, sample complexity $m = m(n)$ and the modulus $p = p(n)$. These bounds will be useful for identifying when short lattice trapdoors for $\mathbf{A}$ might exist. We will also use this calculation for understanding the running time of various lattice-reduction based attacks on sparse LWE.

We start by recalling the minimum distance in the case when matrix $\mathbf{A}$ is randomly chosen from $\mathbb{Z}_p^n$ (in other words, it corresponds to plain LWE). In this case, it can be shown that as n grows, the minimum distance $\lambda_1(\mathcal{L})$ is roughly around $p^{1-n/m}$ for $m \gg n$ (up to polynomial factors, which we ignore). When $m = \Omega(n \log p)$, this is roughly p.

For sparse LWE, since the matrix has a lot of zeros, one would expect the minimum distance to be smaller. However, we show that for interesting parameter regimes (i.e. with $k > \log n$, and p being polynomial in the size n, and for polynomial m) the minimum distance with high probability for the case of sparse LWE is also around p. We give a lower-bound on the minimum distance.

Our main result is the following:

Theorem 4.1. *Let $\mathbf{A}$ be a $n \times m$ sparse matrix each of whose columns are sampled i.i.d. from $\mathcal{D}_{\mathsf{coeff},n,k,p}$. With probability $1-o(1)$, the length of the shortest nonzero vector in $\mathcal{L}(\mathbf{A})$ is $\Theta(p)$ assuming $m \geq 8n \log n$ and $p = \mathsf{poly}(n)$.*

We do this calculation in two steps. First we prove that if the sparse matrix $\mathbf{A}$ satisfies a combinatorial property regarding the positions of non-zero coordinates, then with high probability over the choice of those non-zero coordinates sampled randomly from $\mathbb{Z}_p$, the minimum distance is large.

We start by defining the combinatorial property below.

Definition 4.5. *We say that a matrix $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$ is c-row expanding if the following holds. Let the rows of $\mathbf{A}$ be $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{Z}_p^m$. We define $S_1, \ldots, S_n$ to be the sets of indices $\subset [m]$ that correspond to non-zero entries in $\mathbf{v}_1, \ldots, \mathbf{v}_n$ respectively. Then, we require that for every non-empty $T \subseteq [n]$ of size t,*

$$\left| \bigcup_{i \in T} S_i \right| > c \cdot t$$

Since each row is supported roughly at $\frac{km}{n}$ columns, we would expect c to be roughly $\frac{km}{n}$ for small t. For t closer to n, this should behave more like $\frac{m}{n}$. Nevertheless, we show that minimum distance calculation reduces to understanding this combinatorial property.

Theorem 4.2. *Let $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$ be chosen so that:*

- *It is c-row expanding for some $c > 1$.*
- *Conditioned on that, the non-zero entries are randomly chosen from $\mathbb{Z}_p$.*

Then, with probability $1 - o(1)$, the size of the smallest vector in the lattice is $\Omega\left(\frac{p^{1-\frac{1}{c}}}{n^{1/c}}\right)$

Proof. We will prove a slightly stronger result and show that the stated lower bound actually holds for $\|\lambda_1(\mathcal{L}(\mathbf{A}))\|_\infty$. The theorem statement then follows since the 2-norm is always at least as much as the infinity norm.

Fix a coefficient vector $\mathbf{x} \in \mathbb{Z}_p^n$. If we consider the lattice $\{\mathbf{xA} + p\mathbb{Z}^m\}$, it is easy to see that its shortest vector can be obtained by reducing each coordinate of $\mathbf{xA}$ modulo p so that they lie in $[-p/2, p/2]$. We start by calculating the probability that the infinity norm of this vector is less than B for an arbitrary $B < p/2$, where the probability is taken over the randomness of $\mathbf{A}$.

$$
\begin{aligned}
&\Pr_{\mathbf{A}}\left[\|\lambda_1(\{\mathbf{x}\mathbf{A}+p\mathbb{Z}^m\})\|_\infty < B\right] \\
=&\Pr_{\mathbf{A}}\left[\bigwedge_{j\in[m]}\left(\sum_{i\in[n]}\mathbf{x}[i]\mathbf{A}[i][j] \mod p < B\right)\right] \\
=&\Pr_{\mathbf{A}}\left[\bigwedge_{j\in[m]}\left(\sum_{i\in T}\mathbf{x}[i]\mathbf{A}[i][j] \mod p < B\right)\right] && T\subseteq[n] \text{ is the set of indices where } \mathbf{x}[i]\neq 0 \\
=&\Pr_{\mathbf{A}}\left[\bigwedge_{j\in N(T)}\left(\sum_{i\in T}\mathbf{x}[i]\mathbf{A}[i][j] \mod p < B\right)\right] && N(T)\subseteq[m] \text{ is the set of columns of } \mathbf{A} \text{ spanned by } T \\
=&\prod_{j\in N(T)}\left(\Pr_{\mathbf{A}}\left[\sum_{i\in T}\mathbf{x}[i]\mathbf{A}[i][j] \mod p < B\right]\right) \\
=&\left(\frac{B}{p/2}\right)^{|N(T)|} \\
\leq&\left(\frac{B}{p/2}\right)^{c|T|} && \text{since } \mathbf{A} \text{ is } c\text{-row expanding}
\end{aligned}
$$

Let us denote $|T|$ by t. If we fix t (the number of nonzero entries of $\mathbf{x}$), we have $\binom{n}{t}$ choices for T and p^t choices of $\mathbf{x}$ for each T. Therefore, taking an union bound over all $1\leq t\leq n$, we obtain

$$
\begin{aligned}
&\Pr_{\mathbf{A}}\left[\|\lambda_1(\{\mathbf{x}\mathbf{A}+p\mathbb{Z}^m : \mathbf{x}\in\mathbb{Z}_p^n\})\|_\infty < B\right] \\
=&\Pr_{\mathbf{A}}\left[\bigvee_{\mathbf{x}}\left(\|\lambda_1(\{\mathbf{x}\mathbf{A}+p\mathbb{Z}^m\})\|_\infty < B\right)\right] \\
\leq&\sum_{\mathbf{x}}\Pr_{\mathbf{A}}\left[\|\lambda_1(\{\mathbf{x}\mathbf{A}+p\mathbb{Z}^m\})\|_\infty < B\right] && \text{By the union bound} \\
\leq&\sum_{\mathbf{x}}\left(\frac{B}{p/2}\right)^{c|T|} && T\subseteq[n] \text{ is the set of indices where } \mathbf{x}[i]\neq 0 \\
=&\sum_{t=1}^{n}\binom{n}{t}p^t\left(\frac{2B}{p}\right)^{ct} \\
=&\left(\frac{(2B)^c}{p^{c-1}}+1\right)^n - 1 && \text{By the binomial theorem}
\end{aligned}
$$

The above expression is asymptotically subconstant as long as

$$
\frac{(2B)^c}{p^{c-1}} = o\left(\frac{1}{n}\right) \Leftrightarrow B = o\left(\frac{p^{1-1/c}}{n^{1/c}}\right)
$$

Therefore, we can conclude that with probability $1-o(1)$, the shortest nonzero vector of the lattice $\mathcal{L}(\mathbf{A})$ is $\Omega\left(\frac{p^{1-1/c}}{n^{1/c}}\right)$. □

We then show that random matrices sampled from the coefficient distribution are $\log n$-row expanding with high probability.

Theorem 4.3. *Let $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$ be a matrix whose columns are sampled independently from the coefficient distribution $\mathcal{D}_{\mathsf{coeff},n,k,p}$. If the number of samples, sparsity parameter and the dimension satisfy $m > 4(1+1/k)n \log n$, $\mathbf{A}$ is $\log n$-row expanding with probability $1 - o(1)$.*

Proof. We will bound the probability that $\mathbf{A}$ is *not* $\log n$-row expanding. In that case, there must be some $T \subseteq [n]$ with $|T| = t$ such that the rows of $\mathbf{A}$ indexed by T span at most $t \log n$ columns. Let us calculate the probability of this event for some fixed T.

For each column i, we define the indicator random variable C_i which is 1 if it is spanned by T and 0 otherwise. Since the locality patterns of each column are chosen independently, these are i.i.d. Bernoulli random variables with expectation $1 - \left(\binom{n-t}{k} / \binom{n}{k}\right)$. We are interested in the probability of their sum not exceeding $t \log n$. We can get a fairly tight bound by applying the Chernoff bound.

First, we will show that the expected number of columns spanned by T is more than $4t \log n$.

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{m} C_i\right] &= m\,\mathbb{E}\left[C_i\right] \\
&= m\left(1 - \binom{n-t}{k} \Big/ \binom{n}{k}\right) \\
&= m\left(1 - \frac{(n-t)(n-t-1)\cdots(n-t-k+1)}{n(n-1)\cdots(n-k+1)}\right) \\
&\geq m\left(1 - \left(\frac{n-t}{n}\right)^k\right) \\
&= m\left(1 - \left(1 - \frac{t}{n}\right)^k\right) \\
&\geq m\left(1 - \frac{1}{1+tk/n}\right) \\
&= \frac{mtk}{n+tk} \geq t \cdot \frac{mk}{(k+1)n} \\
&= t \cdot \frac{m}{(1+1/k)n} \geq 4t \log n
\end{aligned}$$

We can now use Chernoff bound to bound the probability that the above quantity is less than a fourth of its expectation.

$$
\begin{aligned}
&\Pr\left[T \text{ spans at most } t\log n \text{ columns}\right] \\
&= \Pr\left[\left(\sum_{i=1}^{m} C_i\right) \leq t\log n\right] \\
&\leq \Pr\left[\left(\sum_{i=1}^{m} C_i\right) \leq \left(1-\frac{3}{4}\right)\mathbb{E}\left[\sum_{i=1}^{m} C_i\right]\right] \\
&\leq \exp\left(-\frac{9m}{32}\left(1-\binom{n-t}{k}\Big/\binom{n}{k}\right)\right) \\
&\leq \exp\left(-\frac{9}{32}\cdot m\left(1-\left(1-\frac{t}{n}\right)^{k}\right)\right) \\
&\leq \exp\left(-\frac{9}{32}\cdot(4t\log n)\right) \\
&\leq n^{-9t/8}
\end{aligned}
$$

Observe that we can choose T in $\binom{n}{t}$ ways. Therefore, the union bound implies that

$$
\begin{aligned}
&\Pr\left[\mathbf{A} \text{ is not } \log n\text{-row expanding}\right] \\
&\leq \sum_{t=1}^{n}\binom{n}{t}n^{-9t/8} \\
&= \left(1+\frac{1}{n\sqrt[8]{n}}\right)^{n}-1 \qquad \text{by the binomial theorem}
\end{aligned}
$$

The above expression is asymptotically subconstant since

$$
\frac{1}{n\sqrt[8]{n}} = o\left(\frac{1}{n}\right)
$$

This concludes our proof that the probability of $\mathbf{A}$ being $\log n$-row expanding is $1-o(1)$. □

We now have all we need to prove our main result.

Proof of theorem 4.1. We just put together the above two theorems. Since $(1+1/k)$ can't exceed 2, setting $m > 8n\log n$ satisfies the condition of theorem 4.3. Therefore, under the parameter regime of theorem 4.1, $\mathbf{A}$ is $\log n$-row expanding with probability $1-o(1)$. Conditioning on this row expansion being true, theorem 4.2 implies that

$$
\lambda_1(\mathcal{L}(A)) = \Omega\left(\frac{p^{1-\frac{1}{\log n}}}{n^{\frac{1}{\log n}}}\right) = \Omega\left(p^{1-\frac{1}{\log n}}\right)
$$

which is only a constant factor away from p since $p = \mathsf{poly}(n)$. □

5 Reduction from LWE to Sparse LWE

5.1 Overview

We now show that given an oracle that breaks sparse LWE of dimension n and sparsity k with m samples, we can break LWE of dimension k with m samples as long as the modulus p is a prime.

We get as input $\mathbf{A}, \mathbf{b} = \mathbf{sA} + \mathbf{e}$, and we will effectively convert it to a tuple of the form $\mathbf{A}', \mathbf{b}' = \mathbf{s}'\mathbf{A}' + \mathbf{e}$ with the same error (if our input $\mathbf{b}$ is sampled randomly instead of from the LWE distribution, our reduction will keep it randomly distributed). The distribution for $(\mathbf{A}', \mathbf{b}')$ will be identical to that of average-case sparse LWE. The nice thing about our reduction is that it does not manipulate the error $\mathbf{e}$ and therefore also applies to the setting of learning parity with noise.

We now describe the conversion procedure. Our rough idea is that given a $k \times m$ dense matrix $\mathbf{A}$ we transform each column into a sample in a related matrix (that we shortly show how to build) $\mathbf{A}'$ where $\mathbf{A}'$ is a $n \times m$ sparse matrix with k nonzero entries per column. So far we keep the $\mathbf{b}$ vector unchanged; it can be written as $\mathbf{b} = \mathbf{zA}' + \mathbf{e}$. At this point, the secret $\mathbf{z}$ is not uniformly random. But we can randomize it by adding $\mathbf{vA}'$ to $\mathbf{b}$ for a randomly chosen $\mathbf{v}$. The new secret $\mathbf{v} + \mathbf{z}$ is random and independent of $\mathbf{A}'$.

We now describe the transformation from $\mathbf{A}$ to $\mathbf{A}'$. We want to ensure that the equation $\mathbf{CA}' = \mathbf{A}$ holds for some $k \times n$ matrix $\mathbf{C}$. This would allow us to write $\mathbf{b} = \mathbf{sA} + \mathbf{e} = (\mathbf{sC})\mathbf{A}' + \mathbf{e}$. Now if we set $\mathbf{A}'$ randomly and try to find $\mathbf{C}$, this might be hard. Therefore, we do the reverse. We first fix $\mathbf{C}$ and then sample only the locality pattern $S_1, \ldots, S_m$ at random from $\binom{n}{k}$. Then, we choose the non-zero coordinates given by S_i so that $\mathbf{CA}'[i]$ is equal to $\mathbf{A}[i]$. This can be done since we can view the above matrix equation as a system of linear equations over $\mathbb{Z}_p$ with k variables corresponding to the nonzero entries of $\mathbf{A}'[i]$ and k equations corresponding to the k rows of $\mathbf{A}[i]$. We will choose $\mathbf{C}$ to be a $k \times n$ Vandermonde matrix which has the property that any k columns are linearly independent. This allows us to guarantee the system of equations described above correspond to an invertible matrix, and is hence solvable.

We have to show that the above process yields a random $\mathbf{A}'$. Observe that each column $\mathbf{A}'[i]$ can be described by its locality pattern, and the sequence of nonzero entries in the indices corresponding to the locality pattern. $\mathbf{A}_i$ is random means the nonzero entries of $\mathbf{A}'_i$ must be random. We chose the locality pattern independently at random to begin with. Therefore, the whole column $\mathbf{A}'[i]$ is distributed uniformly in $\mathcal{D}_{\mathsf{coeff},n,k,p}$ and is independent of the other columns.

5.2 Formal Description

Theorem 5.1. *Assume there exists a PPT algorithm which solves average-case* $\mathsf{sLWE}_{n,k,m,p,\sigma}$ *instances in time T with success probability γ, where the modulus p is a prime and the number of samples $m < p$. Then we can solve average-case* $\mathsf{LWE}_{k,m,p,\sigma}$ *instances in time $T + O(n^3 m \log p)$ with success probability γ.*

Our reduction does not preserve dimension. If we choose k to be small (constant or logarithmic in n), we end up with an algorithm for solving very low dimensional LWE which is anyway easy. On the other hand, if k is very large (say $\Theta(n)$), we conclude that sLWE is very hard if the columns are not too sparse (only a constant fraction of each column vector is required to be zero). The intermediate range, where k is polylogarithmic or polynomial in n, yields more interesting conclusions.

Algorithm $\mathcal{B}$

Input:

- A PPT algorithm $\mathcal{A}$ for solving $\mathsf{sLWE}_{n,k,m,p,\sigma}$
- A matrix $\mathbf{A} \leftarrow \mathbb{Z}_p^{k \times m}$
- A vector $\mathbf{b}$ which is either sampled uniformly from $\mathbb{Z}_p^{1 \times m}$, or by evaluating $\mathbf{b} = \mathbf{sA} + \mathbf{e}$ where $\mathbf{s} \leftarrow \mathbb{Z}_p^{1 \times k}$, $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^{1 \times m}$

Objective: Determine which of the above distributions $\mathbf{b}$ is sampled from.
Algorithm:

1. Let $\mathbf{C} \leftarrow \mathbb{Z}_p^{k \times n}$ be the Vandermonde matrix whose entries are defined as
$$\mathbf{C}[i][j] := j^i \mod p$$
2. Randomly select the locality pattern for a matrix with n rows and m columns. Formally, we randomly sample m subsets $\{S_i\}_{i \in [m]}$ of $[n]$ each of size k. The elements of S_i correspond to the row indices which may contain a nonzero element in column i.
3. Initialize a matrix $\mathbf{A}'$ of size $n \times m$ to all zeros.
4. For $i := 1, 2, \ldots, m$:
   4.1. Define $\mathbf{D}$ to be the square matrix of size k obtained by concatenating the columns of $\mathbf{C}$ indexed by S_i. Formally, the entry in row u, column v of $\mathbf{D}$ is
   $$\mathbf{D}[u][v] := \mathbf{C}[u][S_i[v]] = (S_i[v])^u \mod p$$
   4.2. Let $\mathbf{A}[i]$ be the i^{th} column of $\mathbf{A}$. Define $\mathbf{r} := \mathbf{D}^{-1}\mathbf{A}[i]$. Fill in the nonzero entries of column i of $\mathbf{A}'$ with the entries of $\mathbf{r}$. Formally, set
   $$\mathbf{A}'[S_i[j]][i] := \mathbf{r}[j] \qquad \forall 1 \leq j \leq k$$
5. Sample $\mathbf{v} \leftarrow \mathbb{Z}_p^{1 \times n}$
6. Return $\mathcal{A}(\mathbf{A}', \mathbf{vA}' + \mathbf{b})$

We first prove a few properties of $\mathcal{B}$.

Lemma 5.1. $\mathbf{CA}' = \mathbf{A}$

Proof. If we denote the i^{th} columns of $\mathbf{A}$ and $\mathbf{A}'$ by $\mathbf{t}$ and $\mathbf{t}'$ respectively, we just need to show that $\mathbf{Ct}' = \mathbf{t}$. Since $\mathbf{t}'$ is mostly full of zeros except the indices in S_i, we can express $\mathbf{Ct}'$ as $\mathbf{Dr}$ where

$$\mathbf{r}[j] := \mathbf{A}'[S_i[j]][i] \qquad \& \qquad \mathbf{D}[u][v] := \mathbf{C}[u][S_i[v]]$$

The desired equality now follows directly from line 4.2. □

Lemma 5.2. *The matrix* $\mathbf{D}$ *defined in line 4.1 is invertible.*

Proof. Observe that regardless of what S_i is, $\mathbf{D}$ always satisfies $\mathbf{D}[i][j] = x_j^i$ for some distinct values x_j. This is a square Vandermonde matrix; its determinant is given by $\prod_{i<j}(x_j - x_i)$. As long as each column uses a different x_j, this determinant is nonzero and hence $\mathbf{D}^{-1}$ exists. By construction of $\mathbf{C}$, each column uses a different x since the number of columns m is less than p. Hence, regardless of which columns of $\mathbf{C}$ are chosen to construct $\mathbf{D}$, we always have $x_i \neq x_j$ for any $i \neq j$. □

Lemma 5.3. *The columns of $\mathbf{A}'$ are independent and follow the coefficient distribution $\mathcal{D}_{\mathsf{coeff},n,k,p}$.*

Proof. The locality pattern of $\mathbf{A}'$ is chosen randomly, and the nonzero values of each column are obtained by the formula $\mathbf{D}^{-1}\mathbf{t}$ where $\mathbf{t} \leftarrow \mathbb{Z}_p^k$. Even though the $\mathbf{D}$ used for different columns are not independent, the $\mathbf{t}$s are all i.i.d and hence so are the products $\mathbf{D}^{-1}\mathbf{t}$. □

Proof of theorem 5.1. We will consider the two possible ways of sampling $\mathbf{b}$ separately. Let us first handle the case where $\mathbf{b} \leftarrow \mathbb{Z}_p^{1\times m}$. Since $\mathbf{A}$ and $\mathbf{b}$ are independent, so are $\mathbf{A}'$ and $\mathbf{b}$. Since $\mathbf{b}$ is uniformly random and independent of $\mathbf{v}\mathbf{A}'$, so is the sum $\mathbf{b} + \mathbf{v}\mathbf{A}'$. Therefore the two inputs to $\mathcal{A}$ are independent and both follow the right distribution for $\mathsf{sLWE}_{n,k,m,p,\sigma}$ where the target vector is random. Hence in this case $\mathcal{A}$ succeeds with probability γ.

Now consider the other sampling procedure for $\mathbf{b}$.

$$\mathbf{C}\mathbf{A}' = \mathbf{A} \implies \mathbf{b} = \mathbf{s}\mathbf{A} + \mathbf{e} = \mathbf{s}\mathbf{C}\mathbf{A}' + \mathbf{e} \implies \mathbf{v}\mathbf{A}' + \mathbf{b} = \mathbf{s}'\mathbf{A}' + \mathbf{e}$$

where

$$\mathbf{s}' = \mathbf{s}\mathbf{C} + \mathbf{v}$$

We will now argue that the input to $\mathcal{A}$, $(\mathbf{A}', \mathbf{v}\mathbf{A}' + \mathbf{b})$ has exactly the same distribution as expected by average-case sparse LWE oracle. A sparse LWE instance where we follow the second sampling procedure for the target vector can be characterized by the matrix, the secret vector and the error vector. The error remains unchanged in the reduction. The new secret is uniformly random and independent of $\mathbf{A}'$ since $\mathbf{v}$ is chosen independently uniformly at random. The matrix is sampled from $\mathcal{D}_{\mathsf{coeff},n,k,p}^m$. Hence $\mathcal{A}$ succeeds with probability γ in this case as well.

Therefore, the failure probability of $\mathcal{B}$ is exactly the same as that of $\mathcal{A}$. □

Remark 5.1. We needed p to be a prime in order for $\mathbb{Z}_p$ to be a field. This allowed us to invert matrices.

Remark 5.2. We need m to be large enough in order for $\mathcal{A}$ to have any chance of solving sLWE. This is, however, not a problem. As long as we start with $m > k \log p$, it is always possible to generate more LWE samples by taking small linear combinations. We use this procedure to modify the input and increase m, if necessary, before we begin to run $\mathcal{B}$.

Remark 5.3. Analogously to sLWE, it is possible to define a sparse variant of the LPN problem [ADI+17b]. Our reduction works exactly as is in that setting, since we don't depend on or change the error distribution in any way. Thus a PPT algorithm for $\mathsf{sLPN}_{n,k,m,p,\sigma}$ implies a PPT algorithm for $\mathsf{LPN}_{k,m,p,\sigma}$.

6 Trapdoor Sampling for Sparse LWE

6.1 Overview

We are going to present an algorithm for sampling pairs of matrices $\mathbf{A}, \mathbf{T}$ where $\mathbf{A}$ is a $n \times m$ sparse matrix and $\mathbf{T}$ is a trapdoor for $\mathbf{A}$. The distribution of each column of $\mathbf{A}$ should be statistically close to independent samples from $\mathcal{D}_{\mathsf{coeff},n,k,p}$.

Our approach stands on the shoulders of amazing prior work of [MP12]. Recall that they showed how to sample a matrix $\mathbf{V} \in \mathbb{Z}_p^{n \times m}$ for $m = O(n \log p)$ that is statistically close to random along with a full-rank square matrix $\mathbf{T_V}$ of norm $O(m)$ in the nullspace of $\mathbf{V}$ ($\mathbf{T_V}$ can be converted to a basis generically as shown by lemma 7.1 of [MG02] to a basis for $\mathcal{L}^{\perp}(\mathbf{V})$). The idea is that first we sample a matrix $\mathbf{A} \leftarrow \mathbb{Z}_p^{n \times m'}$ for $m' > 2n \log p$ at random. Then we sample a random binary matrix $\mathbf{R} \leftarrow \{0,1\}^{m' \times n\lceil \log p \rceil}$, and consider $\mathbf{V} = [\mathbf{A} | \mathbf{AR} + \mathbf{G}]$ where $\mathbf{G}$ is the gadget matrix defined as $\mathbf{I}_n \otimes \mathbf{g}$ where $\mathbf{g} = [1\, 2\, 4\, \ldots\, 2^{\lceil \log p \rceil - 1}]$.

Notice that there is a small-norm square matrix $\mathbf{D}$ which satisfies $\mathbf{VD} = [\mathbf{X}|\mathbf{G}]$. In particular, we can pick

$$\mathbf{D} := \left[\begin{array}{c|c} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{array}\right]$$

This is enough to obtain a short trapdoor; consider the matrix

$$\mathbf{E} := \left[\begin{array}{c|c} \mathbf{0} & \mathbf{I} \\ \mathbf{T_G} & -\mathbf{G}^{-1}(\mathbf{X}) \end{array}\right] \text{ where } \mathbf{T_G} \text{ is a trapdoor for } \mathbf{G} \tag{1}$$

It is easy to check that the product $\mathbf{DE}$ is short (since both $\mathbf{D}$ and $\mathbf{E}$ have low norm), has full rank (since both $\mathbf{D}$ and $\mathbf{E}$ are nonsingular), and lies in the nullspace of $\mathbf{V}$.

Note that this approach fails completely for sparse matrices. Even if we sample $\mathbf{A}$ to be sparse, $\mathbf{AR}$ will not be sparse anymore. Our idea is to start exactly as in the step above to construct $\mathbf{V}$. However, this matrix is not sparse at all. We come up with a procedure to "sparsify " $\mathbf{V}$ to obtain a wider k-sparse matrix $\mathbf{C}$. We take each column of $\mathbf{V}$ and replace it by a $n \times n \log n$ sparse matrix whose columns add up to the column we are replacing.

We construct a matrix $\mathbf{C}$ of the form $[\mathbf{C}_1|...|\mathbf{C}_L]$ as follows (L is the width of $\mathbf{V}$).

- Each column of $\mathbf{C}_i$ has a random locality pattern chosen from $\binom{n}{k}$.
- The columns of $\mathbf{C}_i$ sum up to the i^{th} column $\mathbf{v}_i$ of $\mathbf{V}$.
- Subject to the first two constraints $\mathbf{C}_i$ is random.

Such a $\mathbf{C}$ can be easily found provided the width of each $\mathbf{C}_i$ is large enough ($n \log n$ suffices). At this point, $\mathbf{C}$ is from the right distribution. Its locality pattern is randomly chosen, and because the distribution of $\mathbf{V}$ is statistically close to random, so is the distribution of $\mathbf{C}$.

This $\mathbf{C}$ is indeed our matrix. We will now show how to construct the trapdoor. We try to find a small-norm full rank square matrix $\mathbf{D}$ such that $\mathbf{CD}$ has the form $[\mathbf{X}|\mathbf{G}]$. Notice that if we can find this $\mathbf{D}$, we are already done (similar to [MP12]). We again use the same $\mathbf{E}$ from eq. (1). The trapdoor we are looking for is the product $\mathbf{DE}$, since it satisfies the following 3 properties:

- $\mathbf{CDE} = \mathbf{0}$
- $\mathbf{DE}$ has small norm.
- The columns of $\mathbf{DE}$ are linearly independent.

Let us now describe how we can find $\mathbf{D}$. Recall that we started with a matrix $\mathbf{V}$ such that

$$\mathbf{V} \cdot \begin{bmatrix} -\mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G}$$

By construction, we also have $\mathbf{C}_i[1\,1\,\cdots\,1]^T = \mathbf{V}_i$ for each column $\mathbf{v}_i$ of $\mathbf{V}$. Therefore, for any matrix $\mathbf{U}$, we can write $\mathbf{VU} = \mathbf{C}(\mathbf{U} \otimes [1\,1\,\cdots\,1]^T)$ So now our last equation gives us

$$\mathbf{C} \cdot \left(\begin{bmatrix} -\mathbf{R} \\ \mathbf{I} \end{bmatrix} \otimes \left\{ \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} n \log n \right\} \right) = \mathbf{G}$$

So now we have found a matrix $\mathbf{F}$ which satisfies $\mathbf{CF} = \mathbf{G}$, which was the difficult part. To extend it so that we can obtain $\mathbf{CD} = [\mathbf{X}|\mathbf{G}]$, we just choose an arbitrary full rank matrix not in the column span of $\mathbf{F}$ to concatenate to the left of $\mathbf{F}$ until the resulting matrix becomes square. We can do this by picking columns from the identity matrix until there is a linear dependence.

6.2 Formal Description

Theorem 6.1. *There is a PPT algorithm that on input $1^n, p, m \geq 3n^2 \log p \log n$, $k > 1$ outputs a matrix $\mathbf{A}$ distributed statistically close to $\mathcal{D}^m_{\mathsf{coeff},n,k,p}$, and a $O(m)$-good lattice trapdoor $\mathbf{T}$ for $\mathbf{A}$ with success probability $1 - o(1)$.*

This is an analog of theorem 3.1 for sparse LWE. Note that our algorithm requires a much higher number of samples as compared to the regular LWE trapdoor sampling process, $O(n^2 \log n)$ instead of $O(n)$. We will follow a very similar approach to [MP12] in our construction.

Algorithm $\mathcal{SAMP}$

1. Define $\ell := \frac{m}{n \log n} - \lceil n \log p \rceil$
2. Let $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}$ where $\mathbf{g} = [1\, 2\, 4\, \ldots\, 2^{\lceil \log p \rceil - 1}]$.
3. Sample matrices $\mathbf{A} \leftarrow \mathbb{Z}_p^{n \times \ell}$ and $\mathbf{R} \leftarrow \{0,1\}^{\ell \times \lceil n \log p \rceil}$
4. Define $\mathbf{V} := [\mathbf{A} | \mathbf{AR} + \mathbf{G}]$. We will denote the i^{th} column of $\mathbf{V}$ by $\mathbf{v}_i$.
5. For $i \in [1, 2, \ldots, \ell + \lceil n \log p \rceil]$:
   (a) Sample a $n \times n \log n$ sparse matrix $\mathbf{C}_i$ (with k nonzero entries in every column) with a random locality pattern such that the columns sum up to $\mathbf{v}_i$. If this is impossible because one or more rows of $\mathbf{C}_i$ are all 0s, abort.
6. Define $\mathbf{C} := [\mathbf{C}_1 | \mathbf{C}_2 | \cdots | \mathbf{C}_{\ell + \lceil n \log p \rceil}]$
7. Let $\mathbf{D} \in \{-1, 0, 1\}^{m \times m}$ be a full rank square matrix such that $\mathbf{CD} = [\mathbf{X} | \mathbf{G}]$. We claim such a matrix is easy to find (see lemma 6.2 for proof).
8. Define the $\mathbf{G}^{-1}$ function and trapdoor $\mathbf{T_G}$ for $\mathbf{G}$ the same way as in [MP12].
9. Define
$$\mathbf{E} := \left[\begin{array}{c|c} \mathbf{0} & \mathbf{I} \\ \mathbf{T_G} & -\mathbf{G}^{-1}(\mathbf{X}) \end{array}\right]$$
10. Return the sparse matrix $\mathbf{C}$ and the trapdoor $\mathbf{DE}$.

We will first prove two properties of the above algorithm.

Lemma 6.1. *$\mathcal{SAMP}$ aborts with probability $o(1)$.*

Proof. Our procedure only aborts if one of the $\mathbf{C}_i$ matrices sampled in step 5a contains at least one row with all zeros. Because the locality pattern is chosen randomly, the probability of any particular row being all zeros is

$$\left(1 - \frac{k}{n}\right)^{n \log n} \leq \exp(-k \log n) < n^k$$

Applying an union bound over all rows, we can conclude that the abort probability is at most $n^{-(k-1)} = o(1)$. □

Lemma 6.2. *There is an efficient algorithm to find $\mathbf{D}$ in step 7 of $\mathcal{SAMP}$.*

Proof. Denote the $n \log n \times 1$ all-1s vector by $\mathbf{h}$. Observe that step 5 ensures that $\mathbf{C}_i \mathbf{h} = \mathbf{v}_i$ for all i.

Therefore, the following must hold for any matrix $\mathbf{U}$ with $\ell + \lceil n \log p \rceil = m/(n \log n)$ rows:

$$\begin{aligned}
\mathbf{C} \cdot (\mathbf{I}_{m/(n\log n)} \otimes \mathbf{h}) &= \mathbf{V} \\
\implies \mathbf{VU} &= (\mathbf{C} \cdot (\mathbf{I}_{m/(n\log n)} \otimes \mathbf{h})) \cdot \mathbf{U} \\
&= \mathbf{C} \cdot ((\mathbf{I}_{m/(n\log n)} \otimes \mathbf{h}) \cdot \mathbf{U}) \\
&= \mathbf{C} \cdot ((\mathbf{I}_{m/(n\log n)} \otimes \mathbf{h}) \cdot (\mathbf{U} \otimes [1])) \\
&= \mathbf{C} \cdot ((\mathbf{I}_{m/(n\log n)} \cdot \mathbf{U}) \otimes (\mathbf{h} \cdot [1])) \\
&= \mathbf{C}(\mathbf{U} \otimes \mathbf{h})
\end{aligned}$$

By construction,

$$\mathbf{V} \cdot \begin{bmatrix} -\mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G} \implies \mathbf{C} \cdot \left(\begin{bmatrix} -\mathbf{R} \\ \mathbf{I} \end{bmatrix} \otimes \mathbf{h} \right) = \mathbf{G}$$

We have thus found a full rank matrix $\mathbf{F}$ whose infinity norm is 1, whose columns are independent, and which satisfies $\mathbf{CF} = \mathbf{G}$. We can now try to construct $\mathbf{D} = [\mathbf{S}|\mathbf{F}]$ where $\mathbf{S} \in \{0,1\}^{m \times (m - \lceil n \log p \rceil)}$. The columns of $\mathbf{S}$ are chosen from the columns of the identity matrix such that no linear dependence is formed. In other words, we can start with the matrix $[\mathbf{I}_m|\mathbf{F}]$ and then iteratively remove $n \log p$ columns from the $\mathbf{I}_m$ component which were in the column span of the other remaining columns. □

Proof of theorem 6.1. First, we will show that $\mathbf{DE}$ is a valid trapdoor.

- It is a square matrix since both $\mathbf{D}$ and $\mathbf{E}$ are.
- $\mathbf{D}$ is full rank by construction, and $\mathbf{E}$ is full rank since $\mathbf{I}$ and $\mathbf{T_G}$ are both full rank. Therefore their product is also full rank.
- Recall that $\mathbf{GT_G} = \mathbf{0}$ and $\mathbf{GG}^{-1}(\mathbf{X}) = \mathbf{X}$ by construction. Therefore, the product

  $$\mathbf{C}(\mathbf{DE}) = (\mathbf{CD})\mathbf{E} = [\mathbf{X}|\mathbf{G}]\mathbf{E} = [\mathbf{GT_G}|\mathbf{X} - \mathbf{GG}^{-1}(\mathbf{X})] = \mathbf{0}$$

  So the matrix $\mathbf{DE}$ is in the nullspace of $\mathbf{C}$.
- Both $\mathbf{T_G}$ and $\mathbf{G}^{-1}(\mathbf{X})$ have infinity norm at most 2. Therefore, $\|\mathbf{E}\|_\infty \leq 2$. We also have $\|\mathbf{D}\|_\infty = 1$ by construction. Hence their product has infinity norm at most $2m$.

We will now argue that $\mathbf{C}$ is statistically close to $\mathcal{D}^m_{\mathsf{coeff},n,k,p}$. Note that it suffices to show that $\mathbf{V}$ is statistically close to uniform, since each $\mathbf{C}_i$ is chosen independently from $\mathcal{D}^{n\log n}_{\mathsf{coeff},n,k,p}$ subject only to the constraint that their columns sum to $\mathbf{v}_i$. This follows from a straightforward application of the leftover hash lemma since $\ell \geq 2n \log p$.

□

Remark 6.1. Our algorithm requires a much higher number of samples as compared to the regular LWE trapdoor sampling process, $O(n^2 \log n)$ instead of $O(n)$.

7 Cryptanalysis

7.1 Sparse Vectors in the Kernel

One potential approach to attack sparse LWE could be (in the same vein as sparse Learning Parity with Noise) finding sparse vectors $\boldsymbol{x} \in \mathbb{Z}_p^{m\times 1}$ (not necessarily small) so that $\mathbf{Ax} = \mathbf{0}$. If this is true, given a t sparse $\mathbf{x}$ and a sLWE tuple $(\mathbf{A}, \mathbf{b} = \mathbf{sA} + \mathbf{e} \mod p)$, one can observe that:

$$\mathbf{b} \cdot \mathbf{x} = \mathbf{e} \cdot \mathbf{x}.$$

If the error coordinates for $\mathbf{e}$ lie in $[-B, B]$ for some B and t is sufficiently small so that $(2B + 1)^t \ll p$, then this will give an attack that runs in time roughly $B^{O(t)}$. The idea is that since $\mathbf{x}$ is t sparse, the product $\mathbf{e} \cdot \mathbf{x}$ touches upon only t coordinates of $\mathbf{e}$. There are $2B+1$ possible choices for each coordinate. One could brute-force and check if there is a setting that yields the inner product $\mathbf{b} \cdot \mathbf{x}$. One can show that for a typical $\mathbf{x}$ only one choice of $\mathbf{e}$ will agree with p when $(2B + 1)^t$ is sufficiently smaller than p. To rule out such attacks we ask for what sparsity t does there exist solutions to $\mathbf{Ax} = \mathbf{0}$. One can show a naive bound by lifting off analysis for the sparse LPN assumption. Such analysis has been done for sparse LPN over $\mathbb{Z}_2$ that can be helpful here.

The above attack can be significant if there exist very sparse vectors in the kernel. For instance, if t is a constant and B is polynomial, this attack could be implemented in polynomial time. Notice that when the number of samples $m \geq k\binom{n}{k} + 1$, by pigeonhole principle there must exist $k + 1$ coefficient vectors supported on the same set of k input variables. Since there are at least $k + 1$ vectors supported on the same set of k variables they must be linearly dependent. This allows us to find a t-sparse vector in the nullspace of $\mathbf{A}$ for $t \leq k + 1$.

Inspired by the literature on random k-XOR in the field of average case complexity, we analyze a certain graph expansion condition corresponding to the locality set of the column vectors of $\mathbf{A}$. If this condition is met, we would be guaranteed with high probability that t is not too small. As we will describe later, this condition turns out to be conservative.

7.1.1 Locality Graph Expansion

Our main theorem is the following:

Theorem 7.1. *Let $\mathbf{A}$ be a $\mathsf{sLWE}_{n,k,m,p,\sigma}$ matrix. With probability $1 - o(1)$, there does not exist any vector $\mathbf{x}$ in the kernel of $\mathbf{A}$ which has L or fewer nonzero entries in either of the following cases:*

- *$k \geq 3$ is a constant, $m = n^{\delta\frac{k}{2}}$ for some $\delta < 1$, and $L = \Theta(n^{1-\delta})$.*
- *$k = \mathsf{poly}(\log n)$, $m = \mathsf{poly}(n)$, and $L = \Theta(n)$*

We observe that if $\mathbf{Ax} = \mathbf{0}$ for a t-sparse $\mathbf{x}$, the following holds. Let us denote by $S_1, \ldots, S_m \subset [n]$ the locality sets (see Definition 4.2) corresponding to the column vectors of $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$.

Observe that if there exist a t sparse $\mathbf{x}$ so that $\mathbf{Ax} = \mathbf{0}$ this necessarily means that the following expansion property is satisfied:

$$\left| \bigcup_{i:\mathbf{x}[i]\neq 0} S_i \right| < \frac{k \cdot t}{2}. \tag{2}$$

This is because each column of $\mathbf{A}$ is exactly k sparse. A t sparse combination that yields the all 0 must cancel any variable that appears in the combination. Thus, each variable must appear

at least twice. We also state that this condition is a bit conservative. It is plausible that each variables appear multiple times yet the column vectors are linearly independent. This might be more prominent for large p's. We discuss this issue shortly.

We now compute an upper-bound on the probability of the above condition (Equation 2 being violated) as a function of n, k, m (note that the condition is independent of p). From this we can get the desired tradeoffs for n, k, m. This is a routine calculation.

Lemma 7.1. *For n, m, k and $t > k$ if the sets $S_1, \ldots, S_m$ are chosen at random from $\binom{n}{k}$ then the probability that the condition in Equation 2 is violated is bounded by $f_{\text{SPARSE}}(k, n, m)^t$ where $f_{\text{SPARSE}}(k, n, m) = e^{1+3k/2} \cdot (k/2)^{k/2} \cdot \frac{m \cdot t^{k/2-1}}{n^{k/2}}$.*

Proof. We observe that there are $\binom{m}{t}$ ways of sampling t sets, and $\binom{n}{\frac{kt}{2}}$ ways of choosing indices in $[n]$. Once t sets are selected, the probability that each set is supported only over the chosen $\frac{kt}{2}$ indices is bounded by $\frac{\binom{kt/2}{k}}{\binom{n}{k}}$.

The required probability is upper bounded by (using a union bound):

$$\binom{m}{t} \cdot \binom{n}{\frac{kt}{2}} \cdot \left(\frac{\binom{kt/2}{k}}{\binom{n}{k}} \right)^t$$

We now use the naive Binomial approximation $\left(\frac{a}{b}\right)^b \leq \binom{a}{b} \leq e^b \left(\frac{a}{b}\right)^b$. Rephrasing it, we get the following upper bound:

$$e^{t+kt/2+kt} \left(\frac{m}{t} \cdot \left(\frac{2n}{kt} \right)^{k/2} \cdot \left(\frac{kt}{2n} \right)^k \right)^t.$$

We remove all powers of t to examine the probability. We collect some other terms and rearrange them.

$$f_{\text{SPARSE}}(k, n, m) = e^{1+3k/2} \cdot (k/2)^{k/2} \cdot \frac{m \cdot t^{k/2-1}}{n^{k/2}}. \tag{3}$$

□

Our intention should be to set k and m as a function of n so that f_{SPARSE} is small for a large t.

One can examine that for a constant k, t is some polynomial provided $m = O(n^{k/2(1-\epsilon)})$ for some constant $\epsilon > 0$. When $m = \Omega(n^{k/2})$, one can find a constant number of subsets that violate Equation 2.

We now do some (asymptotic) case analysis for a constant k and setting $m = n^{k/2(1-\epsilon)}$. To set f_{SPARSE} to be $o(1)$, we consider another function g_{SPARSE} that is only bigger than f_{SPARSE} but more amenable to analysis.

$$g_{\text{SPARSE}}(k, n, m) = \frac{m \cdot (e^3 \cdot k \cdot t)^{k/2}}{n^{k/2}}. \tag{4}$$

To ensure g_{SPARSE} is small, we set:

$$m^{2/k} e^3 \cdot k \cdot t \ll n$$

Thus, this holds for:

$$t < \frac{n}{m^{2/k} \cdot e^3 \cdot k} \tag{5}$$

Proof of theorem 7.1. The theorem statement follows directly from eq. (5) by plugging in the appropriate values of n, m, k and setting L to the highest value of t that satisfies the inequality. □

For concrete parameter estimation one can compute exact values of eq. (5) and find out how this attack performs in practice. We anticipate if $B^t \gg p$, then, the attack does not apply.

7.1.2 Large p effect

It is clear that for the analysis for this attack we were overly conservative. Not every collection of t sets with less than $\frac{kt}{2}$ neighbours is likely to give a sparse vector in the kernel. Moreover, if p increases the chances of this happening should be really small. Nevertheless as described before while the calculation predicts security against polynomial time adversaries with $m \ll n^{\frac{k}{2}}$ when k is a constant, system over any p can be broken when $m > (k+1)\binom{n}{k}$. Thus, the prediction in terms of sample complexity is potentially only off by a square-root factor. We leave the analysis that utilizes the largeness of p for future work as a great open question. We show how to loosely incorporate this above so that it is better for concrete parameter estimation. Finding precise bounds is a great open question.

Claim 7.1. *If $m \leq n^{k-3}$, the following inequality holds with probability $1 - \mathsf{negl}(n)$ for all $t \leq L$ where $L = \mathsf{poly}(n)$ and k is a constant.*

$$\left| \bigcup_{i:\mathbf{x}[i]\neq 0} S_i \right| > 2 \cdot t \qquad \textit{where } \mathbf{x} \textit{ is } t\textit{-sparse} \tag{6}$$

Proof. We will upper bound the probability of eq. (6) being violated. Let us calculate the probability that there exist t equations which are supported on at most $2t$ variables. We can choose the equations in $\binom{m}{t}$ ways, choose the variables in $\binom{n}{2t}$ ways, and the probability of all these equations being supported on these variables is $(\binom{2t}{k}/\binom{n}{k})^t$. Applying the union bound, we now obtain

$$\begin{aligned}
&\Pr[\text{There exist some } t \text{ equations supported on at most } 2t \text{ variables}] \\
&\leq \binom{m}{t}\binom{n}{2t}\left(\frac{\binom{2t}{k}}{\binom{n}{k}}\right)^t \\
&\leq e^{3t}\left(\frac{m}{t}\right)^t\left(\frac{n}{2t}\right)^{2t}\left(\frac{2t}{n}\right)^{kt} \\
&= \left(\frac{2^{k-2}e^3 m t^{k-3}}{n^{k-2}}\right)^t \\
&\leq \left(\frac{2^{k-2}e^3 t^{k-3}}{n}\right)^t && \text{since } m < n^{k-3}
\end{aligned}$$

The above probability is asymptotically negligible in n whenever $t < \frac{n^{1/(k-3)}}{2e}$. Taking a union bound over all t less than this value finishes the proof. □

If the locality pattern of $\mathbf{A}$ satisfies the above combinatorial property, we can show that there does not exist a L-sparse $\mathbf{x}$ in the kernel of $\mathbf{A}$ with high probability. Since L above is polynomial in n, this rules out the sparse vector in kernel attack. Recall that sparse LPN over $\mathbb{F}_2$ is broken for $m = n^{k/2}$ samples due to this attack (see [AOW15, KMOW17] etc). However, our result holds as long as the sample complexity m is less than n^{k-3}, which is a substantial improvement over theorem 7.1. This is one of the ways increasing the modulus from 2 to some $p = \omega(m)$ increases security.

We can now prove the following result:

Theorem 7.2. *Let $\mathbf{A}$ be a* sLWE *matrix where the sample complexity m, dimension n and sparsity parameter $k = O(1)$ satisfy the relation $m < n^{k-3}$. If the modulus p is $\omega(m)$, then there does not exist any vector $\mathbf{x}$ in the kernel of $\mathbf{A}$ which has L or fewer nonzero entries for some $L = \mathsf{poly}(n)$ with probability $1 - o(1)$.*

Proof. We use Claim 7.1 to obtain L. If $\mathbf{x}$ is at most L-sparse, the probability that $\mathbf{Ax} = \mathbf{0}$ can now be upper bounded by the expression:

$$\sum_{t \in [L]} \binom{m}{t} p^t / p^{2t}$$

The reason for this is that for a fixed non-zero $\mathbf{x}$, $\mathbf{Ax}$ is randomly distributed on the variables in the support of $\bigcup_{\mathbf{x}_i \neq 0} \mathbf{a}_i$. By this property, the number of such variables is at least $2t$.

Note that,

$$\sum_{t \in [L]} \binom{m}{t} p^t / p^{2t}$$
$$< \sum_{t \in [L]} \frac{m^t}{p^t}.$$

This quantity is subconstant since $m/p = o(1)$. □

7.2 Dense Minor Lower-Bound Model

We now introduce a lower-bound model to provide an estimate of lattice attacks that exploit the sparsity structure of the equations. We call our attack the Dense-Minor attack. This captures a variety of different attacks as we will describe later and enables us to reason about security.

The rough idea is that all attacks that break LWE over dimension n requires at least $n + 1$ samples or else secrets are not information theoretically defined. The problem, with the low-locality structure of our assumption is that each sample is constant sparse (say k). Thus, it is plausible that exploiting the low locality, one can find $L+1$ samples supported over L coordinates for some small L, and then use an algorithm that breaks LWE with dimension L and $L+1$ samples. For example, if the number of samples is at least $k\binom{n}{k} + 1$, the pigeonhole principle guarantees the existence of some $k + 1$ columns which are supported on the same set of k variables. If k is a constant, we can solve the resulting k-dimensional LWE problem in constant time; this points to an attack against sparse LWE with constant sparsity and polynomially many samples.

We show how to set parameters to avoid this. In particular, setting the number of samples a reasonable polynomial such as $m = n^2$ or n^4 for target applications, we will aim to find k such that

L turns out to be at least linear in n e.g. $0.9n$ for the above attack strategy. This roughly states that any meaningful attack exploiting the sparsity must involve $\Omega(n)$ variables. Thus, one can appeal to concrete cryptanalysis to make a reasonable conjecture on the security of sparse LWE.

Conjecture 7.1. *$\mathsf{sLWE}_{n,k,m,p,\sigma}$ is at least as hard as $\mathsf{LWE}_{L,f(L),p,\sigma}$ for $L = \Theta(n)$, $k = \Omega(\log n)$, and f is some small polynomial.*

We will use the above heuristic for parameter estimation purposes. We show some examples in Section 7.4.

Definition 7.1 (Dense-Minor of order r)**.** *For any matrix $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$, we say that it has a dense minor of size $L \in [0, n]$, if there exists a subset $T \subseteq [n]$ of size L such that there are $r \cdot L$ column vectors supported entirely over indices given by T. Moreover, L is the minimum such number.*

We will typically be interested in r as $1 + 1/L$ or some constant (like 2), or even $r = \log p$.

Problem with a Small Dense Minor We observe that if for $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$ there is a dense minor of size L and order r, then, this means that we have $r \cdot L$ columns all supported at some L indices. This means that if one is given a sparse LWE sample $\mathbf{A}, \mathbf{sA} + \mathbf{e} \mod p$ one can pick out these $r \cdot L$ equations of the form $\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i \mod p\}_{i\in S}$ for some set $S \subseteq [m]$ of size at least $r \cdot L$. Here we assume $\mathbf{a}_i's$ are column vectors of $\mathbf{A}$. This new system is an LWE type system with $r \cdot L$ samples of dimension L. This can potentially be broken with access to an LWE oracle that solves LWE on dimension L, sample complexity $r \cdot L$ and the same modulus p.

Theorem 7.3. *Let n be the dimension, $k = \log n$ be the sparsity parameter and $m = n^\beta$ be the sample complexity for some constant $\beta > 1$. Let p be the modulus. We have that, with probability $1 - o(1/n)$ over choice of $\mathbf{A} \leftarrow \mathcal{D}^m_{\mathsf{coeff},n,k,p}$ (sampling each column independently from the distribution $\mathcal{D}_{\mathsf{coeff},n,k,p}$ specified in definition 4.1), $\mathbf{A}$ the minimum L for which $\mathbf{A}$ has a dense minor of order $1 + 1/L$ is $L = \Omega(n)$.*

Proof. Once again we will use a union-bound. There are $\binom{n}{t}$ ways of choosing t variables and $\binom{m}{t+1}$ ways of choosing $t + 1$ samples. The probability that each of these $t + 1$ equations depends on the chosen t variables is $\left(\frac{\binom{t}{k}}{\binom{n}{k}}\right)^{t+1}$. Thus, the probability of the dense-minor being smaller than some L is:

$$f_{\text{DENSEMINOR}} = \sum_{t\in[k,L]} \left(\frac{\binom{t}{k}}{\binom{n}{k}}\right)^{t+1} \cdot \binom{n}{t} \cdot \binom{m}{t+1}$$

We now make every term smaller than $1/n^2$ and this will prove the claim.

Since the expression involves Binomial coefficients we will use the naive asymptotics on the Binomial coefficients.

$$\binom{t}{k} \le e^k \left(\frac{t}{k}\right)^k$$
$$\binom{n}{k} \ge \left(\frac{n}{k}\right)^k$$
$$\binom{n}{t} \le e^t \left(\frac{n}{t}\right)^t$$
$$\binom{m}{t+1} \le e^{t+1} \left(\frac{m}{t+1}\right)^{t+1}$$

Then each term for t in the claimed expression f becomes:

$$\begin{aligned}
&\leq \exp\left(k(t+1)+2t+1\right) \cdot \frac{t^{k(t+1)}}{n^{k(t+1)}} \cdot \frac{n^t}{t^t} \cdot \left(\frac{m}{t+1}\right)^{t+1} \\
&\leq \exp\left((k+2)t+k+1\right) \cdot \frac{t^{t(k-1)}}{n^{t(k-1)}} \cdot \frac{n^{\beta(t+1)}}{t^{t+1}} \\
&\leq \exp\left((k+4)t\right) \cdot \frac{t^{tk}}{n^{tk-2t\beta-t}} \\
&\leq \left(\exp\left(k+4\right) \cdot t^k / n^{k-1-2\beta}\right)^t
\end{aligned}$$

We set the term inside that is raised to the power t to be lesser than $\frac{1}{4}$. This yields, the final term denoted by term

$$\begin{aligned}
\mathsf{term} \leq \exp\left(k+4\right) \cdot t^k / n^{k-1-2\beta} &\leq \frac{1}{4} \\
\impliedby t^k &\leq \frac{n^{k-1-2\beta}}{4\exp\left(k+4\right)}
\end{aligned}$$

Since, $k = \Omega(\log n)$ and β is a constant, we can enure that term is smaller than $1/4$ provided $t \leq \alpha n$ for some constant $\alpha > 0$.

This proves the theorem. □

Remark 7.1. We remark that most attacks care about $L \log p$ samples or cL samples for some $c > 1$. In the above theorem, we rule out minors with just $L + 1$ samples. This only makes our model stronger.

The expression above is quite loose as we make several approximations regarding Binomial coefficients. If one computes the expression

$$f_{\text{DENSEMINOR}} = \sum_{t \in [k,L]} \left(\frac{\binom{t}{k}}{\binom{n}{k}}\right)^{t+1} \cdot \binom{n}{t} \cdot \binom{m}{t+1}$$

it is possible to find tradeoffs exactly using a computer program. We use it to estimate the exact size of the dense minor. For more discussion on this, see section 7.4.

Now we describe how our dense-minor model captures attacks such as Arora-Ge, BKW and Lattice Reductions.

Capturing Arora-Ge Style Attacks Recall that for the Arora-Ge attack [AG11] on LWE, we assume that the infinity norm of the error vector is at most B for some $B < p/2$. Each LWE sample $(\mathbf{a}, b)$ gives rise to the following equation on the entries of the secret vector:

$$\prod_{j=-B}^{B} \left(\sum_{i=1}^{n} a_i s_i - b - j\right) = 0$$

The degree of each equation is $2B + 1$. Since there are n variables, the total number of terms is $\binom{n+2B+1}{n}$. We treat each unique term as a separate variable, and we can solve the resulting linear

system directly as long as the number of equations, m, exceeds the total number of terms. If we have fewer equations, we first multiply each equation by every possible term of degree $\leq d$. This increases the number of equations to $m\binom{n+d}{d}$ and the total number of terms to $\binom{n+2B+1+d}{n}$, and we can solve this system as soon as the former exceeds the latter.

Let us now consider what happens if we try to apply the same strategy to sparse LWE. The only difference is the additional constraint that each of the original equations are supported on k variables instead of n. Since our dense minor attack already provides an algorithm for solving sLWE when we are given more than $k\binom{n}{k}$ samples, we will assume that $m < k\binom{n}{k} < \binom{n+1}{k+1}$ below.

- If $2B+1 < k$, there is virtually no difference between LWE and sparse LWE for this approach. This is because in either case, the original equations do not have terms containing more than $2B+1$ distinct variables. We will therefore focus on the case where k is smaller than $2B+1$.

- The total number of terms is bounded above by the number of all degree $2B+1$ monomials supported on at most k out of n variables. This equals

$$\binom{n+1}{k+1}\binom{2B+1+k}{k} > \binom{n+1}{k+1} > m$$

  Clearly, the number of terms is greater than the number of equations, and trying to linearize the system naively does not work.

- We can consider increasing the number of equations by following the exact recipe used by Arora-Ge. For some d, multiply each equation by every possible monomial of degree at most d. The problem is that this breaks the locality pattern that distinguishes sLWE from LWE in the Arora-Ge approach, since each equation can be now supported on many more variables.

- We can instead multiply each equation by all the monomials of degree at most d which are supported on the same k variables as the equation. This preserves the locality, but does not suffice since the number of terms continue to exceed the number of equations

$$\binom{n+1}{k+1}\binom{2B+d+1+k}{k} > m\binom{d+k}{k}$$

Another natural way to utilize the sparsity is to try to find a large number of samples supported on the same set of t variables for some $t < n$, and then run the Arora-Ge algorithm on those samples with an effective dimension of t. The dense minor lower bound rules out this approach for any $t = o(n)$. Therefore, the effective dimension in this approach will be linear in n, and we would still require many more samples than what we have available.

Capturing BKW Style Attacks The BKW family of attacks ([BKW03, Lyu05, ACF+15]) look for a short vector $\mathbf{x}$ such that the product $\mathbf{Ax}$ is either 0 or very sparse. They first partition the columns of $\mathbf{A}$ into blocks of size B. New samples are then created by finding collisions in these blocks and reducing those blocks to zero by subtraction. Iteratively applying this process for each block, we can find vectors of magnitude $2^{n/B}$ in the nullspace of $\mathbf{A}$.

To execute this attack naively, we need access to a very large number of samples. For LWE, we can generate new samples by taking random linear combinations with small norm. However, new samples generated by combining columns of $\mathbf{A}$ destroy sparsity really fast. Therefore, the search step of BKW attack does not advantage sparse LWE in any way over standard LWE.

We can try to take advantage of the fact that the columns of sparse LWE are mostly filled with zeros, and hence it will be easier to find collisions in blocks. While this is indeed true for the first few blocks, observe that we subtract columns from each other at every iteration of BKW. Each subtraction operation would increase the number of nonzero entries in the remaining blocks exponentially, and hence the required sample complexity for the attack will at least be the same as that of regular LWE with a constant factor smaller dimension.

So a natural way to leverage sparsity is therefore to work with a dense minor of order $\log p$ of size L. Our theorem suggests that $L = \Omega(n)$, and we would again require more samples than we have available.

Capturing Lattice-Style Attacks Typically, lattice attacks are applied by applying lattice reduction to solve either:

- Finding a short $\mathbf{x}$ so that $\mathbf{Ax} = \mathbf{0}$. We already discussed why this approach doesn't have an advantage for sLWE over LWE in the BKW section.
- Solving BDD with respect to lattice $\mathcal{L} = \{\mathbf{xA}\}$ and target point $\mathbf{b} = \mathbf{sA} + \mathbf{e}$. The hardness of this depends on the ratio γ of $\lambda_1(\mathcal{L})$ and $||\mathbf{e}||$.

In section 4.1, we argued that the size of the shortest nonzero vector in $\mathcal{L}$ is similar to what it would have been if $\mathbf{A}$ was chosen randomly. In fact, the minimum distance is smaller than a random lattice. We therefore, expect γ to only be smaller. So if anything, so called primal attacks against LWE work worse than expected when they are given sparse LWE instances.

On apparent inspection of lattice reduction algorithms, naive application of those don't seem to exploit sparsity at all. That said, most of our column vectors are mutually perpendicular since the dot product of two column vectors is nonzero only if their locality patterns intersect (these columns are however large in norm roughly $\sqrt{k}p$, and so they do not necessarily represent a good basis). So, although unlikely, it is plausible that lattice reduction based methods turn out to do better. We leave the study of this phenomenon to future work.

The most natural way to exploit the small sparsity of the system is to find a dense minor and apply lattice algorithms to an instance with a smaller dimension. As long as the noise ratio is inverse polynomial in the dimension n, the best lattice based attacks take $2^{O(n)}$ time (see the fantastic thesis of Rachel Player [Pla18] or Albrecht et.al. [APS15, ACD+18] for an excellent survey on concrete running times). Thus if we are able to find dense minor of size (say) $n/2$ of order $\log p$, that is an exponential improvement in the running time over a dense matrix. Our dense-minor model captures such improvements. In fact, we rule out attacks that makes use of only $L + 1$ samples in L variables. Typical lattice attacks require at least cL (where $c > 1$ is some constant) or even $L \log p$ samples.

7.3 Open questions

We leave some exciting open problems for future work:

1. Are there other ways of leveraging sparsity besides the dense-minor method?
2. Analyze the performance of current lattice reduction algorithms with sparse LWE matrices.
3. Build new lattice reduction algorithms that naturally exploit sparsity

Exploiting Special Structure One aspect that we did not touch at all in this work is whether structured assumptions for sparse LWE are secure and/or useful for cryptographic constructions. Some variants include circular sparse LWE and small secret LWE. While we did not observe vulnerabilities in circular sparse LWE, we note that one has to be careful working with small secret sparse LWE. Unlike LWE where these two variants are equivalent [ACPS09, GKPV10, Mic18], it is not the case here.

For instance, if each secret coordinate is boolean, a single sample of k-sparse LWE would leak a lot of information about the secret vector. There are 2^k possible choices of secret coordinates occurring in each equation. If k is constant or logarithmic, this could effectively lead to a search algorithm for sparse LWE. We note that however, some applications might need a small secret and it might be plausible that once $(\|\mathbf{s}\|_\infty)^k$ is large enough, this variant becomes hard. We leave understanding small secret LWE as a great open problem.

7.4 Concrete Parameter Estimation

In this section, we focus on concrete security. Based on the cryptanalysis so far, we identify two major avenues of attack against sLWE specifically:

- Find a t-sparse vector in the kernel. To rule out this attack, we merely need to set $p < B^t$.
- Find a dense minor of order at least $1 + 1/L$. We write a script that estimates the size of the smallest dense minor as a function of the dimension n, sparsity k and sample complexity m. To see the code, please refer to appendix A.

To get a desired level of security using sLWE, we first look at the parameters for regular LWE to obtain the necessary dimension n' for m samples, modulus p and noise σ. We then compute a small k and a dimension n for which the size of dense minor in a $\mathsf{sLWE}_{n,k,m,p,\sigma}$ matrix exceeds n'. We set concrete parameters based on the following assumption:

Conjecture 7.2. *Let the expected size of the dense minor in* $\mathsf{sLWE}_{n,k,m,p,\sigma}$ *matrix be* $L > n'$*. Then the* $\mathsf{sLWE}_{n,k,m,p,\sigma}$ *problem is at least as hard as* $\mathsf{LWE}_{n',m,p,\sigma}$

We show a few recommended parameter settings in the table below. We use LWE dimension $n' = 2^{10}$ and database size N between 2^{26} and 2^{42} as per [HHC+23]. Note that the number of samples $m = \sqrt{N}$. For all of these settings we suggest setting the modulus to either 2^{16}, 2^{32} or 2^{64} since there is generally hardware support for arithmetic modulo these values. We also set the noise parameter $\sigma = 6.4$. Neither σ nor p depends on any of n, m or k.

Sparsity parameter k \ Number of samples m	2^{13}	2^{17}	2^{21}
20	1218	1425	1656
30	1143	1265	1395
40	1110	1195	1285
50	1090	1158	1234

Table 1: The recommended dimension for sLWE whose security guarantees are equivalent to LWE with dimension 1024

We need not worry about the sparse vector in kernel attack since the size of the dense minor in each of the above settings is 1023, which is much larger than the logarithm of the modulus. Furthermore, it is evident from the above table that for typical parameters used in practice, we

only need to increase the dimension by a small constant factor < 1.5 in most cases to achieve equivalent security guarantees.

8 Applications of Sparse LWE

Our main applications lie in (secret-key) encryption schemes that support homomorphism. We start by recalling the definition of homomorphic encryption.

Definition 8.1 (Homomorphic Encryption for $\mathcal{F}$)**.** *A public key homomorphic encryption scheme $\mathcal{E}$ for function class $\mathcal{F}$ over message space $\mathcal{M}$ consists of the following efficient algorithms:*

- $\mathsf{Setup}(1^\lambda)$ *takes the security parameter λ. It outputs public parameters* pp*, which is implicitly given as input to all remaining algorithms.*
- $\mathsf{KeyGen}()$ *outputs a public, secret, and evaluation key* $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$ *takes public key* pk *and messages $m\mathcal{M}$. It outputs a ciphertext* ct*.*
- $\mathsf{Eval}(\mathsf{evk}, f, \{\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell\})$ *takes the evaluation key* evk*, ciphertext* $(\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell)$*, and a function $f \in \mathcal{F}$. It outputs an evaluated ciphertext* ct_f*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_f)$ *takes the secret key* sk *and evaluated ciphertext* ct_f*. It outputs a decrypted message y.*

Correctness. *The scheme is correct if for all $\lambda \in \mathbb{N}$, function $f \in \mathcal{F}$ defined over $\mathcal{M}^\ell$, and messages $m_1, \cdots, m_\ell \in \mathcal{M}^\ell$:*

$$\Pr\left[y = f(m_1, \cdots, m_\ell) \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{KeyGen}() \\ \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i) \\ \mathsf{ct}_f \leftarrow \mathsf{Eval}(\mathsf{evk}, f, \{\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell\}) \\ y \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_f) \end{array} \right] \geq 1 - \mathsf{negl}(\lambda)\ .$$

A secret key homomorphic encryption scheme $\mathcal{E}$ is one defined as above except that $\mathsf{pk} = \mathsf{sk}$.

Compactness. *We say that a scheme is compact if:*

- *The running time of encryption is polynomial in the message length and sublinear in the function size.*
- *The size of the evaluated ciphertext is sublinear in the function size.*

For regular circuits, we call the scheme satisfies levelled compact if instead of sizes/running times being sublinear in function size they grow polynomially in the depth of the circuit (but not in its size). A scheme is fully compact, if size/running times are independent of the circuit size.

Definition 8.2 (Semantic security)**.** *A public key homomorphic scheme $\mathcal{E}$ is semantically secure if: For any sequence of pairs of messages $\{m_0, m_1 \in \mathcal{M}\}_\lambda$ the following distributions are computationally indistinguishable:*

$$\left\{ \mathsf{pp}, \mathsf{pk}, \mathsf{evk}, \mathsf{Enc}(\mathsf{pk}, m_0) \;\middle|\; \begin{array}{rl} \mathsf{pp} & \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) & \leftarrow \mathsf{KeyGen}() \end{array} \right\}_\lambda$$

$$\approx \quad \left\{ \mathsf{pp}, \mathsf{pk}, \mathsf{evk}, \mathsf{Enc}(\mathsf{pk}, m_1) \;\middle|\; \begin{array}{rl} \mathsf{pp} & \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) & \leftarrow \mathsf{KeyGen}() \end{array} \right\}_\lambda$$

A secret key homomorphic scheme $\mathcal{E}$ is semantically secure if: For any polynomial $\ell = \ell(\lambda)$ and any sequence of pairs of ℓ messages $\{\{m_{i0}\}_{i\in[\ell]}, \{m_{i1}\}_{i\in[\ell]}\}_\lambda$ the following distributions are computationally indistinguishable:

$$\left\{ \mathsf{pp}, \mathsf{evk}, \{\mathsf{Enc}(\mathsf{sk}, m_{i0})\}_{i\in[\ell]} \;\middle|\; \begin{array}{rl} \mathsf{pp} & \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{sk}, \mathsf{evk}) & \leftarrow \mathsf{KeyGen}() \end{array} \right\}_\lambda$$

$$\approx \quad \left\{ \mathsf{pp}, \mathsf{evk}, \{\mathsf{Enc}(\mathsf{sk}, m_{i1})\}_{i\in[\ell]} \;\middle|\; \begin{array}{rl} \mathsf{pp} & \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{sk}, \mathsf{evk}) & \leftarrow \mathsf{KeyGen}() \end{array} \right\}_\lambda$$

8.1 Linearly Homomorphic Encryption

We recall the Brakerski-Vaikuntanahthan secret key homomorphic encryption scheme from LWE [BV11, BGV12b]. We then focus on analyzing the efficiency improvement when the scheme is instantiated with sparse LWE and used for computing linear functions, in comparison with instantiation using LWE or Ring LWE.

The Basic Secret Key Homomorphic Encryption Scheme

Let $p = p(\lambda)$ be a modulus and the message space be $\mathbb{Z}_p$.

$\mathsf{pp} \leftarrow \mathsf{sklhe.Setup}(1^\lambda, 1^D)$**:** Based on the security parameter λ, choose a modulus q, where $\gcd(p, q) = 1$, a dimension n, a B-bounded distribution of noise elements χ. Output $\mathsf{pp} := (q, n, \chi)$.

$(\mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{sklhe.KeyGen}()$**:** Sample a secret key $\boldsymbol{s}' \leftarrow \mathbb{Z}_q^{1\times n}$ and set $\boldsymbol{s} = (\boldsymbol{s}', -1)$. Output $\mathsf{sk} = \boldsymbol{s}$.

$\mathsf{ct} \leftarrow \mathsf{sklhe.Enc}(\mathsf{sk}, m)$**:** Parse the message m as an element of $\mathbb{Z}_p$. Sample a random k-sparse coefficient vector $\boldsymbol{a} \in \mathbb{Z}_q^n$, and a noise $e \leftarrow \chi$. Output $\mathsf{ct}^\mathsf{T} := (\boldsymbol{a}^\mathsf{T}, b = \boldsymbol{s}^\mathsf{T}\boldsymbol{a} + m + e \cdot p \bmod q)$.

$m \leftarrow \mathsf{sklhe.Dec}(\boldsymbol{s}, \mathsf{ct})$**:** Parse $\mathsf{ct} = (\boldsymbol{a}, b) \in \mathbb{Z}_q$. Output $m := (b - \boldsymbol{s}^\mathsf{T}\boldsymbol{a}) \bmod q, p$.

$\mathsf{ct} \leftarrow \mathsf{sklhe.Add}(\mathsf{ct}_1, \mathsf{ct}_2, l)$**:** This algorithm takes two ciphertexts and scalar $l \in \mathbb{Z}_p$ and homomorphically computes $m_1 + l \cdot m_2 \bmod p$. Output $\mathsf{ct} := \mathsf{ct}_1 + l \cdot \mathsf{ct}_2 \bmod q$.

Using the above operation, one can evaluate any linear function L specified by a coefficient vector $\boldsymbol{l} \in \mathbb{Z}_p^\ell$, on fresh ciphertexts $\mathsf{ct}_1, \cdots, \mathsf{ct}_\ell$ with noises bounded by B, and obtain and output ciphertext ct with noise bounded by $\bar{B} = \ell \cdot p \cdot B$. Decryption correctness is guaranteed if $p \cdot \bar{B} \le q$, that is, $q > \ell \cdot p^2 B$.

Sparse Representation and Storage In the above scheme, while the sk is a dense vector in $\mathbb{Z}_q^{1\times n}$, the ciphertext ct is only $(k+1)$-sparse. We can store ct using the following succinct representation. For every integer vector $\boldsymbol{a}$ in Z_Q^n, define

$$\begin{array}{rl} \text{indicator set:} & I_{\boldsymbol{a}} = \{i_j \mid i_j \in [n] s.t.\ a_{i_j} \neq 0\}, \\ \text{sparsity of } \boldsymbol{a}\text{:} & k_{\boldsymbol{a}} = |I_{\boldsymbol{a}}| \\ \text{succinct representation:} & \widetilde{\boldsymbol{a}} := \mathsf{SparseRep}(\boldsymbol{a}) = \{(i_j, a_{i_j}) \mid i_j \in S_{\boldsymbol{a}}\} \end{array}$$

The size of the sparse representation is only $k_{\boldsymbol{a}}(\log n + \log Q)$, as opposed to $n \log Q$. As a result, encryption scheme has a good rate as shown below. In particular, when $\log p = \Omega(\log n + \log q)$,

the rate is $O(k)$, which could be as small as a super-constant, or even constant if the number of ciphertexts encrypted under $\boldsymbol{s}$ is a bounded polynomial.

$$\text{rate} = \frac{|\mathsf{ct}|}{|m|} = \frac{O(k(\log n + \log q))}{\log p} \qquad = O(k) \text{ if } \log p = \Omega(\log n + \log q)$$

Scheme & Assumptions	# of ciphertexts	rate	encryption overhead	evaluation overhead
Elgamal & DDH [1]	unbounded	$O(\lambda)$	$\widetilde{O}(\lambda)$	$O(\lambda)$
Elgamal & DCR [Pai99] [2]	unbounded	$O(1)$	$\widetilde{O}(\lambda)$	$O(\lambda)$
BV [BV11] & LWE [3]	unbounded	$O(n)$	$O(n \log\log q)$	$O(n)$
BV & RLWE [3]	unbounded	$O(n)$	$\widetilde{O}(n \log\log q)$	$O(n)$
[GHS12] & RLWE	unbounded	$O(1)$	$O(1)$	$\mathrm{polylog}(n)$ [4]
BV & $O(1)$-sparse LWE (Ours) [3]	bounded poly	$O(1)$	$O(\log\log q)$	$O(1)$
BV & $\omega(1)$-sparse LWE (Ours) [3]	unbounded	$\omega(1)$	$\omega(\log\log q)$	$\omega(1)$

[1] Use a group G with $|G| = 2^{\Omega(\lambda)}$ and generator g. For $\boldsymbol{m} \in \{0,1\}^\ell$, $\mathsf{ct} = g^{\boldsymbol{r}}, g^{s\boldsymbol{r}+\boldsymbol{m}}$. Linear function $\boldsymbol{l} \in \mathbb{Z}_{|G|}^\ell$ is evaluated over $\mathbb{Z}_{|G|}$ and output must be small in order to decrypt.

[2] Use group $\mathbb{Z}_{N^2}^*$ for an RSA integer $N = 2^{\Omega(\lambda)}$, with hard group generator g. For $\boldsymbol{m} \in \mathbb{Z}_{N^e}^\ell$, $\mathsf{ct} = g^{\boldsymbol{r}}, g^{s\boldsymbol{r}}(1+N)^{\boldsymbol{m}}$. Linear function $\boldsymbol{l} \in \mathbb{Z}_N^\ell$ is evaluated over $\mathbb{Z}_N$.

[3] For all the BV schemes, we consider the case $\log p = \Omega(\log n + \log q)$.

[4] The scheme [GHS12] has large polylogarithmic overhead.

Table 2: Efficiency comparison of different linearly homomorphic schemes.

Efficiency Analysis We now analyze the efficiency overhead, in particular, we measure the encryption overhead as ratio between the encryption time and the number of bits encrypted, and the evaluation overhead as the ratio between the evaluation time and the time for evaluating the linear function in the clear.

The sparsity of the ciphertext can be leveraged for efficiency in the RAM model, where one can access an element in a vector or matrix at index $i \in [n]$ in logarithmic time $O(\log n)$ (note the index already has length $O(\log n)$). In contrast, in the circuit model this takes linear time. Consider encryption, where the dense secret key $\boldsymbol{s}$ is multiplied with a k-sparse coefficient vector $\boldsymbol{a}$, in the RAM model, this can be implemented in time $O(k(\log n + \log q \log\log q))$, involving fetching all the elements s_j for $j \in I_{\boldsymbol{a}}$, and performing a k-wise inner product between $\boldsymbol{s}_{I_{\boldsymbol{a}}}$ and $\boldsymbol{a}_{I_{\boldsymbol{a}}}$. The evaluation of linear function $\boldsymbol{l} \in \mathbb{Z}_p^\ell$ can be done using ℓ invocation of $\mathsf{sklhe.Add}(\mathsf{ct}, \mathsf{ct}_i, l_i)$, where ct_i is the ciphertext of the i^{th} input element. Since ct_i is $(k+1)$-sparse, this can be implemented in time $O(k(\log n + \log q \log\log q))$.

$$\text{encryption overhead} = \frac{T_{\mathsf{Enc}}}{|m|} = \frac{O(k(\log n + \log q \log\log q))}{\log p} \qquad = O(k \log\log q)$$

$$\text{evaluation overhead} = \frac{T_{\mathsf{Eval}}}{T_L} = \frac{O(\ell k(\log n + \log q \log\log q))}{O(\ell \log p \log\log p)} \qquad = O(k)$$

The right hand side equality holds when $\log p = \Omega(\log n + \log q)$. By setting k to be super-constant or even constant, we get nearly constant overheads.

Comparison with other secret-key LHE To the best of our knowledge, the BV scheme instantiated with sparse LWE is the only scheme achieving nearly constant overheads in all aspects, storage, encryption time, and evaluation time. A comparison with previous schemes is provided in table 8.1.

The most competitive scheme is by [GHS12] instantiated by RLWE, which presented a polylogarithmic overhead FHE scheme. To achieve this low overhead, they use the BV/BGV encryption scheme to encrypt a vector of messages $\mathbb{Z}_p^n$ in the same ciphertext, resulting in a "packed" ciphertexts. These packed ciphertexts natively support SIMD computation i.e., coordinate-wise addition and multiplication. To support general, non SIMD, computation, [GHS12] introduced novel methods for permuting elements encrypted in the same ciphertexts, and combining elements in different ciphertexts into the same ciphertexts. Combined with the use of permutation networks, they can implement arbitrary permutation over elements encrypted in multiple ciphertexts with polylogarithmic amortized overhead. Since general circuit computation can be implemented using SIMD operation and permutation, they achieve homomorphic evaluation with polylogarithmic amortized overhead. The only drawback is that the polylogarithmic overhead is very high and unlikely going to be practical.

8.1.1 Applications to Private Information Retrieval

A nice application that could help test concrete efficiency gains afforded by sparse LWE could be private-information retrieval schemes. In particular, consider the practical scheme SimplePIR [HHC+23] that builds on top of [KO97] and instantiates the linearly-homomorphic encryption scheme using LHE from LWE. The idea in that scheme is that the data $\widehat{\mathbf{D}} \in \{0,1\}^N$ is processed into a square shape yielding a matrix $\mathbf{D} \in \{0,1\}^{\sqrt{N}\times\sqrt{N}}$. For $i \in [\sqrt{N}]$, we let $\mathbf{D}_i$ as the i^{th} column of $\mathbf{A}$.

The client samples a matrix $\mathbf{A} \in \mathbb{Z}_p^{n\times\sqrt{N}}$ at random from the LWE distribution. To fetch the date index (i,j) the client computes a query $\mathbf{b} = \mathbf{sA}+2\cdot\mathbf{e}+\mathbf{v}_i$ where $\mathbf{v}_i \in \{0,1\}^{1\times\sqrt{N}}$ is an indicator vector with 1 at the i^{th} coordinate. The server computes $(\mathbf{bD}_1,\ldots,\mathbf{bD}_{\sqrt{N}})$ and $(\mathbf{AD}_1,\ldots,\mathbf{AD}_{\sqrt{N}})$ and sends these to the client. Note that $\mathbf{bD}_j \approx \mathbf{sAD}_j + \mathbf{D}[i,j]$ upto small and even error. The client can therefore derive $\mathbf{D}[i,j]$ using the secret $\mathbf{s}$. The scheme can be proven to be secure due to the security of LWE. Note that the total communication from client to the server is $\widetilde{O}(n\sqrt{N})$ where n is the dimension of LWE as the client needs to send $\mathbf{A}$ along with $\mathbf{b}$. The query compute time is also $\widetilde{O}(n\sqrt{N})$ which is also the time to compute $\mathbf{b}$. We note that the client communication can be improved to $\widetilde{O}(\sqrt{N})$ if $\mathbf{AD}$ can be precomputed. We will assume this in the following discussion. For server, given the preprocessing, the computation consists of evaluating $\mathbf{bD}$ which takes $\widetilde{O}(N)$ times but the communication is $\widetilde{O}(\sqrt{N})$. The downside to this scheme is that if a data entry needs to be changed, it takes $\widetilde{O}(n)$ time to change the preprocessing $\mathbf{AD}$.

If one used sparse LWE instead, it will reduce the client query generation time to $\widetilde{O}(k\sqrt{N})$ as computing LWE samples will now be faster. This might be useful if the client is low-complexity. Further, the update time reduces to $\widetilde{O}(k)$ as opposed to $\widetilde{O}(n)$. This might be useful in dynamic settings where the update needs to be faster. To summarize, we have the following comparison.

	LWE	k-sparse LWE
Total Communication	$\widetilde{O}(\sqrt{N})$	$\widetilde{O}(\sqrt{N})$
Update Time	$\widetilde{O}(n)$	$\widetilde{O}(k)$
Client Query Time	$\widetilde{O}(n\cdot\sqrt{N})$	$\widetilde{O}(k\cdot\sqrt{N})$

We leave concretely implementing the PIR scheme for a future update.

8.2 Constant Degree Homomorphic Encryption

We now describe how to perform homomorphic multiplication. In order to achieve the best efficiency for evaluating constant-degree polynomials, we will make two modifications to the Brakerski-Gentry-Vaikuntanathan [BGV12b] homomorphic multiplication. First, instead of using the gadget matrix consisting of powers of 2, we will use the gadget matrix consisting of powers a larger base, in particular, p the modulus of the message space. Second, instead of performing modulus reduction on the ciphertexts, we will perform the modulus reduction on the secret instead. The second modification is necessary since we want to rely on large secret sparse LWE instead of small secret sparse LWE. In contrast, standard modulus reduction (performed on ciphertexts) relies on having small secrets in order to ensure that errors introduced by rounding is bounded. While using small secrets is without loss of generality for LWE, it is not so for sparse LWE.

Let $\boldsymbol{g}_{Q,p} = (1, p, p^2, \cdots, p^{\lfloor \log_p Q \rfloor})$ be the gadget matrix in $\mathbb{Z}_Q$ with base p. Correspondingly, define operation $\mathsf{Decomp}(\boldsymbol{x}, p)$ that decomposes the integer vector $\boldsymbol{x}$ in base p. We will also make use of the subroutine Scale defined in [BV11, BGV12b].

Definition 8.3. *For integer vector $\boldsymbol{x} \in \mathbb{Z}_q$ and integers $q \geq p \geq r$, we define $\boldsymbol{x}' \leftarrow \mathsf{Scale}(\boldsymbol{x}, q, p, r)$ to be the $\mathbb{Z}_p$-vector closest to $(p/q) \cdot \boldsymbol{x}$ that satisfies $\boldsymbol{x}' = \boldsymbol{x} \bmod r$.*

They proved the following key lemma about the scale operation, showing that if the inner product between two vectors $\mathsf{ip}\boldsymbol{u}, \boldsymbol{v}$ over $\mathbb{Z}_q$ is not too large, then inner product with the scaled vector $\mathsf{ip}\boldsymbol{u}' = \mathsf{Scale}(q, p, r), \boldsymbol{v}$ over $\mathbb{Z}_p$ is congruent to $\mathsf{ip}\boldsymbol{u}, \boldsymbol{v} \bmod r$, and is roughly q/p times smaller than $\mathsf{ip}\boldsymbol{u}, \boldsymbol{v}$ with an rounding error bounded by $r\|\boldsymbol{v}\|_1/2$.

Lemma 8.1. *Let $q > p > r$ be positive integers satisfying $q = p = 1 \bmod r$. Let $\boldsymbol{u} \in \mathbb{Z}_q^n$ and $\boldsymbol{u}' \leftarrow \mathsf{Scale}(\boldsymbol{u}, q, p, r) \in \mathbb{Z}_p^n$. Then, for any $\boldsymbol{v} \in \mathbb{Z}_q^n$, the following holds*

$$\begin{aligned} &\textit{If,} && [\langle \boldsymbol{u}, \boldsymbol{v} \rangle]_q < q/2 - (q/p) \cdot (r/2) \cdot \|\boldsymbol{v}\|_1 \,, \\ &\textit{Then,} && [\langle \boldsymbol{u}', \boldsymbol{v} \rangle]_{p,r} = [\langle \boldsymbol{u}, \boldsymbol{v} \rangle]_{q,r} \quad \textit{and} \quad [\langle \boldsymbol{u}', \boldsymbol{v} \rangle]_p < \frac{q}{p}[\langle \boldsymbol{u}', \boldsymbol{v} \rangle]_q + \frac{1}{2} \cdot r \cdot \|\boldsymbol{v}\|_1 \end{aligned}$$

In the BGV scheme, the inner product between the secret key and a ciphertext $\langle \boldsymbol{s}, \mathsf{ct} \rangle$ over $\mathbb{Z}_q$ is the approximate message $m + p \cdot E$. The modulus reduction step scales the ciphertext to a smaller modulus q' with respect to p. By the above lemma, they have that $\langle \boldsymbol{s}, \mathsf{Scalect}, q, q', p \rangle \bmod q' = m + p \cdot e$, where the noise is bounded by $|e| \leq \frac{q}{q'}|E| + p \cdot \|\boldsymbol{s}\|_1$, which is small if $\boldsymbol{s}$ is short.

In our setting, we use large secret $\boldsymbol{s}$ and instead scale the secret. To start with, observe that $\langle \boldsymbol{s} \otimes \boldsymbol{g}_{q,p}, \mathsf{Decomp}(\mathsf{ct}, p) \rangle \bmod q = m + p \cdot E$. Therefore, we have $\langle \mathsf{Scale}(\boldsymbol{s} \otimes \boldsymbol{g}_{q,p}, q', p), \mathsf{Decomp}(\mathsf{ct}, p) \rangle \bmod q' = m + p \cdot e$, where the noise is bounded by $|e| \leq \frac{q}{q'}|E| + p \cdot \|\mathsf{Decomp}(\mathsf{ct}, p)\|_1$. In the actual scheme described below, we apply this scaling w.r.t. the tensor product of two ciphertexts $\mathsf{ct} = \mathsf{ct}_1 \otimes \mathsf{ct}_2$, and the secret key is $\boldsymbol{s} \otimes \boldsymbol{s}$. The operation is summarized in the following corollary.

Corollary 8.1. *Let $q > q' > p$ be positive integers satisfying $q = q' = 1 \bmod p$. For every $\boldsymbol{s} \in \mathbb{Z}_q^{1 \times (n+1)}$,*

and $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathbb{Z}_q^{(n+1)}$.

$$\begin{aligned} &\textit{If,} && m + p \cdot E = [(\boldsymbol{s} \otimes \boldsymbol{s}) \cdot (\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_q = [(\boldsymbol{s} \otimes \boldsymbol{s} \otimes \boldsymbol{g}_{q,p}) \cdot (\mathsf{Decomp}(\mathsf{ct}_1 \otimes \mathsf{ct}_2, p))]_q \\ &&& < q/2 - (q/q') \cdot (p/2) \cdot (n^2 \cdot p \cdot \log q / \log p) \,, \\ &\textit{Let} && \bar{\boldsymbol{t}} = \mathsf{Scale}(\boldsymbol{s} \otimes \boldsymbol{s} \otimes \boldsymbol{g}_{q,p}, q', p) \\ &&& m' + p \cdot e = [\bar{\boldsymbol{t}} \cdot (\mathsf{Decomp}(\mathsf{ct}_1 \otimes \mathsf{ct}_2, p))]_{q'} \\ &\textit{Then,} && m = m', \quad \textit{and} \quad |m' + p \cdot e| \leq \frac{q}{q'} |m + p \cdot E| + (p/2) \cdot (n^2 \cdot p \cdot \log q / \log p) \end{aligned}$$

Constant Degree Homomorphic Encryption Scheme

$\mathsf{pp} \leftarrow \mathsf{skhe.Setup}(1^\lambda, 1^D)$**:** Based on the security parameter λ and maximal depth D, choose a tower of decreasing modulus $q = q^{(D)} \geq q^{(D-1)} > q^{(D-2)} > \cdots > q^{(0)} > p$, where $\gcd(p, q^{(j)}) = 1$, $q^{(d)} = 1 \bmod p$, and $q^d / q^{d-1} > \Delta$. Also choose a dimension n, a B-bounded distribution of noise elements χ. Output $\mathsf{pp} := (\boldsymbol{q}, n, \chi, p)$, where $\boldsymbol{q}$ is the vector of modulus.

$\mathsf{sk}, \mathsf{evk} \leftarrow \mathsf{skhe.KeyGen}()$**:** For every $d \in [D]$, sample a random k-sparse coefficient matrix $\mathbf{B}^{(d)}$ over $\mathbb{Z}_{q^{(d)}}$, a noise vector $\boldsymbol{e}^{(d)} \leftarrow \chi^{1 \times m}$, and a secret key $\boldsymbol{s}'^{(d)} \leftarrow \bar{\mathbb{Z}}_p^{1 \times n}$. Let $\boldsymbol{s}^{(d)} := (\boldsymbol{s}'^{(d)}, -1)$. Compute the level d evaluation key as follows:

$$\begin{aligned} \boldsymbol{t}^{(d+1)} &:= \boldsymbol{s}^{(d+1)} \otimes \boldsymbol{s}^{(d+1)} \otimes \boldsymbol{g}_{q^{(d+1)},p} \bmod q^{(d+1)} && \in \mathbb{Z}_{q^{(d+1)}}^{1 \times m^{(d+1)}} \\ \bar{\boldsymbol{t}}^{(d+1)} &:= \mathsf{Scale}(\boldsymbol{t}, q^{(d+1)}, q^{(d)}, p) && \in \mathbb{Z}_{q^{(d)}}^{1 \times m^{(d+1)}} \\ \mathsf{evk}^{(d)} &:= \begin{pmatrix} \mathbf{B}^{(d)} \\ \boldsymbol{s}^{(d)} \mathbf{B}^{(d)} + \bar{\boldsymbol{t}}^{(d+1)} + p \cdot \boldsymbol{e}^{(d)} \bmod q^{(d)} \end{pmatrix} && \in \mathbb{Z}_{q^{(d)}}^{(n+1) \times m^{(d+1)}} \end{aligned}$$

Above dimension $m^{(d)} = \Theta(n \log q^{(d+1)} / \log p) = O(n \log q^{(D)} / \log p)$.

Output $(\mathsf{sk} = \{\boldsymbol{s}^d\}, \mathsf{evk} = \{\mathsf{evk}^d\})$.

$\mathsf{ct}^{(D)} \leftarrow \mathsf{skhe.Enc}(\mathsf{sk}, m)$**:** Parse the message m as an element of $\mathbb{Z}_p$. Sample a random k-sparse coefficient vector $\boldsymbol{a} \in \mathbb{Z}_{q^{(D)}}$, and a noise $e \leftarrow \chi$. Output $\mathsf{ct}^{(D)\mathsf{T}} := (\boldsymbol{a}^\mathsf{T}, b = \boldsymbol{s}'^{(D)} \boldsymbol{a} + m + e \cdot p \bmod q^{(D)})$.

$m \leftarrow \mathsf{skhe.Dec}(\mathsf{sk}, \mathsf{ct}^{(d)}, d)$**:** Parse $\mathsf{ct}^{(d)} = (\boldsymbol{a}, b) \in \mathbb{Z}_{q^{(d)}}$. Output $m := (b - \boldsymbol{s}^{(d)} \boldsymbol{a}) \bmod q^{(d)}, p$.

$\mathsf{ct}^{(d+1)} \leftarrow \mathsf{skhe.Mult}(\mathsf{ct}_1^{(d+1)}, \mathsf{ct}_2^{(d+1)})$**:** This algorithms takes two ciphertexts at level $d+1$ in $\mathbb{Z}_{q^{(d+1)}}^{n+1}$ and outputs a ciphertext of dimension $(n+1)^2$. Output $\mathsf{ct}^{(d+1)} := \mathsf{ct}_1^{(d+1)} \otimes \mathsf{ct}_2^{(d+1)} \bmod q^{(d+1)}$.

$\mathsf{ct}^{(d)} \leftarrow \mathsf{skhe.Reduce}(\mathsf{evk}, \mathsf{ct}^{(d+1)})$**:** This algorithms takes the evaluation key and a ciphertext at level $d+1$ of dimension $(n+1)^2$ and returns a ciphertext at level d of dimension $n+1$.

$$\mathsf{ct}^{(d)} := \mathsf{evk}^{(d)} \cdot \mathsf{Decomp}(\mathsf{ct}^{(d+1)}, p) \bmod q^{(d)} \,.$$

Evaluating Constant-Degree Polynomials and Noise Bounds Using the above operations, we can homomorphically evaluate a constant-degree polynomial, expressed as a sum of degree 2^D

monomials, by homomorphic evaluation of all monomial followed by homomorphic additions. To homomorphically compute a degree 2^{D+1-d} monomial $\prod_{j\in[2^{D+1-d}]} m_{i_j}$ (initially $d = D$, and at the last step $d = 1$), suppose we have already obtained ciphertext ct_1 encrypting $\prod_{j\in[2^{D-d}]} m_{i_j}$ and ct_2 encrypting $\prod_{j\in[2^{D-d}+1,2^{D+1-d}]} m_{i_j}$, both at level d. We can obtain a ciphertext ct encrypting $\prod_{j\in[2^{D+1-d}]} m_{i_j}$ at level $d-1$ by first invoking $\mathsf{skhe.Mult}$ on $\mathsf{ct}_1, \mathsf{ct}_2$ obtaining an intermediate ciphertext of dimension $(n+1)^2$, followed by $\mathsf{skhe.Reduce}$ to reduce the dimension to n while also reducing the modulus to $q^{(D-d)}$, i.e.,

$$\mathsf{ct} = \mathsf{evk}^{(d-1)} \cdot \mathsf{Decomp}(\mathsf{ct}_1 \otimes \mathsf{ct}_2) \bmod q^{(d-1)} .$$

Correctness Proof To see the correctness and analyze noise growth, consider the following setting

$$q^{(d)} = O(q^{(0)}\Delta^d),\ q^{(0)} = O(\Delta), \quad \Delta = 3r^2 \tag{7}$$
$$r > B \tag{8}$$
$$r^2 > 3(p/2) \cdot (n^2 \cdot p \cdot \log q^{(D)}/\log p) \tag{9}$$
$$r^2 > 3p^2Bn \tag{10}$$

We keep the invariant that ciphertexts at level d satisfy that

$$[\boldsymbol{s} \cdot \mathsf{ct}^{(d)}]_{q^{(d)}} = m + p \cdot e \le r^2$$

For fresh ciphertexts, i.e., at level D, this holds as $m + p \cdot e < pB < r^2$ (condition (8)). Suppose this holds for two level d ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$, we show it also holds for level $d-1$ ciphertexts ct described above. We want to bound

$$\begin{aligned}
&[\boldsymbol{s}^{(d-1)} \cdot \mathsf{ct}]_{q^{d-1}} \\
=&[\boldsymbol{s}^{(d-1)} \cdot \mathsf{evk}^{(d-1)} \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)}} \\
=&[(\bar{\boldsymbol{t}}^d + p \cdot \boldsymbol{e}^{(d-1)}) \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)}} \\
=&[p \cdot \boldsymbol{e}^{(d-1)} \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)}} \\
\le& p^2Bn \le r^2/3
\end{aligned}$$

The last inequality follows from condition (10), while the second last follows from the fact that $e^{(d-1)}$ is the noise in $\mathsf{evk}^{(d-1)}$ and is B bounded. To analyze the other term, $[\bar{\boldsymbol{t}}^d \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)}}$, we will rely on Corollary 8.1.

Before scaling, $[(\boldsymbol{s} \otimes \boldsymbol{s}) \cdot (\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d)}} < r^4$

$$\begin{aligned}
r^4 &< q^{(1)}/2 - r^4 && \text{[condition(7)]} \\
&= q^{(1)}/2 - (\Delta/3)r^2 \\
&< q^{(1)}/2 - (\Delta/3)(3(p/2) \cdot (n^2 \cdot p \cdot \log q^{(D)}/\log p)) && \text{[condition(9)]} \\
&< q^{(d)}/2 - \Delta((p/2) \cdot (n^2 \cdot p \cdot \log q^{(d)}/\log p)) && \text{[precondition of Cor 8.1]}
\end{aligned}$$

By Corollary 8.1,

$$[\bar{\boldsymbol{t}}^{(d)} \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)}} < (1/\Delta) \cdot r^4 + (p/2) \cdot (n^2 \cdot p \cdot \log q^{(d)}/\log p) < (2/3)r^2$$
$$[\bar{\boldsymbol{t}}^{(d)} \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)},p} = [(\boldsymbol{s}^{(d)} \otimes \boldsymbol{s}^{(d)} \otimes \boldsymbol{g}_{q^{(d)},p}) \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d)},p} = m_1 \cdot m_2$$

Therefore,

$$[\boldsymbol{s}^{(d-1)} \cdot \mathsf{ct}]_{q^{d-1}} = [(\bar{\boldsymbol{t}}^d + p \cdot \boldsymbol{e}^{(d-1)}) \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)}} \leq r^2$$
$$[\boldsymbol{s}^{(d-1)} \cdot \mathsf{ct}]_{q^{d-1},p} = [\bar{\boldsymbol{t}}^d \cdot \mathsf{Decomp}_p(\mathsf{ct}_1 \otimes \mathsf{ct}_2)]_{q^{(d-1)},p} = m_1 \cdot m_2$$

This ensures the correctness of evaluation and decryption.

Efficiency Analysis We now analyze efficiency in the following setting:

$$\log p = \Omega(\log B, \log n) = \Omega(\log r), \qquad q^{(d)} = p^{O(d)}, \qquad \log q^{(d)} / \log p = O(D)$$

We can upper bound the cost of each homomorphic multiplication by the cost of the final multiplication, where the two operand ciphertexts (at level 1) have the worst sparsity.

Let's first see how fast sparsity deteriorates. If two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$ at level d have sparsity k_d, $\mathsf{skhe.Mult}$ creates a ciphertext ct of sparsity k_d^2. Decomposition further increases sparsity to $k_d^2 \log q^{(d)} / \log p = k_d^2 O(D)$. Finally, $\mathsf{skhe.Reduce}$ increases sparsity further to $k_{d-1} = k_d^2 \cdot O(D) \cdot (k+1)$ since the sparsity of all the evaluation keys are $k+1$. Therefore, for $k \geq 3$,

$$k_d = k^{2^{D-d}} \cdot O(D)^{(2^{D-d}-1)/2} \cdot (k+1)^{(2^{D-d}-1)/2} < k^{2^{D-d+1}} O(D)^{2^{D-d-1}}$$

Each multiplication at level d involves at most $O(k_{d-1})$ non-zero $\mathbb{Z}_{q^{(d)}}$ elements. For each element, computation involved is random access – $O(\log n)$ time and $\mathbb{Z}_{q^{(d)}}$ addition and multiplication. Hence, the cost of multiplication is bounded by

$$k_{d-1} \cdot O(\log n + \log q^{(d)} \log\log q^{(d)}) \leq k^{2^{D+1}} \cdot O(D)^{2^{D-1}} \cdot O(\log n + D \log p \log\log p)$$

In comparison, a multiplication in the clear costs time $O(\log p \log\log p)$. Therefore the evaluation overhead is:

$$\text{evaluation overhead} \leq k^{2^{D+1}} O(D)^{2^{D-1}} \cdot O(D)$$

The overhead is a constant when $k = O(1)$ for releasing a bounded polynomial number of ciphertexts and is slightly super constant when $k = \omega(1)$ for unbounded number of ciphertexts.

A comparison of the efficiency of our scheme with previous schemes is provided in Table 8.2.

Scheme & Assumptions	# of ciphertexts	rate	encryption overhead	evaluation overhead
BV [BV11] & LWE [1]	unbounded	$O(n)$	$O(n \log\log q)$	$\Omega(n^3)$
BV & RLWE [1]	unbounded	$O(n)$	$\widetilde{O}(n \log\log q)$	$\Omega(n \log n)$
[GHS12] & RLWE	unbounded	$O(1)$	$O(1)$	$\mathrm{polylog}(n)$ [2]
BV & $O(1)$-sparse LWE (Ours) [1]	bounded poly	$O(1)$	$O(\log\log q^{(D)})$	$O(1)$
BV & $\omega(1)$-sparse LWE (Ours) [1]	unbounded	$\omega(1)$	$\omega(\log\log q^{(D)})$	$\omega(1)$

[1] For all the BV schemes, we consider the case $\log p = \Omega(\log n + \log q)$.
[2] The scheme [GHS12] has large polylogarithmic overhead.

Table 3: Efficiency comparison of different homomorphic schemes supporting constant-degree polynomials.

9 References

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the third round of the nist post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2022. 1

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 171–180. ACM Press, June 2010. 2, 4

[ACC+18] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018. 1

[ACD+18] Martin R Albrecht, Benjamin R Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, September 5–7, 2018, Proceedings 11*, pages 351–367. Springer, 2018. 6, 29

[ACF+14] Martin Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for lwe problems. 2014. 4

[ACF+15] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the bkw algorithm on lwe. *Des. Codes Cryptography*, 74(2):325–354, feb 2015. 4, 5, 28

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Heidelberg, August 2009. 30

[ADI+17a] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 223–254. Springer, Heidelberg, August 2017. 2

[ADI+17b] Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *Annual International Cryptology Conference*, pages 223–254. Springer, 2017. 17

[ADRSD15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2n time using discrete gaussian sampling. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 733–742, 2015. 4

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011. 2, 5, 27

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc^0. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, volume 4110 of *Lecture Notes in Computer Science*, pages 260–271. Springer, 2006. 2, 4

[AL16] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1087–1100. ACM Press, June 2016. 2, 4

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th FOCS*, pages 298–307. IEEE Computer Society Press, October 2003. 2, 4

[AOW15] Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In Venkatesan Guruswami, editor, *56th FOCS*, pages 689–708. IEEE Computer Society Press, October 2015. 2, 4, 25

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 4, 6, 29

[Ban93] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993. 4, 8

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th FOCS*, pages 320–331. IEEE Computer Society Press, October 2018. 1

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014. 1

[BGV12a] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325, 2012. 7

[BGV12b] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012. 32, 35

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003. 2, 5, 28

[BLNS23] Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. Cryptology ePrint Archive, Report 2023/077, 2023. `https://eprint.iacr.org/2023/077`. 1

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. 1, 5

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012. 1

[Bra18] Zvika Brakerski. Quantum FHE (almost) as secure as classical. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 67–95. Springer, Heidelberg, August 2018. 1

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. 1, 32, 33, 35, 38

[CJJ22] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for $\mathcal{P}$ from LWE. In *62nd FOCS*, pages 68–79. IEEE Computer Society Press, February 2022. 1

[CM01] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC. In Jiri Sgall, Ales Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings*, volume 2136 of *Lecture Notes in Computer Science*, pages 272–284. Springer, 2001. 2, 4

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Heidelberg. 2

[DIJL23] Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 315–348. Springer, Heidelberg, August 2023. 2

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *34th ACM STOC*, pages 534–543. ACM Press, May 2002. 2, 4

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009. 1

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, Heidelberg, April 2012. 3, 7, 33, 34, 38

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240. Tsinghua University Press, 2010. 30

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017. 1

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018. 1

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000. 2, 4

[GPV08a] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. 2, 4

[GPV08b] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008. 9

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013. 1

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013. 1

[HDCZ23] Alexandra Henzinger, Emma Dauterman, Henry Corrigan-Gibbs, and Nickolai Zeldovich. Private web search with tiptoe. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann, and Jonathan Mace, editors, *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 396–416. ACM, 2023. 1

[HHC+23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 3889–3905. USENIX Association, 2023. 1, 3, 6, 7, 30, 34

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 433–442. ACM Press, May 2008. 2, 4

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021. 2

[KMOW17] Pravesh K. Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 132–145. ACM Press, June 2017. 2, 4, 25

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, October 1997. 34

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982. 2, 5

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2009. 1

[LNP22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plan$\subset$con. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Heidelberg, August 2022. 1

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010. 1, 3

[Lyu05] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 378–389. Springer, 2005. 5, 28

[Mah18a] Urmila Mahadev. Classical homomorphic encryption for quantum circuits. In Mikkel Thorup, editor, *59th FOCS*, pages 332–338. IEEE Computer Society Press, October 2018. 1

[Mah18b] Urmila Mahadev. Classical verification of quantum computations. In Mikkel Thorup, editor, *59th FOCS*, pages 259–267. IEEE Computer Society Press, October 2018. 1

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671. Springer Science & Business Media, 2002. 18

[Mic18] Daniele Micciancio. On the hardness of learning with errors with binary secrets. *Theory of Computing*, 14(13):1–17, 2018. 30

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012. 18, 19, 20

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 21–39. Springer, Heidelberg, August 2013. 4, 9

[MST03] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *FOCS*, pages 136–145, 2003. 2, 4

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999. 33

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009. 1, 5

[Pla18] Rachel Player. *Parameter selection in lattice-based cryptography*. PhD thesis, Royal Holloway, University of London, 2018. 4, 6, 29

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005. 1, 5

[RSSS17] Miruna Rosca, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Middle-product learning with errors. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 283–297. Springer, Heidelberg, August 2017. 1

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987. 5

[Sch94] Claus-Peter Schnorr. Block reduced lattice bases and successive minima. *Comb. Probab. Comput.*, 3:507–522, 1994. 2

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66:181–199, 1994. 5

[SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.*, 71(1):57–81, 2014. 3

[Vai20] Vinod Vaikunthananthan. Cs 294 lecture 4: Worst-case to average-case reductions for lwe. Class at UC Berkeley, 2020. `http://people.csail.mit.edu/vinodv/CS294/lecture6.pdf`. 9

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017. 1

A Script for Concrete Security

The following code is used to calculate the expected size of the smallest dense minor for sLWE.

```
// Usage instructions:
//      Adjust the values of n,m,k at the top of the main method
//      Running the program will give you a value of n' up to which
//          the probability upper bound is at most 10%.
//
//      Our formula for probability upper bound is
//          \sum_{t=k}^{n'} nCt \times mC{t+1} \times (tCk/nCk)^{t+1}
//      The purely locality based attack will not find a smaller
//          sample of size n' as long as this value is quite low.

#include <iostream>
#include <vector>

using namespace std;

double term_t(int n, int m, int k, int t) {
    // Calculate nCt . mC(t+1) . (tCk / nCk)^{t+1}

    vector<int> numerator, denominator;
    numerator.clear();
    denominator.clear();

    // Expression for nCt
    for (int i = 1; i <= t; i++) {
        numerator.push_back(n-i+1);
        denominator.push_back(i);
    }

    // Expression for mC(t+1)
    for (int i = 1; i <= t+1; i++) {
        numerator.push_back(m-i+1);
        denominator.push_back(i);
    }

    // Raising to the power t+1
    for (int j = 0; j < t+1; j++) {
        // Expression for tCk/nCk
        for (int i = 0; i < k; i++) {
            numerator.push_back(t-i);
            denominator.push_back(n-i);
        }
    }

    // Evaluate num/den.
```

```
    double d = 1.0;
    while (true) {
        if (d < 0.000001) {
            // If d is already too small, prefer to multiply
            if (!numerator.empty()) {
                d *= (double)(numerator.back());
                numerator.pop_back();
            } else if (!denominator.empty()) {
                d /= (double)(denominator.back());
                denominator.pop_back();
            } else {
                break;
            }

            continue;
        }

        // Prefer to divide
        if (!denominator.empty()) {
            d /= (double)(denominator.back());
            denominator.pop_back();
        } else if (!numerator.empty()) {
            // The term should always be < 1 to be useful
            if (d > 2) {
                cerr << "Term evaluates to > 1";
                cerr << " for n = " << n;
                cerr << " for m = " << m;
                cerr << " for k = " << k;
                cerr << " for t = " << t << endl;
                return d;
            }
            d *= (double)(numerator.back());
            numerator.pop_back();
        } else {
            break;
        }
    }

    return d;
}

int main() {
    int n = 1200;
    int m = 2097152; // 2^21
    // int m = 524288; // 2^19
    int k = 50;

    double prob_upper_bound = 0.0;
```

```
    for (int t = k; t <= n; t++) {
        double d = term_t(n,m,k,t);
        if (prob_upper_bound + d > 0.1) {
            cout << "We reach a high probability at n'=" << t;
            cout << " n=" << n;
            cout << " m=" << m;
            cout << " k=" << k;
            cout << "\n The probability up to the previous term is at most ";
            cout << prob_upper_bound << endl;
            cout << "We suggest n' = " << t-1 << endl;
            return 0;
        }
        prob_upper_bound += d;
    }

    return 0;
}
```