A New Approach for LPN-based Pseudorandom Functions: Low-Depth and Key-Homomorphic

Youlong Ding[*] Aayush Jain[†] Ilan Komargodski[‡]

Abstract

We give new constructions of pseudorandom functions (PRFs) computable in NC^1 from (variants of the) Learning Parity with Noise (LPN) assumption. Prior to our work, the only NC^1-computable PRF from LPN-style assumptions was due to Boyle et al. (FOCS 2020) who constructed a *weak* PRF from a new heuristic variant of LPN called variable-density LPN. We give the following three results:

- A weak PRF computable in NC^1 from standard LPN.
- A (strong) *encoded-input* PRF (EI-PRF) computable in NC^1 from sparse LPN. An EI-PRF is a PRF whose input domain is restricted to an efficiently sampleable and recognizable set. The input encoding can be computed in $\mathsf{NC}^{1+\epsilon}$ for any constant $\epsilon > 0$, implying a strong PRF computable in $\mathsf{NC}^{1+\epsilon}$.
- A (strong) PRF computable in NC^1 from a (new, heuristic) *seeded LPN* assumption. In our assumption, columns of the public LPN matrix are generated by an n-wise independent distribution. Supporting evidence for the security of the assumption is given by showing resilience to linear tests.

As a bonus, all of our PRF constructions are key-homomorphic, an algebraic property that is useful in many symmetric-cryptography applications. No previously-known LPN-based PRFs are key-homomorphic, even if we completely ignore depth-efficiency. In fact, our constructions support key homomorphism for linear functions (and not only additive), a property that no previously-known PRF satisfies, including ones from LWE.

Additionally, all of our PRF constructions nicely fit into the substitution-permutation network (SPN) design framework used in modern block ciphers (e.g. AES). No prior PRF construction that has a reduction to a standard cryptographic assumptions (let alone LPN) has an SPN-like structure.

Technically, all of our constructions leverage a new *recursive derandomization* technique for LPN instances, which allows us to generate LPN error terms deterministically. This technique is inspired by a related idea from the LWE literature (Kim, EUROCRYPT 2020) for which devising an LPN analogue has been an outstanding open problem.

[*]Hebrew University of Jerusalem, Israel. Email: `youlong.ding@mail.huji.ac.il`.
[†]Carnegie Mellon University, USA. Email: `aayushja@andrew.cmu.edu`.
[‡]Hebrew University of Jerusalem, Israel; and NTT Research, USA. Email: `ilank@cs.huji.ac.il`.

Contents

1 Introduction

A pseudorandom function family (PRF) is a keyed function $f\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that $f(k, \cdot)$ for a random $k \leftarrow \mathcal{K}$ is indistinguishable from a random function $R(\cdot)$ by an efficient adversary who has oracle access to it. A *strong* PRF (or simply PRF) is secure against distinguishers that adaptively query the function on arbitrary inputs $x \in \mathcal{X}$, while for a *weak* PRF (WPRF) the distinguisher is only given samples $(x_i, f(k, x_i))$ for $k \leftarrow \mathcal{K}$ and uniformly random inputs $x_i \leftarrow \mathcal{X}$. Since the introduction of PRFs (due to Goldreich, Goldwasser, and Micali [GGM86]), they have become a fundamental cryptographic primitive and they have innumerable applications spanning virtually every sub-area in cryptography, complexity theory, and more. That being so, understanding the complexity of PRFs is a holy-grail of foundational cryptography.

It is well known (due to HILL [HILL99] and GGM [GGM86]) that PRFs are existentially equivalent to one-way functions, the minimal cryptographic assumption. However, this implication incurs a high cost in efficiency [HRV10, VZ12, BMM24]. Intriguingly, a central measure of efficiency of PRFs is their parallel complexity, or circuit depth, affecting central aspects of efficiency in applications (e.g., round complexity of secure computation protocols [DI06, IKOS08, BJKL21], realization of advanced cryptographic systems [GVW12, Zha24], and many more). Another motivation comes from the theory of computational complexity. The (conjectured) existence of a PRF in a certain complexity class offers some explanation for why it is "hard". Concretely, PRFs in a certain complexity class $\mathcal{C}$, imply that this class is not PAC-learnable [Val84, KV89] (some form of the converse implication also exists [IL90, BFKL94]).

Over the years, a considerable research effort has been made to understand the necessary assumptions needed to obtain depth-efficient PRFs (for example, [GGM86, HILL99, Kha93, NR97, NR99, NRR02, LW09, BPR12, AR16, BCG+20, LLHG21] to name a few). The basic comparison point is the original GGM [GGM86] construction which transforms a pseudorandom generator (PRG) on n input bits computable in depth $d(n)$ into a PRF computable in depth $\omega(d(n) + \log n)$, assuming the PRG has polynomial stretch (to allow n-ary branching in the GGM tree construction).

Constructions with asymptotically better depth are known from almost any standard assumption. Naor, Reingold, and Rosen [NR97, NRR02] gave constructions in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$ from DDH and factoring.[1] Lewko and Waters [LW09] gave a construction in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$ from the k-Lin assumption (which is possibly weaker than DDH). Banerjee, Peikert, and Rosen [BPR12] gave a construction in $\mathsf{TC}^0 \subseteq \mathsf{NC}^1$ from the ring variant of learning with errors assumption (ring-LWE), where the modulus to noise ratio is super-polynomial. Viola [Vio13] gave a PRF in $\mathsf{AC}^0[2] \subseteq \mathsf{NC}^1$ assuming sub-exponential hardness of factoring. Intriguingly, we still do not know of any logarithmic depth construction from LPN or any of its natural variants.

LPN-style assumption. Let $\mathcal{C}$ be a probabilistic code generation algorithm such that $\mathcal{C}(n, m)$ outputs a matrix in $\mathbb{F}_2^{n \times m}$. Let Ber_μ be the Bernoulli distribution with parameter $\mu \in (0, 1/2)$. An

[1] TC^0 consists of all languages decidable by constant depth and polynomial size Boolean circuits, containing only unbounded fan-in AND gates, OR gates, NOT gates, and majority gates. NC^1 consists of all languages decidable by polynomial size and logarithmic depth circuits, containing constant fan-in AND, OR, and NOT gates. AC^0 consists of all languages decidable by polynomial size and constant depth circuits, containing unbounded fan-in AND, OR, and NOT gates. $\mathsf{AC}^0[m]$ is similar to AC^0 except that the circuits can can additionally have unbounded fan-in MOD_m gates. ACC^0 is the union over all m of $\mathsf{AC}^0[m]$.

LPN-style assumption (over $\mathbb{F}_2$) postulates that the following two distributions are indistinguishable:

$$(\boldsymbol{A}, \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e}) \overset{c}{\approx} (\boldsymbol{A}, \boldsymbol{u}), \quad \text{where } \boldsymbol{A} \leftarrow \mathcal{C}(n, m),\ \boldsymbol{s} \leftarrow \mathbb{F}_2^n,\ \boldsymbol{e} \leftarrow \mathsf{Ber}_\mu^m,\ \boldsymbol{u} \leftarrow \mathbb{F}_2^m.$$

In the original formulation of the LPN assumption [BFKL94, Ale03a], $\mathcal{C}(n, m)$ outputs a uniformly random matrix. A very popular variant is called *sparse LPN* and there each entry of the $\boldsymbol{A}$ matrix is uniformly random conditioned on each column being sparse [Fei02, Ale03a, IKOS08, ABW10, BCGI18, CRR21, DIJL23]). More variants have been proposed and studied over the years (e.g., [HKL+12, MBD+18, MTSB13, BCG+20]) for interesting cryptographic applications. None of these known LPN variants are currently known to imply a logarithmic depth PRF.

While new assumptions in code-based cryptography have a great potential, leading to interesting applications [BCGI18, BCG+19b, BCG+19a, BCG+22, BCCD23], it often takes time for cryptanalytic study to stabilize [CRR21, CD23]. The main focus of this work is primarily on devising general techniques that enable design of low-depth PRFs from well-studied LPN variants including standard LPN and sparse LPN, both of which have been well-studied for a couple of decades.

Yu and Steinberger [YS16] constructed the first LPN-based PRF. Their PRF construction relied on the GGM transformation and so resulted with a $\log^{1+\epsilon} n$ depth construction. The only signal for the possible existence of a logarithmic depth LPN-based PRFs comes from a recent work [BCG+20] (see also [CD23]) who constructed a *weak* PRF computable in logarithmic depth from a new heuristic variant of the LPN assumption called "variable-density LPN".

Low-depth LPN-based PRFs are well motivated by the following reasons. First, it increases diversity in the assumptions implying efficient "post-quantum" PRFs. Second, besides asymptotic efficiency, it is plausible that low-depth LPN-based PRFs will be practically favorable to their LWE counterparts since modular multiplications and additions in LWE would be simplified to AND and XOR operations under LPN. This leaves a significant gap in our understanding of the landscape of LPN-based symmetric-key cryptography, leading us to the following main question:

Are there logarithmic depth PRFs based on (variants of) LPN?

Key-homomorphism. Let $(\mathcal{K}, \oplus)$, $(\mathcal{Y}, +)$ be two groups. A PRF $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is key-homomorphic if for *any two keys* $k_1, k_2 \in \mathcal{K}$, and any input $x \in \mathcal{X}$, it holds that $f(k_1 \oplus k_2, x) = f(k_1, x) + f(k_2, x)$ [NPR99, BLMR13]. Key-homomorphic PRFs (KH-PRF) have many useful applications in symmetric cryptography and give rise to distributed PRFs, symmetric-key proxy re-encryption, and updatable encryption [BLMR13, EPRS17, LT18, KLR19, BDGJ20, BEKS20], OT extension [Sch18], and single-server secure aggregation [LLPT23].

However, such KH-PRFs are only known to exist in the random oracle model. In the plain model, existing KH-PRFs constructions satisfy a relaxed notion called almost-key-homomorphism [BLMR13], where the homomorphism requirement is "approximately" correct, i.e., $f(k_1 \oplus k_2, x) \in f(k_1, x) + f(k_2, x) \pm \delta$ for some (typically small) "error" parameter $\delta > 0$. Fortunately, such variants of key-homomorphism turn out to suffice for all the applications mentioned above.[2] Throughout this paper, we will only be interested in constructions of KH-PRF that do not rely random oracle model and when we refer to "key-homomorphism", we actually mean its relaxed variant.

[2]The basic idea is to encode (via an appropriate error correcting code) everything that is to be masked by the output of the PRF.

To date, the only constructions of KH-PRFs known from well-established assumptions are based on LWE [BLMR13, BP14, Kim20].[3] No previously-known LPN-based PRF is key-homomorphic, even if we ignore non-trivial efficiency constraints. This raised the following question:

Are there key-homomorphic PRFs based on (variants of) LPN?

Bridging theory and practice. As discussed in [ABG$^+$14, CRR21], there are two main paradigms for designing cryptographic primitives. The first is the "theory-oriented" or "provable security" approach, which focuses on developing constructions whose security can be rigorously reduced to the hardness of well-established computational problems. The second, "practice-oriented" approach emphasizes achieving efficient constructions for specific functionalities, where security is based on fundamental cryptanalytic techniques rather than formal security reductions.

There is often a significant gap between these two approaches, which is most pronounced in the context of PRFs. As mentioned in [MV12], this gap is both quantitative and methodological. Quantitatively, PRFs are less efficient than their counterparts, such as block ciphers. Methodologically, modern block ciphers (e.g., AES) typically follow the substitution-permutation network (SPN) paradigm, whereas constructions of PRFs based on well-established assumptions in the literature often diverge from this paradigm.

The SPN paradigm (dating back to Shannon [Sha49]) captures structural properties of secure ciphers. Specifically, an SPN is computed over a number of rounds, where each round "confuses" the input by dividing it into bundles and applying a substitution function (S-box) to each bundle, "diffuses" the bundles by applying a matrix with certain "branching" properties, and then performs "key mixing" by XORing with the private round key; see Figure 1 for an illustration.

There have been exciting recent works trying to provide provable bounds on practically-used block-cipher constructions in SPN framework, e.g., [DSSL16, CDK$^+$18, LTV21, LPTV23]. However, the current understanding is that existing tools are insufficient in order to provide formal PRF-style security guarantees for block ciphers used in practice assuming any reasonable mathematical hypothesis. Designing new PRFs with SPN-like structure can be viewed as a complementary goal, trying to go from theory to practice instead of from practice to theory. Miles and Viola [MV12] proposed heuristic low-depth PRF candidates that are inspired by the SPN paradigm. To date, we are not aware of constructions of PRFs that have reductions to standard cryptographic assumptions (let alone LPN) that have an SPN-like structure. We believe that such successful attempts might yield conceptual insights that could (1) help argue security for practically used block ciphers (or a suitable modification thereof), or (2) help design efficient block ciphers with provable security.

1.1 Our Results

In this paper, we propose new PRF constructions that answer all of the above questions affirmatively and simultaneously. We show for the first time the possibility of building PRFs with logarithmic

[3]We also know constructions based on a decision-linear assumption in ℓ-linear maps [BLMR13], where ℓ is proportional to the security parameter. We only have well-established candidate constructions for $\ell = 2$.

evaluation depth from LPN-style assumptions. At the same time, our constructions are all key-homomorphic and nicely fit into the SPN paradigm. Next, we elaborate on our results, and we refer to Table 1.1 for a summary.

Result 1: Log-depth weak PRF from LPN. Our first result is the construction of weak PRFs in NC^1 from the standard LPN assumption. That is, $\boldsymbol{A} \in \mathcal{C}(n, m)$ outputs a uniformly random matrix $\mathbb{F}_2^{n\times m}$. As mentioned, the only previously-known weak PRF NC^1 from LPN-style assumptions was due to [BCG+20] and it relied on a new heuristic variant called variable-density LPN.

Apart from its independent interest as a weak PRF, our weak PRF construction serves to illustrate the core idea of our new technique underlying all of our constructions in a relatively simple manner. In a nutshell, our key technique involves a recursive de-randomization technique to introduce "deterministic" noise into LPN sample.

Theorem 1 (**informal; see Corollary 1**). *Assume that LPN with any inverse polynomially low noise is hard. Then, there exist key-homomorphic weak PRFs computable in* NC^1.

Result 2: Log-depth encoded-input PRF from sparse LPN. Leveraging our recursive derandomization technique and relying on the *sparse LPN* assumption, we construct a depth-efficient (strong) PRF. The construction, as is, can be placed in $\mathsf{NC}^{1+\epsilon}$ for any $\epsilon > 0$. However, a useful feature of the construction is that it can be made in NC^1 as an encoded-input PRF (EI-PRF). An EI-PRF [BIP+18] is defined similarly to a PRF except that its input domain is restricted to some efficiently sampleable and recognizable set.

From a complexity-theoretic perspective, it characterizes the minimal depth required to transform any adversarially chosen input into something unpredictable by any efficient adversary. From an application standpoint, EI-PRF can be used as drop-in replacement for strong PRFs in concrete applications. Low-depth EI-PRFs give rise to both efficient symmetric-key cryptosystems [BIP+18] and efficient advanced cryptographic primitives [Zha24].

Theorem 2 (**informal; see Corollary 2**). *Assume hardness of sparse LPN with any sub-polynomial sparsity (i.e., $t = n^{o(1)}$). Then, there exist key-homomorphic PRFs in $\mathsf{NC}^{1+\epsilon}$ for any $\epsilon > 0$, and key-homomorphic encoded-input PRFs in NC^1.*

We note that our strong PRFs in Theorem 2 only support a slightly super-polynomial input domain $\mathcal{X} = \{0, 1\}^{\omega(\log n)}$. One can use generic domain extension techniques to support an exponentially large input domain [Lev85, BHKN13, DS15]. While these transformations preserve the low depth of our PRFs, they do not preserve key-homomorphism.

Result 3: Log-depth strong PRF from seeded LPN. Lastly, we use our recursive derandomization technique and with an even stronger seeded LPN-style assumption in order to obtain strong PRFs in NC^1 (in the standard model). In our variant of LPN, termed *seeded LPN*, the matrix sampler $\mathcal{C}(n, m)$ output a matrix $\boldsymbol{A} \in \mathbb{F}_2^{n\times m}$ wherein the columns come from an n-wise independent distribution. Using this assumption we manage to construct the first logarithmic-depth PRF from an LPN-style assumption.

Theorem 3 (informal; see Corollary 3). *Assume the hardness of seeded LPN. Then, there exist key-homomorphic PRFs in* NC^1.

While we were not able to break our seeded LPN assumption, admittedly, it is a new and strong assumption, and as such a healthy dose of skepticism is warranted. To give some formal confidence in the validity of the assumption, we perform initial cryptanalysis showing that it is secure within the "linear test framework". Within this framework, the adversary only tries to detect a bias in the observed samples by computing linear functions of these samples. As argued by [CRR21, BCG+22], this framework captures most known attacks on LPN-style assumptions (Information-Set Decoding [Pra62], BKW [BKW00], Gaussian Elimination [EKM17], Statistical Decoding [Al 01], and more).

We note that our strong PRFs in Theorem 3 only support a slightly super-polynomial input domain $\mathcal{X} = \{0,1\}^{\omega(\log n)}$. To support an exponentially large input domain, one can either use generic domain extension techniques discussed above, or settle for non-adaptive security.

Remark 1. (Concrete security) In terms of the concrete security (T, Q, ϵ) of PRFs (where any adversary making at most Q queries and running in time T has an advantage of at most ϵ), all our PRFs are only *provably secure* under (slightly) superpolymial T, Q, and $1/\epsilon$. Note that this is true even if we assume subexponential hardness for LPN problem. However, we are not aware of any quasipolynomial-time attacks that break the security of our PRFs.

Key homomorphism for linear functions. All of our constructions satisfy arbitrary linear key homomorphic operations. Specifically, we have $\lambda_1 \cdot F(k_1, x) + \lambda_2 \cdot F(k_2, x) \in F(\lambda_1 \cdot k_1 \oplus \lambda_2 \cdot k_2, x) \pm \delta$ for *any integers* $\lambda_1, \lambda_2 \in \mathbb{Z}$, which might be helpful in future applications requiring exact reconstruction or large combinations. In comparison, known KH-PRF from LWE or Ring-LWE [BLMR13, BP14, Kim20] are limited to small coefficients.

Construction	Assumption	KH	Weak/Strong	Eval. Depth
[YS16, Theorem 4.4]	LPN	✗	strong	$O(\log^{1+\epsilon} n)$
[BCG+20]	VD-LPN	✗	weak	$\underline{O(\log n)}$
Construction 1	LPN	$\underline{\checkmark}$	weak	$\underline{O(\log n)}$
Construction 2	sparse LPN	$\underline{\checkmark}$	strong	$O(\log^{1+\epsilon} n)$ $\underline{O(\log n)}$ with **EI**
Construction 3	seeded LPN	$\underline{\checkmark}$	strong	$\underline{O(\log n)}$

Table 1: Comparison of the PRF constructions in this work and the existing PRF constructions based on LPN (or its variants). The noise level are all polynomially low, i.e., Bernoulli noise with $\mu = n^{-c}$ for $c \in (0, 1)$. **KH** stands for key homomorphism (Definition 12). **EI** stands for encoded-input (Definition 4).

The SPN Paradigm. As an intriguing coincidence, all our constructions fit within the SPN paradigm which is used in modern block cipher design (e.g., AES). Specifically, our constructions can be specified by a tuple of algorithms $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{S\text{-}Box}, \mathsf{Linear})$ and the evaluation is depicted in Figure 1. Apart from having an SPN-like structure, our PRFs have additional features that align with common practice in modern block cipher design. For instance, as the round number increases, our PRFs become *provably* more secure.

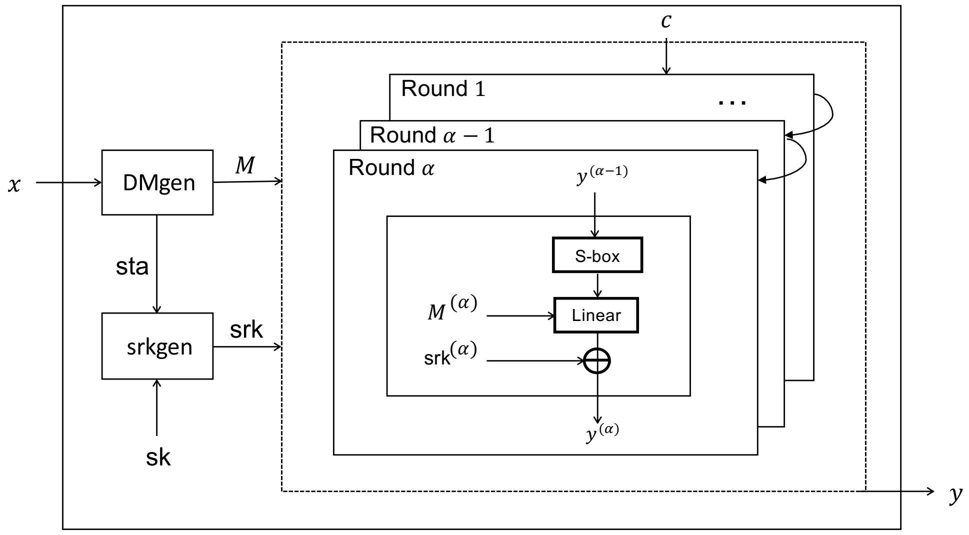

Figure 1: Structure of our PRFs, which align with the SPN paradigm.

1.2 Related Work

Constructions of low-depth PRF is a fundamental question with a rich history of study. In this section, we provide a brief literature review of constructions of low-depth PRFs, beyond what has already been talked about.

Yu and Steinberger [YS16] constructed PRFs in $\mathsf{AC}^{0+\epsilon} \subseteq \mathsf{NC}^{1+\epsilon} \subseteq \mathsf{NC}^2$ from learning parity with noise assumption (LPN) with polynomially low Bernoulli noise (same assumption as that used in our Construction 1). Concretely, they first constructed a PRG in AC^0 from LPN and then apply the GGM transformation to yield the result. This is the best result one can hope for if we use the GGM transformation in a black-box manner to construct a PRF.

The idea of building (weak) PRFs via "LPN with deterministic noise" has its origins in [ABG+14]. It was also explicitly discussed in [BIP+18] (their alternative weak PRFs can be written as $F_s(x) = \langle x, s\rangle + [\langle x, s\rangle \mod 3] \mod 2$, which is an LPN with a "deterministic noise" of rate 1/3) and formalized in [BCG+21], who coined the term "LPN with simple noise" and provided some preliminary analysis of its security against linear tests. Our second approach to obtaining strong PRFs from our weak PRFs is also reminiscent of the hash-then-evaluate paradigm. The possibility of instantiating in the standard model for various weak PRFs, including VDLPN, was recently studied in [BCE+24].

Another line of work focuses on constructing low depth PRFs from non-standard assumptions. To name a few, Miles and Viola [MV12] constructed PRFs candidates in TC^0. [ABG+14] obtained weak PRFs in $\mathsf{AC}^0 \circ \mathsf{MOD}_2$ from a new hard learning problem. [BIP+18] constructed weak PRFs in ACC^0. Applebaum and Raykov [AR16] gave a PRF AC^0 based on a variant of Goldreich's low-locality one-way function [Gol00]. [BCG+20] (see also [CD23]) constructed weak PRFs in $\mathsf{AC}^0[\mathsf{MOD}_2]$ from a so-called variable-density LPN (VD-LPN). [BCG+21] constructed weak PRFs in $\mathsf{AC}^0 \circ \mathsf{MOD}_2$.

2 Technical Overview

Noisy linear algebraic assumptions, such as the Learning Parity with Noise (LPN) assumption [BFKL94] and Learning with Errors (LWE) assumption [Reg05], postulate that samples of the form $\langle \boldsymbol{a}, \boldsymbol{s} \rangle + \boldsymbol{e}$, where $\boldsymbol{a}, \boldsymbol{s}$ are random and $\boldsymbol{e}$ is a small/sparse error, are pseudorandom. This structure makes them appealing for designing pseudorandom functions (PRFs). Indeed, at least in the case of building a weak PRF, one can imagine treating $\boldsymbol{a}$ as the input and $\boldsymbol{s}$ as the key, and computing the inner product $\langle \boldsymbol{a}, \boldsymbol{s} \rangle$. However, the challenge is in finding a way to deterministically introduce sufficiently independent error terms for $\langle \boldsymbol{a}_i, \boldsymbol{s} \rangle$ when queried on different $\boldsymbol{a}_i$.

A quick review of derandomization for LWE samples For LWE-based PRF constructions, there is an elegant method to generate these errors deterministically, known as Learning with Rounding (LWR) [BPR12, AKPW13]. Instead of adding a random small error to LWE samples $\langle \boldsymbol{a}_i, \boldsymbol{s} \rangle \in \mathbb{Z}_q$ so as to hide their exact value, LWR releases a deterministically rounded version of $\langle \boldsymbol{a}_i, \boldsymbol{s} \rangle$, i.e., $\lfloor \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle \rceil_p \in \mathbb{Z}_p$ where $\lfloor x \rceil_p \overset{\text{def}}{=} \lfloor (p/q) \cdot x \mod q \rceil \bmod p$. Geometrically, for some $p < q$, we partition the elements of $\mathbb{Z}_q$ into p contiguous intervals, each containing approximately q/p elements. Then, the rounding function $\lfloor \cdot \rceil_p \colon \mathbb{Z}_q \to \mathbb{Z}_p$ assigns $x \in \mathbb{Z}_q$ to the corresponding interval index. It was proved in [BPR12] that as long as q/p is super-polynomially large in the security parameter, LWR is as hard as LWE. The main idea is that $\lfloor \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle \rceil_p = \lfloor \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + \boldsymbol{e} \rceil_p$ except with negligible probability over random $\boldsymbol{a}_i \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{e} \leftarrow \chi$, where χ is the error distribution.

Such "rounding" technique to derandomize LWE samples yields constructions of new PRFs, including PRFs in TC^0 [BPR12] from ring-LWR (or equivalently, ring-LWE [LPR10] if q/p is super-polynomially large), and almost key-homomorphic PRFs [BLMR13, BP14, Kim20]. Notice that because they only obtain almost key-homomorphism, namely, that there is small additive error introduced with every key-homomorphic operation, one can only support an a priori bounded number of key combinations before the noise becomes too large.

Trying to "port" the rounding function to the LPN setting breaks down completely since the error is no longer small rather sparse. To the best of our knowledge, there is no known method to derandomize LPN samples in an analogous manner. Devising such a method is the key technical contribution of the current work, and the new conceptual contribution. As a consequence of our LWR-style derandomization technique for LPN samples, we get new constructions of low-depth and key-homomorphic PRFs from (variant of) the LPN assumption.

In what follows, we present the key ideas underlying our constructions. We start by first describing a construction of bounded-query key-homomorphic weak PRF. We then explain how one can turn it into an (unbounded-query) key-homomorphic weak PRF, though it is not yet depth-efficient. We then introduce our recursive derandomization technique that allows us to obtain logarithmic-depth PRF (while preserving key-homomorphism).

A bounded-query key-homomorphic weak PRF. Consider the following weak PRF construction $F\colon \mathbb{F}_2^{(\ell+1)\times n} \times \mathbb{F}_2^n \to \mathbb{F}_2$.

- Parameter: LPN parameter (n, μ), where $\mu = 2^{-\ell}$ for some $\ell \in \mathbb{Z}$.
- Secret key: Uniformly random vectors $\boldsymbol{s}, \boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_\ell \in \mathbb{F}_2^n$.
- Evaluation: On input vector $\boldsymbol{a} \in \mathbb{F}_2^n$, do:
  - Compute $\boldsymbol{r} = (\langle \boldsymbol{a}, \boldsymbol{s}'_1 \rangle, \ldots, \langle \boldsymbol{a}, \boldsymbol{s}'_\ell \rangle) \in \mathbb{F}_2^\ell$.
  - Use these ℓ bits as seed to sample from Ber_μ, where $\mu = 2^{-\ell}$. This can be done by computing $\widetilde{e} = S(\boldsymbol{r})$, where $S(\boldsymbol{r})$ is defined as $S(\boldsymbol{r}) = \prod_{i=1}^{\ell} r_i \in \mathbb{F}_2$.
  - Output $\langle \boldsymbol{a}, \boldsymbol{s} \rangle + \widetilde{e} \in \mathbb{F}_2$.

By the LPN assumption, it is straightforward to show that this construction is a weak PRF for an *a priori bounded number of* queries. The reason is that $\boldsymbol{r}$ (and therefore the Bernoulli error $\widetilde{e}$ derived from it) cannot be argued to be pseudorandom after $\widetilde{\Omega}(n\ell)$ different input queries; indeed, it is extracted from secret key, that has only $O(n\ell)$ bits of entropy.

We will turn the above PRF into an unbounded-query weak PRF below, but for now observe that the PRF template we provided has a linear structure that natively supports key-homomorphism. The same high level structure for key-homomorphism is common to all of our constructions. Fix two secret keys $\boldsymbol{S}_1 = (\boldsymbol{s}_1, \boldsymbol{s}'_{1,1}, \ldots, \boldsymbol{s}'_{1,\ell}), \boldsymbol{S}_2 = (\boldsymbol{s}_2, \boldsymbol{s}'_{2,1}, \ldots, \boldsymbol{s}'_{2,\ell})$ and an input $x = \boldsymbol{a}$. It holds that $F(\boldsymbol{S}_1 + \boldsymbol{S}_2, x) = \langle \boldsymbol{a}, \boldsymbol{s}_1 + \boldsymbol{s}_2 \rangle + \widetilde{e}_3$ and $F(\boldsymbol{S}_1, x) + F(\boldsymbol{S}_2, x) = \langle \boldsymbol{a}, \boldsymbol{s}_1 + \boldsymbol{s}_2 \rangle + \widetilde{e}_1 + \widetilde{e}_2$ for error terms $\widetilde{e}_1, \widetilde{e}_2, \widetilde{e}_3$ computed as in the construction. Assume for now that $\boldsymbol{S}_1, \boldsymbol{S}_2$, and x are chosen uniformly at random. Then $\widetilde{e}_1 + \widetilde{e}_2$ is "close" (in some appropriate sense) to $\widetilde{e}_3$ and so key-homomorphism holds for random keys. To support key-homomorphism for arbitrary keys, we enforce sparsity of errors retrospectively. Specifically, we repeat PRF n times, and when we evaluate the PRF with input x, and secret key $(\boldsymbol{S}^{(1)}, \ldots, \boldsymbol{S}^{(n)})$, we check whether $(\widetilde{e}^{(1)}, \ldots . \widetilde{e}^{(n)})$ is a sparse vector; if not, we simply set $(\widetilde{e}^{(1)}, \ldots . \widetilde{e}^{(n)})$ to $\mathbf{0}$. This will not break PRF security since it will happen only with negligible probability over a random key.

A key-homomorphic weak PRF. The main idea to get a provably secure unbounded-query weak PRF is to have the entries of $\boldsymbol{r}$ be themselves noisy. That is, we will now have $\boldsymbol{r} = (\langle \boldsymbol{a}, \boldsymbol{s}'_1 \rangle + e_1, \ldots, \langle \boldsymbol{a}, \boldsymbol{s}'_\ell \rangle + e_\ell)$, where the e_i's are (pseudo)random Bernoulli noise terms. Indeed, if we assume that the e_i's are freshly-sampled Bernoulli noise terms, then by a hybrid argument one can base the security of the modified weak PRF from above on LPN (preserving key-homomorphism).

However, since we are building a PRF, the e_i's need to be deterministically generated. A trivial solution would be to use a PRF F' to generate them! Specifically, add a PRF key k to the secret key of our new PRF and let $(e_1, \ldots, e_\ell) = S(F'(k, x))$, where x is the input of the weak PRF F. By instantiating F' with any LPN-based PRF construction, say [YS16], we obtain a key-homomorphic weak PRF from LPN.

Getting logarithmic depth. So far we have ignored depth efficiency altogether. In particular, since the PRF from above uses another PRF as a black box within its construction, it cannot be more depth efficient than previously known PRFs. We avoid invoking a generic PRF to sample the noise terms by using a recursive technique. Instead of using another PRF to pseudorandomly generate the noise terms, we generate the noise terms by recursively invoking the bounded-query weak PRF, hoping that sufficiently many recursive invocations will suffice for security to hold.

To make the idea more precise, we now describe our scheme; see Figure 2 for an illustration. We then analyze efficiency and provide intuition as to why the recursion depth that we use (denoted τ below) suffices to prove security.

- Parameter: LPN parameter (n, μ), where $\mu = 2^{-\ell}$ for some $\ell \in \mathbb{Z}$. Recursion depth τ.
- Secret key: Uniformly random vectors $\boldsymbol{s}, (\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_\ell), \ldots, (\boldsymbol{s}'''_1, \ldots, \boldsymbol{s}'''_{\ell^\tau}) \in \mathbb{F}_2^n$.
- Evaluation: On input vectors $\boldsymbol{a} \in \mathbb{F}_2^n$, do:
  - For $\alpha = \tau, \tau-1, \ldots, 2$ in sequential:
    1. If $\alpha = \tau$, compute $\boldsymbol{r}_\alpha = (\langle \boldsymbol{a}, \boldsymbol{s}'''_1 \rangle, \ldots, \langle \boldsymbol{a}, \boldsymbol{s}'''_{\ell^\tau} \rangle) \in \mathbb{F}_2^{\ell^\tau}$
    2. If $\alpha < \tau$, compute $\boldsymbol{r}_\alpha = (\langle \boldsymbol{a}, \boldsymbol{s}''_1 \rangle + e_1, \ldots, \langle \boldsymbol{a}, \boldsymbol{s}''_{\ell^\alpha} \rangle + e_{\ell^\alpha}) \in \mathbb{F}_2^{\ell^\alpha}$
    3. Compute $(e_1, \ldots, e_{\ell^{\alpha-1}}) = S(\boldsymbol{r}_\alpha) \in \mathbb{F}_2^{\ell^{\alpha-1}}$, where S is extended by applying it on each ℓ-bit block separately.
  - Output $\langle \boldsymbol{a}, \boldsymbol{s} \rangle + e_1 \in \mathbb{F}_2$.

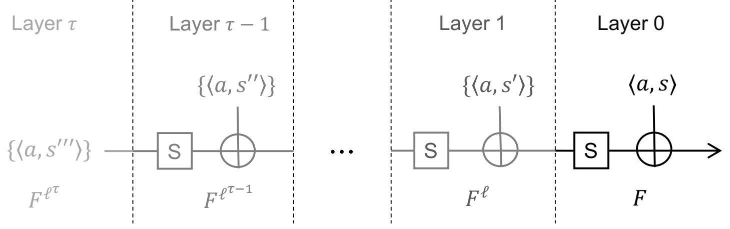

Figure 2: An illustration of weak PRF construction. Color indicates its influence on the output of PRF. Lighter color indicate lower sensitivity, i.e., the probability of the event that perturbing its output value (even arbitrarily) leaves the PRF output unchanged. The intuition is that we can replace the base layer by a truly random value without changing the PRF output.

In words, the output of the weak PRF F is $\langle \boldsymbol{a}, \boldsymbol{s} \rangle + e \in \mathbb{F}_2$, where $e \in \mathbb{F}_2$ is extracted from the preceding layer of the form $\{\langle \boldsymbol{a}, \boldsymbol{s}'_i \rangle + e'_i\}_{i\in[\ell]} \in \mathbb{F}_2^\ell$, where $\{e'_i\}_{i\in[\ell]} \in \mathbb{F}_2^\ell$ is again extracted from its preceding layer of the form $\{\langle \boldsymbol{a}, \boldsymbol{s}''_i \rangle + e''_i\}_{i\in[\ell]} \in \mathbb{F}_2^{\ell^2}$, and so on until we recurse to extract errors from the base layer, which is of the vanilla form $\{\langle \boldsymbol{a}, \boldsymbol{s}'''_i \rangle\}_{i\in[\ell^\tau]} \in \mathbb{F}_2^{\ell^\tau}$.

Efficiency: Each inner product of the form $\langle \boldsymbol{a}, \boldsymbol{s} \rangle$ can be computed in logarithmic depth. The main observation is that all of these inner products can be computed in parallel in depth $O(\log n)$. Thus, as long as $\tau = O((\log n)/\log \ell)$, we can evaluate the function in logarithmic depth.

Security: The crucial observation underlying our security proof is that the output remains almost unchanged even if in layer τ we replace $\{\langle \boldsymbol{a}, \boldsymbol{s}'''_i \rangle\}$ by the output of a random function R.[4]

[4]Recall that this phenomenon is of the same flavor as the "rounding" function used to derandomize LWE samples, wherein $\lfloor \langle \boldsymbol{a}, \boldsymbol{s} \rangle \rceil_p$ remains almost unchanged if we replace it by $\lfloor \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e \rceil_p$ for a small error e. It is also inspired by Kim's chaining idea [Kim20] that was introduced in the context of LWE-based PRFs (with the goal of getting constructions with polynomial modulus).

To this end, we define a "hybrid" PRF construction $\tilde{F}$ that will help us in the security proof. $\tilde{F}$ is defined identical to F except that it sets $\boldsymbol{r}_\tau = R(x)$ where x is input of $\tilde{F}$. By a hybrid argument, one can show that assuming that LPN is hard, $\tilde{F}$ a secure weak PRF. In more detail, it holds that

$$R \text{ is random function} \overset{\text{LPN}}{\Longrightarrow} \boldsymbol{r}_{\alpha-1} \text{ is (pseudo)random} \Longrightarrow \boldsymbol{e} = S(\boldsymbol{r}_{\alpha-1}) \sim \mathsf{Ber}_\mu$$
$$\overset{\text{LPN}}{\Longrightarrow} \boldsymbol{r}_{\alpha-2} \text{ is (pseudo)random} \Longrightarrow \overbrace{\cdots\cdots\cdots}^{\text{repeat } \tau \text{ times}} \overset{\text{LPN}}{\Longrightarrow} y \text{ is (pseudo)random}$$

Finally, we need to argue that any efficient adversary cannot distinguish F from $\tilde{F}$. To this end, we show an even stronger (information theoretic) claim: except with negligible probability, F and $\tilde{F}$ have the same input-output behavior. That is,

$$\Pr_{x \leftarrow \mathbb{F}_2^n, \boldsymbol{S} \leftarrow \mathcal{K}} \left[F(\boldsymbol{S}, x) \neq \tilde{F}(\boldsymbol{S}, x) \right] = O\left((\mu \ell^\tau)^\tau \right) = \mathsf{negl}(n). \tag{1}$$

We now provide some intuition as to why (1) is true. As depicted in Figure 2, the base layer (layer τ) has diminishing impacts on PRF output. This is because each time the function S is applied, it is highly likely to output 0 (as long as its input is not an all-one vector, which happens with probability at most $\mu\ell^\tau$). Consequently, the PRF output becomes almost independent of the values in the base layer (except in the unlikely case where bad events occur simultaneously across all τ layers, which happens with probability at most $O\left((\mu\ell^\tau)^\tau\right)$). Setting $\mu = n^{-c}$ (then $\ell = -\log \mu = O(\log n)$), and $\tau = O(\log n / \log \ell)$ is sufficient to make this probability super-polynomially low while enabling the function to be computable in logarithmic depth.

Getting a strong PRF. So far, the random coefficient $\boldsymbol{A}$ within the LPN sample $\langle \boldsymbol{s}, \boldsymbol{A} \rangle$ is derived from the input of the weak PRF which is chosen at random. In a standard PRF, however, the inputs could be arbitrary and we must devise a way to generate these coefficients in low depth and that enable a security proof under LPN (or variants). We propose two pathways: The first relies on sparse LPN and results with an input-encoded PRF in NC^1, and the second relies on a new assumption, called seeded LPN, and results in a (standard) PRF in NC^1. For now, we only consider an input domain of slightly super-polynomial size, i.e., $\{0,1\}^{\omega(\log n)}$. One can extend the input domain to exponentially large, albeit at the cost of losing key-homomorphism or losing adaptive security.

Approach 1: Leveraging sparse LPN. Our first approach leverages sparse LPN. Recall that sparse LPN is a variant of LPN where $\mathcal{C}(n,m)$ outputs a random matrix $\boldsymbol{A} \in \mathbb{F}_2^{n\times m}$ such that each column is t-sparse for a parameter $t = n^{o(1)}$. We do not know whether sparse LPN is hard assuming LPN although there has been recent progress on this from [JLS24, BBTV24]. Nevertheless, sparse LPN has been studied and used for over a couple of decades in cryptography and average-case complexity [Fei02, Ale03b, AIK08, IKOS08, ABW10, AOW15, KMOW17, JLS21, JLS22, DIJL23, RVV24, CGHKV24].

Our PRF construction takes inspiration from [BLMR13]. For sake of illustration, assume that the input x consists of only two bits; it will be natural to extend the ideas to a larger input domain. The authors of [BLMR13] constructed a PRF from LWE that has the form $f_{\boldsymbol{s}}^{\text{BLMR}}(x_1 \| x_2) = \left\lfloor \boldsymbol{s} \cdot \boldsymbol{A}_{x_1} \boldsymbol{A}_{x_2} \right\rceil_p$, where the $\boldsymbol{A}$ matrices are low-norm[5] (i.e., matrices in $\{0,1\}^{n\times n}$). The fact that the $\boldsymbol{A}$'s are small norm

[5]LWE with small norm $\boldsymbol{A}$ is hard assuming LWE [BLMR13].

is crucial for rounding operation as we need to make sure that $\lfloor \boldsymbol{s} \cdot \boldsymbol{A}_{x_1} \boldsymbol{A}_{x_2} \rceil_p = \lfloor (\boldsymbol{s} \cdot \boldsymbol{A}_{x_1} + \boldsymbol{e}') \boldsymbol{A}_{x_2} + \boldsymbol{e}'' \rceil_p$ with high probability for randomly chosen small errors $\boldsymbol{e}', \boldsymbol{e}''$.

Our goal is to apply a similar ideas in LPN land. Useful facts are that the product of sparse matrices $\boldsymbol{A}_{x_1} \boldsymbol{A}_{x_2}$ is still sparse, and if $\boldsymbol{e}'$ and $\boldsymbol{e}''$ are Bernoulli distributed, $\boldsymbol{e}' \boldsymbol{A}_{x_2} + \boldsymbol{e}''$ remains sparse. As before, we start by constructing a bounded-query strong PRF F, as we informally describe next.

- Parameter: Random t-sparse matrices $\boldsymbol{A}_0, \boldsymbol{A}_1 \in \mathbb{F}_2^{n \times n}$, $\boldsymbol{A}'_0, \boldsymbol{A}'_1 \in \mathbb{F}_2^{n \times n\ell}$, where $\mu = 2^{-\ell}$. Output length M ($M = 1$ for now).
- Secret key: Uniformly random vectors $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3 \in \mathbb{F}_2^n$.
- Evaluation: On input $x = x_1 x_2 \in \{0,1\}^2$, do:
  - Compute $\boldsymbol{r}_{x_1} = \boldsymbol{s}_2 \cdot \boldsymbol{A}'_{x_1}$ and $\boldsymbol{r}_{x_1 x_2} = \boldsymbol{s}_3 \cdot \boldsymbol{A}'_{x_1} \boldsymbol{A}'_{x_2}$.
  - Compute $\widetilde{\boldsymbol{e}}_{x_1} = S(\boldsymbol{r}_{x_1}) \in \mathbb{F}_2^n$ and $\widetilde{\boldsymbol{e}}_{x_1 x_2} = S(\boldsymbol{r}_{x_1 x_2}) \in \mathbb{F}_2^n$.
  - Compute $\boldsymbol{y} = (\boldsymbol{s}_1 \cdot \boldsymbol{A}_{x_1} + \widetilde{\boldsymbol{e}}_{x_1}) \cdot \boldsymbol{A}_{x_2} + \widetilde{\boldsymbol{e}}_{x_1 x_2} = \boldsymbol{s}_1 \cdot \boldsymbol{A}_{x_1} \boldsymbol{A}_{x_2} + \widetilde{\boldsymbol{e}}_{x_1} \cdot \boldsymbol{A}_{x_2} + \widetilde{\boldsymbol{e}}_{x_1 x_2} \in \mathbb{F}_2^n$.
  - Output the first bit of $\boldsymbol{y}$, i.e., $y_1 \in \mathbb{F}_2$.

Using the similar argument as before, if we have $\boldsymbol{r}_{x_1} = \boldsymbol{s}\boldsymbol{A}'_{x_1} + \boldsymbol{e}$, and $\boldsymbol{r}_{x_1,x_2} = (\boldsymbol{s} \cdot \boldsymbol{A}'_{x_1} + \boldsymbol{e}') \cdot \boldsymbol{A}'_{x_2} + \boldsymbol{e}''$ for some random Bernoulli error $\boldsymbol{e}, \boldsymbol{e}', \boldsymbol{e}''$, then it will become a secure PRF from (sparse) LPN assumption. We can then recursively invoke this PRF to provide randomness to sample Bernoulli error whenever we need. Similar to Figure 2, we now have the following recursion structure:

$$F^{\beta^\tau} \to F^{\beta^{\tau-1}} \to \cdots \to F^{\beta} \to F,$$

where β is the blowup factor[6] to be determined. While these are the main ideas, there are several technical issues that come up. We highlight one such issue in the box below and refer to the technical section for the full details. On a first pass, this can be skipped.

In each recursion layer, we will go through the following sub-procedure of F:

- **Goal**: Sample $\boldsymbol{e} \sim \mathsf{Ber}_\mu^n$ from $\boldsymbol{r}$. Compute $\boldsymbol{e}\boldsymbol{A} \in \mathbb{F}_2^n$. Output the first M bits of $\boldsymbol{e}\boldsymbol{A}$.

Naively, computating $\boldsymbol{e}\boldsymbol{A}$ requires $\log n$ depth due to the vector-matrix multiplication, where each is of dimension n. Thus, in order to be able to have $\tau = \omega(1)$ number of recursion layers while keeping the total evaluation depth $O(\log n)$, we must make the evaluation depth of each recursion layer $o(\log n)$.

Let us first consider the case $M = 1$. Notice that the first bit of $\boldsymbol{e}\boldsymbol{A}$ is equal to $\boldsymbol{e} \cdot \boldsymbol{a}$ where $\boldsymbol{a} \in \mathbb{F}_2^n$ is the first column of $\boldsymbol{A}$ and has Hamming weight t. Thus, we observe that one can "simulate" the first bit of $\boldsymbol{e}\boldsymbol{A}$ by sampling only t Bernoulli errors, and computing their summation in depth $O(\log t)$. Specifically,

[6]In the previous weak PRF (Figure 2), $\beta = \ell$. Namely, β is defined to be $\frac{\#\boldsymbol{e} \cdot \ell}{M}$, where $\#\boldsymbol{e}$ denotes how many Bernoulli bits F consumes, and M is the output length of F (i.e., how many random bits F produces). In words, β is how many times F needs to invoke itself to provide the required Bernoulli errors. The goal is to decrease β by increasing M and decreasing $\#\boldsymbol{e}$.

- **Simulating a 1-bit output:** Sample $\boldsymbol{e}' \sim \mathsf{Ber}_\mu^t$ from $\boldsymbol{r}$. Output the sum of entries of $\boldsymbol{e}'$.

We now explain how we "simulate" the first M bits of $\boldsymbol{eA}$ in depth $o(\log n)$. Unfortunately, we cannot simulate M outputs by independently repeating the above "1-bit simulation" process M times, since different entries of $\boldsymbol{eA}$ might not be independent due to the overlap in the support of columns of $\boldsymbol{A}$. Instead, we observe that the above "1-bit simulation" can be alternatively viewed as follows: pick the first column of $\boldsymbol{A}$, delete all zeros to obtain $\boldsymbol{a}' = (1, \ldots, 1) \in \mathbb{F}_2^t$, and output $\boldsymbol{e}' \cdot \boldsymbol{a}'$. We obtain the following method:

- **Simulating an M-bit output:** Sample $\boldsymbol{e}' \sim \mathsf{Ber}_\mu^{t \cdot M}$ from $\boldsymbol{r}$. Output $\boldsymbol{e}'\boldsymbol{A}' \in \mathbb{F}_2^M$, where $\boldsymbol{A}' \in \mathbb{F}_2^{tM \times M}$ is devised from $\boldsymbol{A} \in \mathbb{F}_2^{n \times n}$ as follows. Pick the first M columns of $\boldsymbol{A}$ to obtain a matrix in $\mathbb{F}_2^{n \times M}$, and then delete all zero rows to obtain $\boldsymbol{A}' \in \mathbb{F}_2^{tM \times M}$.

Computing $\boldsymbol{e}'\boldsymbol{A}'$ can be done in depth $O(\log(tM)) = o(\log n)$ (assuming $tM = o(n)$).

Approach 2: PRF in NC^1 from Seeded LPN. LPN is truly unique in that it allows for making assumptions with respect to structured (and sometimes even deterministic matrices $\boldsymbol{A}$). For LWE type assumptions if the matrix $\boldsymbol{A} \in \mathbb{Z}_p^{n \times m}$ is wide-enough for $m \gg n \log p$, then, there would always exist short (in fact Boolean) non-zero vectors $\boldsymbol{x}$ so that $\boldsymbol{Ax} = 0$ regardless of the distribution of $\boldsymbol{A}$. This makes it a bit tricky to make structured LWE assumptions as the security implicitly depends on the hardness of being able to find such short-vectors.

For LPN, one could consider making structured assumptions for even deterministically chosen matrices $\boldsymbol{A}$, provided their dual distance is large (i.e., there does not exist very sparse $\boldsymbol{x}$ such that $\boldsymbol{Ax} = 0$). The reason is that there are no (known) natural trapdoors for LPN type problems. The best known generic attack on an LPN variants typically amount to finding sparse $\boldsymbol{x}$ so that $\boldsymbol{Ax} = 0$. This gives rise to a linear test of the form $(\boldsymbol{sA} + \boldsymbol{e})\boldsymbol{x} = \boldsymbol{ex}$. It then extracts a biased bit $\boldsymbol{ex}$, which may be biased with inverse-polynomial bias if the Bernoulli error probability $\mu < \frac{O(\log n)}{d}$ where d is the dual-distance of $\boldsymbol{A}$. If one keeps the error probability at least $\frac{\omega(\log n)}{d}$, then such linear tests do not yield a valid efficient attack. This makes LPN variants interesting even in a scenario where the adversary is allowed to compute arbitrary preprocessing on the matrix (provided it has large dual distance).

Linear tests are thought to be optimal for most distributions (barring a few counter-examples that are derived from algebraically structured codes such as Reed-Solomon code). In recent years, we have seen a lot of interesting applications from such structured assumptions in cryptography, especially in context of efficient MPC (e.g. [BCG+20, CRR21, BCG+22, BCCD23]), where the security of these new assumptions is typically justified by proving resilience to linear tests.

In this work, we propose a very natural LPN variant, which we call *seeded LPN*, that immediately gives rise to a key-homomorphic PRF in NC^1. We observe that the bottleneck of our first approach is computing the coefficient matrix $\boldsymbol{A}_x = \prod_i \boldsymbol{A}_{x_i}$. Here we come up with a matrix distribution $\boldsymbol{A}$ where each column can be generated in NC^1, and the matrix $\boldsymbol{A}$ has a linear dual distance, therefore resisting linear attacks.

In our seeded LPN assumption, columns of the public matrix $\boldsymbol{A}$ follows certain n-wise independent distribution. Specifically, let n, m be integers, $p > 2^n$. Let H be an n-wise independent hash function mapping from $[m]$ to $\mathbb{Z}_p$. Let $\mathsf{bit\text{-}decompose}$ be the bit decomposition function that takes on input in $\mathbb{Z}_p$ and outputs $\mathbb{F}_2^n$. Then for $i \in [m]$, the i-th column of $\boldsymbol{A}$ is $\mathsf{bit\text{-}decompose}(H(i)) \in \mathbb{F}_2^n$.

For secret key $\boldsymbol{S}$, and input $x \in [m]$, the strong PRF is defined as

$$F(\boldsymbol{S}, x) = F'(\boldsymbol{S}, \mathsf{bit\text{-}decompose}(H(x))),$$

where F' is the weak PRF in logarithmic depth constructed before. Security is proven under seeded LPN assumption (F' is used in a non-black box manner). A key component in the proof is using the n-wise independence property to argue large dual distance. See details in Section 8. In terms of efficiency, since H, $\mathsf{bit\text{-}decompose}$, and F' can all be computed in depth of $O(\log n)$, we obtain the first PRF construction from LPN-style assumptions in NC^1.

Extension to LPN over Rings. All of our constructions naturally extend to LPN over Rings. Over large rings, our blowup factor is even smaller, allowing for more recursive layers which might be useful for arguing exponential security. Ring version could also be useful for designing efficient block ciphers. We leave coming up with such optimizations to future work. We refer to Appendix I for details.

3 Preliminaries

We use $[n]$ to denote set $\{1, \ldots, n\}$. For a probability distribution X over a domain D, let X^n denote its n-fold product distribution over D^n. The uniform distribution over a finite domain D is denoted by $U(D)$. We use Ber_μ to denote the Bernoulli distribution with parameter μ, i.e., $\Pr[\mathsf{Ber}_\mu = 1] = \mu$, $\Pr[\mathsf{Ber}_\mu = 0] = 1 - \mu$. Extending it over larger rings, we use $\mathsf{Ber}_\mu(\mathbb{Z}_p)$ to denote the Bernoulli distribution that returns 0 with probability $1 - \mu$, and a uniformly random non-zero element of $\mathbb{Z}_p$ otherwise. For a binary vector $\boldsymbol{x}$, we use $|\boldsymbol{x}|$ or $\mathsf{wt}(\boldsymbol{x})$ denotes the Hamming weight of $\boldsymbol{x}$. We denote the Hamming distance between two binary vectors x and y by $\Delta(x, y)$. For $n \in \mathbb{N}$, U_n denotes the uniform distribution over $\mathbb{F}_2^n$. $X \sim D$ denotes that random variable X follows distribution D. For a distribution S, we use $s \leftarrow S$ to denote sampling an element s according to distribution S. For a finite set S, we use $s \leftarrow S$ to denote sampling s uniformly from finite set S. For a value x, we use $y \leftarrow x$ to denote assigning the value of x to y. We use $\mathrm{Funcs}[\mathcal{X}, \mathcal{Y}]$ to denote the set off all functions mapping $\mathcal{X}$ to $\mathcal{Y}$. We adopt the following vector convention: when we left-multiply a vector $\boldsymbol{a}$, $\boldsymbol{a}$ is treated as a row vector; when we right multiply a vector $\boldsymbol{a}$, it is treated as a column vector.

We use the following simplified notation to ease presentation. The security parameter through this paper is n, and all algorithms (including the adversary) are implicitly given the security parameter n in unary, and most other parameters are functions of n. We omit n when it is clear from the context. For example, $\mu = \mu(n) \in (0, 1/2)$, $q = q(n) \in \mathbb{N}$, and $m = m(n) \in \mathsf{poly}(m)$, where poly refers to the set of all polynomial function. We say that a function $\epsilon = \epsilon(n)$ is negligible if for sufficiently large n, it is smaller than $1/n^c$ for any constant c.

We consider adversaries interacting as part of probabilistic experiments called games. For an adversary $\mathcal{A}$ and two games $H_0(n)$, $H_1(n)$ with which it can interact, $\mathcal{A}$'s distinguishing advantage (as a function of n) is $\mathsf{Adv}_{H_0,H_1}(\mathcal{A}) := |\Pr[\mathcal{A} \text{ accepts in } H_0] - \Pr[\mathcal{A} \text{ accepts in } H_1]|$. When we say that an algorithm is *efficient*, we mean that it is a probabilistic polynomial-time algorithm.

Definition 1 (Computational Indistinguishability). We say that games H_0, H_1 are computationally indistinguishable, and denote $H_0 \overset{c}{\approx} H_1$, if for any probabilistic polynomial-time $\mathcal{A}$, there exists a negligible function $\epsilon(n)$ such that $\mathsf{Adv}_{H_0,H_1}(\mathcal{A}) \leq \epsilon(n)$.

3.1 Pseudorandom Functions

In this section, we formally define pseudorandom functions [GGM86], encoded-input PRFs [BIP+18], and key-homomorphic PRFs [NPR99, BLMR13].

Definition 2 (Pseudorandom Functions (PRFs) [GGM86]). A *pseudorandom function* for an efficiently sampleable key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is an efficiently computable deterministic function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for any efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left| \Pr_{k \leftarrow \mathcal{K}} \left[\mathcal{A}^{F(k,\cdot)}(1^n) = 1 \right] - \Pr_{f \leftarrow \mathrm{Funcs}[\mathcal{X},\mathcal{Y}]} \left[\mathcal{A}^{f(\cdot)}(1^n) = 1 \right] \right| = \mathsf{negl}(n).$$

If $\mathcal{A}$ is limited to be non-adaptive, (i.e., it has to write all his oracle calls before making the first call), then F is called non-adaptive PRF.

We generally refer to the experiment where the adversary $\mathcal{A}$ is given oracle access to the real PRF $F(k,\cdot)$ as the *real* PRF experiment. Analogously, we refer to the experiment where the adversary $\mathcal{A}$ is given oracle access to a truly random function $f(\cdot)$ as the *ideal* PRF experiment.

In this work, we will mainly construct PRFs for input domain $\mathcal{X} = \{0,1\}^{\omega(\log n)}$ of slightly super-polynomial size. One can transform them to support input domain $\mathcal{X} = 2^{\mathsf{poly}(n)}$ via domain extension techniques [Lev85, BHKN13, DS15].

Lemma 1 (Domain Extension [Lev85, BHKN13, DS15]). *If there exists a PRF $F\colon \mathcal{K} \times \{0,1\}^{\omega(\log n)} \to \mathcal{Y}$ computable in depth d, then there exists a PRF $F'\colon \mathcal{K} \times \{0,1\}^{\mathsf{poly}(n)} \to \mathcal{Y}$ computable in depth $d + O(\log n)$.*

Weak PRFs are non-adaptive PRFs that satisfy security as above assuming that queries are done only on uniformly random values. For every function f and every sequence $X = [x_1, x_2, \ldots, x_k]$ of values in the domain of f, denote by $V(X, f)$ the sequence $[x_1, f(x_1), x_2, f(x_2), \ldots, x_k, f(x_k)]$ (V stands for values).

Definition 3 (Weak PRFs). A *weak pseudorandom function* for an efficiently sampleable key space $\mathcal{K}$, domain $\mathcal{X}$, and range $\mathcal{Y}$ is an efficiently computable deterministic function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for any efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any polynomial $q = q(n)$,

$$\left| \Pr_{\substack{k \leftarrow \mathcal{K} \\ X \leftarrow \mathcal{X}^q}} [\mathcal{A}(1^n, V(X, F(k,\cdot))) = 1] - \Pr_{\substack{f \leftarrow \mathrm{Funcs}[\mathcal{X},\mathcal{Y}] \\ X \leftarrow \mathcal{X}^q}} [\mathcal{A}(1^n, V(X, f)) = 1] \right| = \mathsf{negl}(n).$$

Encoded-input PRFs (EI-PRFs) [BIP+18] are PRFs where the input domain is no longer $\{0,1\}^n$, but an efficiently sampleable and recognizable set.

Definition 4 (Encoded-input PRFs [BIP$^+$18]). Let $E\colon \mathcal{X} \to \mathcal{X}'$ be an efficient encoding function. An *encoded-input pseudorandom function* for an efficiently sampleable key space $\mathcal{K}$, input domain $\mathcal{X}$, and range $\mathcal{Y}$ is an efficiently computable deterministic function $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that for any efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left| \Pr_{k \leftarrow \mathcal{K}} \left[\mathcal{A}^{F(k,E(\cdot))}(1^n, E) = 1 \right] - \Pr_{f \leftarrow \mathrm{Funcs}[\mathcal{X},\mathcal{Y}]} \left[\mathcal{A}^{f(\cdot)}(1^n, E) = 1 \right] \right| = \mathsf{negl}(n).$$

Moreover, we say that F is computable in NC^1 if $F(k, x')$ is computable in NC^1 for $x' = E(x)$.

Obviously, an EI-PRF can be viewed as a PRF by simply absorbing the encoding function E into its implementation. This, however, increases its complexity, proportionally to the complexity of evaluating E.

As discussed in [BIP$^+$18], EI-PRFs have applications in symmetric cryptography applications that require computationally efficient PRFs. Indeed, $E(x)$ can be made public without compromising the pseudorandomness of $F(k, E(x))$ and therefore decryption does not require re-evaluation of E on x. This is useful in many applications such as CCA-secure encryption, MACs, and more.

Key-homomorphism. Key-homomorphic PRFs are a special family of PRFs that satisfy an additional algebraic property. Specifically, for a key-homomorphic PRF, the key space $\mathcal{K}$ and the range $\mathcal{Y}$ of the PRF exhibit certain group structures such that its evaluation on any fixed input $x \in \mathcal{X}$ is homomorphic with respect to these group structures. We first define a linear operator, followed by the definition of key-homomorphic PRFs.

Definition 5 (Linear Operator). Let $(\mathbb{G}, \oplus), (\mathbb{G}', +)$ be two groups. Let m be an integer. We say that $T\colon \mathbb{G} \to \mathbb{G}'$ is a linear operator if for any $x_1, x_2 \in \mathbb{G}$, and any integer $\lambda_1, \lambda_2 \in \mathbb{Z}$, it holds that

$$T(\lambda_1 \cdot x_1 \oplus \lambda_2 \cdot x_2) = \lambda_1 \cdot T(x_1) + \lambda_2 \cdot T(x_2). \tag{2}$$

Definition 6 (Key-Homomorphic PRFs [NPR99, BLMR13]). Let $(\mathcal{K}, \oplus)$, $(\mathcal{Y}, +)$ be groups. Let $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PRF (Definition 2). F is a key-homomorphic if for every $x \in \mathcal{X}$, $F(\cdot, x)\colon \mathcal{K} \to \mathcal{Y}$ is a linear operator (Definition 5).

Key-homomorphic PRFs have many useful applications in symmetric cryptography and give rise to distributed PRFs, symmetric-key proxy re-encryption, and updatable encryption [BLMR13, EPRS17, LT18, KLR19, BDGJ20, BEKS20], OT extension [Sch18], and single-server secure aggregation [LLPT23]. To date, KH-PRFs (Definition 6) are only known to exist in the random oracle model. In the plain model, we only know how to achieve a relaxation of key-homomorphism called almost-key-homomorphism [BLMR13].

Definition 7 (Almost Linear Operator). Let $(\mathbb{G}, \oplus)$ be a group. Let $\gamma \in (0, 1)$. Let q be an integer. We say $T\colon \mathbb{G} \to \mathbb{Z}_q$ is a γ-almost linear operator if it satisfies

- **(Approximate Additivity)** For any $x_1, x_2 \in \mathbb{G}$, it holds that

$$T(x_1 \oplus x_2) = T(x_1) + T(x_2) + \gamma q \tag{3}$$

Definition 8 (**Almost-Key-Homomorphic PRFs [BLMR13]**)**.** Let $(\mathcal{K}, \oplus)$, $(\mathcal{Y}, +)$ be groups. Let $F\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PRF (Definition 2). F is a γ-almost-key-homomorphic if for every $x \in \mathcal{X}$, $F(\cdot, x)\colon \mathcal{K} \to \mathcal{Y}$ is a γ-almost-linear operator (Definition 7).

Almost-key-homomorphic PRFs are sufficient for all applications of key-homomorphic PRFs mentioned above. Specifically, to get rid of small approximate error, one first multiplies the message that is to be masked by the output of the PRF by some factor c. Then, as long as $\gamma q < c/2$, one can recover the message. To date, almost-key-homomorphic PRFs are known from (ring-)LWE/LWR assumptions [BLMR13, BP14, Kim20].

3.2 Substitution-Permutation Networks

The SPN paradigm (dating back to Shannon [Sha49]) captures structural properties of secure ciphers, and is used to construct modern block ciphers (e.g., AES). Specifically, an SPN is computed over a number of rounds, where each round "confuses" the input by dividing it into bundles and applying a substitution function (S-box) to each bundle, "diffuses" the bundles by applying a matrix with certain "branching" properties, and then performs "key mixing" by XORing with the private round key.

3.3 The LPN and Sparse LPN Assumptions

Here, we recall the LPN [BFKL94] and sparse LPN [Fei02, Ale03a] assumptions.

Definition 9 (**LPN**)**.** Let n be a security parameter. Let $m = \mathsf{poly}(n)$, and let $\mu = n^{-c}$ for $c \in (0, 1)$. The $\mathsf{LPN}_{n,\mu,m}$ assumption states that

$$(\boldsymbol{A}, \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e}) \overset{c}{\approx} (\boldsymbol{A}, U_m), \tag{4}$$

where $\boldsymbol{A} \leftarrow \mathbb{F}_2^{n\times m}$, $\boldsymbol{s} \leftarrow \mathbb{F}_2^n$, and $\boldsymbol{e} \leftarrow \mathsf{Ber}_\mu^m$.

The sparse LPN assumption, denoted $\mathsf{sparseLPN}$, is a natural variant of the LPN assumption, where each column of the public matrix is t-sparse for a parameter t. First introduced in [Ale03a], that used it for obtaining hardness of approximation results, variants of the $\mathsf{sparseLPN}$ assumption were subsequently used for constructing local pseudorandom generators [AIK08], cryptography with constant computational overhead [IKOS08], public-key encryption schemes [ABW10], pseudorandom correlation generators [BCGI18], multi-party homomorphic secret sharing [DIJL23], and more.

We consider $\mathsf{sparseLPN}$ over binary field $\mathbb{F}_2$. This regime is well-studied and believed to be hard [Fei02, Ale03a, AOW15, KMOW17]. In fact, for our sparse LPN-based PRF constructions, it suffices to set a slightly sub-polynomial sparsity parameter $t = n^{o(1)}$. Therefore, our assumption is more conservative than previous works using $t = O(1)$. Formally, the assumption is stated as follows.

Definition 10 (**Sparse LPN**)**.** Let n be a security parameter. Let $m = \mathsf{poly}(n)$, and let $\mu = n^{-c}$ for $c \in (0, 1)$. Let $t = \omega(1)$ be the sparsity parameter. The $\mathsf{sparseLPN}_{n,\mu,t,m}$ assumption states that

$$(\boldsymbol{A}, \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e}) \overset{c}{\approx} (\boldsymbol{A}, U_m), \tag{5}$$

where $\boldsymbol{s} \leftarrow \mathbb{F}_2^n$, $\boldsymbol{e} \leftarrow \mathsf{Ber}_\mu^m$, and $\boldsymbol{A} \leftarrow \mathbb{F}_2^{n\times m}$ conditioned on every column of $\boldsymbol{A}$ having exactly t non-zero elements.

One benefit of LPN and sparse LPN is plausible post-quantum security: they appear to resist attacks even from quantum algorithms [BLVW19, DJ24]. Another benefit of basing PRFs on LPN (over $\mathbb{F}_2$) is that, besides asymptotic efficiency, it is plausible that low-depth LPN-based PRFs will be practically favorable since multiplications and additions are simply AND and XOR operations.

Extending to multiple secrets. We extend LPN (resp. sparse LPN) assumption to M secrets. Let $\boldsymbol{A} \in \mathbb{F}_2^{n\times m}$ be the public matrix, $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_M) \leftarrow \mathbb{F}_2^{M\times n}$ be M secrets. The LPN (resp. sparse LPN) instance becomes $(\boldsymbol{A}, \boldsymbol{S}\cdot\boldsymbol{A}+\boldsymbol{E})$, where $\boldsymbol{E} \leftarrow \mathsf{Ber}_\mu^{M\times m}$. By a standard hybrid argument, distinguishing samples of LPN (resp. sparse LPN) with M secrets from uniformly random $(\boldsymbol{A}, U_{M\times m})$ is as hard as the original LPN (resp. sparse LPN) assumption, up to a multiplicative factor of M.

4 Approximate Key-Homomorphic PRFs

4.1 Definition

We introduce the notion of *approximate* key-homomorphic PRFs, which is stronger than *almost* homomorphic PRFs introduced in [BLMR13]. We first define an approximate linear operator as follows.

Definition 11 (Approximate Linear Operator). Let $\gamma \in (0,1)$ and $m \in \mathbb{N}$. Let $(\mathbb{G}, +)$ be a group. We say $T\colon \mathbb{G} \to \mathbb{F}_2^m$ is a γ-approximate linear operator if it satisfies

1. (**Approximate Additivity**) For any $x_1, x_2 \in \mathbb{G}$, it holds that
$$T(x_1 + x_2) = T(x_1) + T(x_2) + \boldsymbol{\epsilon},$$
where $\boldsymbol{\epsilon} \in \mathbb{F}_2^m$ is a sparse vector such that $\mathsf{wt}(\boldsymbol{\epsilon}) \le \gamma m$.
2. (**Approximate Homogeneity**) For any $x \in \mathbb{G}$ and any integer $\lambda \in Z$, it holds that
$$T(\lambda \cdot x) = \lambda \cdot T(x) + \boldsymbol{\epsilon},$$
where $\boldsymbol{\epsilon} \in \mathbb{F}_2^m$ is a sparse vector such that $\mathsf{wt}(\boldsymbol{\epsilon}) \le \gamma m$.

Equivalently, we say $T\colon \mathbb{G} \to \mathbb{F}_2^m$ is a γ-approximate linear operator if for any $x_1, x_2 \in \mathbb{G}$, and any $\lambda_1, \lambda_2 \in \mathbb{Z}$, it holds that
$$T(\lambda_1 \cdot x_1 + \lambda_2 \cdot x_2) = \lambda_1 \cdot T(x_1) + \lambda_2 \cdot T(x_2) + \boldsymbol{\epsilon},$$
where $\boldsymbol{\epsilon} \in \mathbb{F}_2^m$ is a sparse vector such that $\mathsf{wt}(\boldsymbol{\epsilon}) \le \gamma m$.

Remark 2. Approximate linear operator satisfies both two requirements of linear operator (Definition 5) approximately, whereas the notion of almost linear operator (Definition 7) underlying almost KH-PRFs (Definition 8) only satisfies additivity approximately. Thus, approximate key-homomorphic PRFs, as defined below, are strictly stronger than "almost key-homomorphic PRFs" (as defined and constructed in [BLMR13, BP14, Kim20]).

Definition 12 (Approximate Key-Homomorphic PRFs). Let $\gamma \in (0,1)$ and $m \in \mathbb{N}$. Let $(\mathcal{K}, \oplus)$ be a group. We say that a PRF (Definition 2) $F: \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2^m$ is a γ-approximate key-homomorphic PRF (γ-AKH-PRF) if for every $x \in \mathcal{X}$, $F(\cdot, x): \mathcal{K} \to \mathbb{F}_2^m$ is a γ-approximate linear operator (Definition 11). The same definition extends to non-adaptive PRFs (Definition 2), weak PRFs (Definition 3), and encoded-input PRFs (Definition 4).

Remark 3. From this point forward, when we refer to "key-homomorphic PRFs (KH-PRFs)", we actually mean "approximate key-homomorphic PRFs (AKH-PRFs)".

4.2 From PRFs with AKH structure to AKH-PRFs

Definition 13 (PRFs with AKH structure). Let $\gamma \in (0, 1/2)$. Let $(\mathcal{K}, \oplus)$ be a group. Let $F: \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2$ be a PRF. We say F has γ-AKH structure if there exists a tuple (f, h, z) such that for any $k \in \mathcal{K}, x \in \mathcal{X}$, F can be represented as

$$F(k,x) = \langle f(x), h(k) \rangle + z(k,x), \tag{6}$$

where

- $f: \mathcal{X} \to \mathbb{F}_2$ is some function.
- $h: \mathcal{K} \to \mathbb{F}_2^m$ is a group homomorphism.
- $z: \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2$ is a function satisfying:
  1. **(Efficiency)** For any $x \in \mathcal{X}, k \in \mathcal{K}$, $z(k,x)$ can be computed as efficiently as $F(k,x)$.
  2. **(Vanishing)** For any $x \in \mathcal{X}$, $\Pr_{k \leftarrow \mathcal{K}}[z(k,x) = 1] \leq \gamma$. (In the context of a weak PRF, we require $\Pr_{k \leftarrow \mathcal{K}, x \leftarrow \mathcal{X}}[z(k,x) = 1] \leq \gamma$.),

In the following theorem, we show that having an AKH structure can be used to turn any such PRF into an AKH-PRF.

Theorem 4 (AKH structure implies AKH-PRFs). *Let $n, m \in \mathbb{N}$. If there exists a PRF $F: \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2$ with γ-AKH structure, then there exists a 6γ-AKH-PRF $F': \mathcal{K}^m \times \mathcal{X} \to \mathbb{F}_2^m$ such that for any PRF adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}_{F'}(\mathcal{A}, n, m) \leq m \cdot \mathsf{Adv}_F(\mathcal{A}) + \exp\left(\log|\mathcal{X}| - \frac{m\gamma}{3}\right).$$

Moreover, if F can be computed in depth d, then F' can be computed in depth $2d + \log m$.

Main idea. We consider a "repeated" (direct product) version of the PRF to get m bits of output. Each time we evaluate the PRF with input x and secret key $(k_1, \dots, k_n)$. Then we want to show that $(z_1(k_1, x), \dots z_n(k_n, x))$ is a sparse vector, which in fact (only) holds for random keys by the vanishing property[7]. Our solution is to check whether $(z_1(k_1, x), \dots z_n(k_n, x))$ is a sparse vector. (This does not change the asymptotic complexity of the PRF due to the efficiency property of z.) If it is not a sparse vector, we enforce sparsity by simply setting it to zero. This bad event will not happen (except with negligible probability) in the pseudorandomness game of the PRF (where the key is chosen at random) and therefore does not break security.

Proof of Theorem 4. Assume that $F \colon \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2$ is a PRF with γ-AKH structure. We shall construct a 6γ-AKH-PRF $F' \colon \mathcal{K}^m \times \mathcal{X} \to \mathbb{F}_2^m$. First, we consider $F^m \colon \mathcal{K}^m \times \mathcal{X} \to \mathbb{F}_2^m$, which is an m-repetition PRF of F with m different keys. More precisely, on input $x \in \mathcal{X}$, $F^m_{\boldsymbol{k}}(x)$ is defined as

1. Parse $\boldsymbol{k} = (k_1, k_2, \dots, k_m)$, where $k_i \in \mathcal{K}$ for $i \in [m]$.
2. Compute $y_i \leftarrow F(k_i, x) \in \mathbb{F}_2$ for $i \in [m]$ in parallel.
3. Set $\boldsymbol{y} \leftarrow (y_1, \dots, y_m) \in \mathbb{F}_2^m$.
4. Output $\boldsymbol{y} \in \mathbb{F}_2^m$.

We define $z^{m\prime} \colon \mathcal{K}^m \times \mathcal{X} \to \mathbb{F}_2^m$ analogously to F^m. We now define our AKH-PRF $F' \colon \mathcal{K}^m \times \mathcal{X} \to \mathbb{F}_2^m$ based on $F^m \colon \mathcal{K}^m \times \mathcal{X} \to \mathbb{F}_2^m$ plus an additional sanity check on the sparsity of z^m. More precisely, on input $x \in \mathcal{X}$, $F'_{\boldsymbol{k}}(x)$ is defined as

1. Compute $\boldsymbol{y} \leftarrow F^m(\boldsymbol{k}, x) \in \mathbb{F}_2^m$.
2. Compute $\boldsymbol{z} \leftarrow z^m(\boldsymbol{k}, x) \in \mathbb{F}_2^m$.
3. Set $\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{z} \cdot \mathbb{I}[\mathsf{wt}(\boldsymbol{z}) > 2m\mu] \in \mathbb{F}_2^m$, where $\mathbb{I}[\mathsf{wt}(\boldsymbol{z}) > 2m\mu]$ equals 1 if and only if $\mathsf{wt}(\boldsymbol{z}) > 2m\mu$; otherwise equals 0.
4. Output $\boldsymbol{y} \in \mathbb{F}_2^m$.

Efficiency. Suppose that F can be evaluated in depth d. The evaluation depth of F' is $2d + \log m$. This is because F^m has the same evaluation depth as F, i.e., d, and z^m has the same evaluation depth as z, which is most d by definition of AKH structure (Definition 13). The additional depth of $\log m$ comes from computing $\mathsf{wt}(\boldsymbol{z})$ for $\boldsymbol{z} \in \mathbb{F}_2^m$.

Approximate key homomorphism. For any $x \in \mathcal{X}$, $\boldsymbol{k}_1, \boldsymbol{k}_2 \in \mathcal{K}$, denote $\boldsymbol{z}_1 = \boldsymbol{z}(k_1, x) \cdot \mathbb{I}[\mathsf{wt}(\boldsymbol{z}(k_1, x)) < 2m\mu]$. Define $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$ analogously with respect to $\boldsymbol{k}_2$, $\boldsymbol{k}_1 + \boldsymbol{k}_2$, respectively. It

[7]Note that $(k_1, \dots, k_n)$ is not necessarily random, and can be arbitrary, since linear property (Definition 11) should hold for every keys in the domain.

holds that for any $x \in \mathcal{X}$, $\boldsymbol{k}_1, \boldsymbol{k}_2 \in \mathcal{K}$, $\lambda_1, \lambda_2 \in \mathbb{Z}$,

$$\begin{aligned} & F'(\lambda_1 \cdot \boldsymbol{k}_1 + \lambda_1 \cdot \boldsymbol{k}_2, x) \\ =& \langle f(x), h(\lambda_1 \cdot \boldsymbol{k}_1 + \lambda_1 \cdot \boldsymbol{k}_2)\rangle + \boldsymbol{z}_3 \\ =& \langle f(x), \lambda_1 \cdot h(\boldsymbol{k}_1) + \lambda_2 \cdot h(\boldsymbol{k}_2)\rangle + \boldsymbol{z}_3 \\ =& \lambda_1 \cdot \langle f(x), h(\boldsymbol{k}_1)\rangle + \lambda_2 \cdot \langle f(x), h(\boldsymbol{k}_2)\rangle + \boldsymbol{z}_3 \\ =& \lambda_1 \cdot (\langle f(x), h(\boldsymbol{k}_1)\rangle + \boldsymbol{z}_1) + \lambda_2 \cdot (\langle f(x), h(\boldsymbol{k}_2)\rangle + \boldsymbol{z}_2) + (\lambda_1 \boldsymbol{z}_1 + \lambda_2 \boldsymbol{z}_2 + \boldsymbol{z}_3) \\ =& \lambda_1 F'(\boldsymbol{k}_1, x) + \lambda_2 F'(\boldsymbol{k}_2, x) + (\lambda_1 \boldsymbol{z}_1 + \lambda_2 \boldsymbol{z}_2 + \boldsymbol{z}_3), \end{aligned} \tag{7}$$

where the first equality is due to definition of F', the second equality is due to group homomorphism h. Since $\mathsf{wt}(\boldsymbol{z}_i) \le 2\gamma m$ for $i = 1, 2, 3$, and $\mathsf{wt}(\lambda_i \boldsymbol{z}_i) = \mathsf{wt}(\boldsymbol{z}_i)$, we have $\mathsf{wt}(\lambda_1 \boldsymbol{z}_1 + \lambda_2 \boldsymbol{z}_2 + \boldsymbol{z}_3) \le 6\gamma \cdot m$. Therefore, there exists $\boldsymbol{\eta} \in \mathbb{F}_2^m$ where $\mathsf{wt}(\boldsymbol{\eta}) \le 6\gamma \cdot m$, such that $F'(\lambda_1 \cdot \boldsymbol{k}_1 + \lambda_2 \cdot \boldsymbol{k}_2, x) = \lambda_1 \cdot F'(\boldsymbol{k}_1, x) + \lambda_2 \cdot F'(\boldsymbol{k}_2, x) + \boldsymbol{\eta}$, as desired.

Security. By a standard hybrid argument, F^m is as secure as F (up to a multiplicative factor of m in distinguishing advantage). It then suffices to prove the following lemma.

Lemma 2. *For any adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}_{F'}(\mathcal{A}, n, m) \le \mathsf{Adv}_{F^m}(\mathcal{A}) + \exp\left(n - \frac{m\mu}{3}\right).$$

Proof. Without loss of generality, we assume that $\mathcal{A}$ is deterministic. If $F'(\boldsymbol{k}, \cdot)$ and $F^m(\boldsymbol{k}, \cdot)$ have the same input-output except with some small probability (over the randomness of secret key $\boldsymbol{k} \leftarrow \mathcal{K}$), then the distinguishing advantage of $\mathcal{A}$ is information-theoretically upper bounded by that probability.

Claim 1. *For any $x \in \mathcal{X}$, $\Pr_{\boldsymbol{k} \leftarrow \mathcal{K}^m}[F'(\boldsymbol{k}, x) \ne F^m(\boldsymbol{k}, x)] \le \exp\left(-\frac{\gamma m}{3}\right)$.*

Proof. $F'(\boldsymbol{k}, x) \ne F^n(\boldsymbol{k}, x)$ if and only if $\mathsf{wt}(z^m(\boldsymbol{k}, x)) > 2\gamma m$. Let $z_i = z(k_i, x) \in \mathbb{Z}$ for $i \in [m]$. Then $\mathsf{wt}(z^m(\boldsymbol{k}, x)) = z_1 + \ldots + z_m$. It holds that

$$\mathbb{E}_{(k_1,\ldots,k_m) \leftarrow \mathcal{K}^m}[z_1 + \ldots + z_m] = \sum_{i=1}^{m} \mathbb{E}_{k_i \leftarrow \mathcal{K}}[z_i] \le \gamma m, \tag{8}$$

where the inequality is by the degradation property of z. It then follows by Chernoff's bound (Lemma 12) that

$$\Pr\left[\mathsf{wt}(z^m(\boldsymbol{k}, x)) > 2\gamma m\right] \le \Pr\left[\mathsf{wt}(z^m(\boldsymbol{k}, x)) > 2\mathbb{E}[\mathsf{wt}(z^m(\boldsymbol{k}, x))]\right] \le \exp\left(-\frac{m\gamma}{3}\right).$$

□

By a union bound over all $x \in \mathcal{X}$, it follows that $F'(\boldsymbol{k}, \cdot)$ has exactly the same input-output behavior as $F^m(\boldsymbol{k}, \cdot)$ except with probability at most $\exp\left(\log|\mathcal{X}| - \frac{m\gamma}{3}\right)$, which concludes the proof of the lemma. □

This concludes the proof of the theorem. □

5 Bernoulli Sample Function over GF(2)

In this section, we introduce a central gadget in our constructions, the Bernoulli sample function $\mathsf{S\text{-}Ber}$. We first give its definition and then prove its key properties.

Definition 14 (**Bernoulli Sample Function over** $\mathbb{F}_2$)**.** Let $\ell \in \mathbb{N}$ and set $\mu = 2^{-\ell}$. For an input $x \in \mathbb{F}_2^\ell$, the Bernoulli sample function $\mathsf{S\text{-}Ber} = \mathsf{S\text{-}Ber}_\mu : \mathbb{F}_2^\ell \to \mathbb{F}_2$ is defined as follows.

$$\mathsf{S\text{-}Ber}(x) = \prod_{i=1}^{\ell} x_i. \tag{9}$$

We abuse notation and extend $\mathsf{S\text{-}Ber}$ to handle inputs $x \in \mathbb{F}_2^{m\ell}$ by applying $\mathsf{S\text{-}Ber}$ separately on each ℓ-bit block, yielding an output in $\mathbb{F}_2^m$.

An important feature of the above function is that it is highly parallelizable: it can be computed by a circuit of depth $\log \ell$ containing constant fan-in AND gates, independently of the input length.

Claim 2. *Fix $\ell, m \in \mathbb{N}$ and $\mu \in (0,1)$. The function $\mathsf{S\text{-}Ber}_\mu(\cdot)$ operating on inputs from $\mathbb{F}_2^{m\ell}$ can be computed by a circuit of depth $O(\log \ell)$, where $\ell = -\log \mu$. In particular, for $\mu = n^{-c}$ and $c \in (0,1)$, $\mathsf{S\text{-}Ber}_\mu(\cdot)$ is computable in depth $O(\log \log n)$.*

Claim 3. *It holds that $\mathsf{S\text{-}Ber}_\mu(\boldsymbol{u}) \sim \mathsf{Ber}_\mu^m$ for $\boldsymbol{u} \leftarrow \mathbb{F}_2^{m\ell}$.*

The next property of the Bernoulli sample function, which we call derandomization effect, says the following. If the input x of $\mathsf{S\text{-}Ber}$ is perturbed by arbitrary value (independent of x), its output will remain unchanged except with small probability.

Lemma 3 (**Derandomization Effect**)**.** *Let $\ell, m \in \mathbb{N}$ and let $\mu = 2^{-\ell}$. Consider $\boldsymbol{x} \leftarrow \mathbb{F}_2^{m\ell}$, where $\boldsymbol{x} = (\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(m)}), \boldsymbol{x}_{(i)} \in \mathbb{F}_2^\ell$ for $i \in [m]$. For any random variable $(\boldsymbol{e}, \boldsymbol{e}')$ over $\mathbb{F}_2^{m\ell} \times \mathbb{F}_2^{m\ell}$ that is independent of $\boldsymbol{x}$, it holds that*

$$\Pr_{\boldsymbol{x},\boldsymbol{e},\boldsymbol{e}'}[\mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}) = \mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}')] \geq (1-2\mu)^m. \tag{10}$$

Proof. By law of total probability,

$$\begin{aligned}
&\Pr_{\boldsymbol{x},\boldsymbol{e},\boldsymbol{e}'}[\mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}) = \mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}')] \\
&= \sum_{\boldsymbol{v} \in \mathbb{F}_2^m} \Pr_{\boldsymbol{x},\boldsymbol{e},\boldsymbol{e}'}[\mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}) = \boldsymbol{v} \wedge \mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}') = \boldsymbol{v}] \\
&\geq \Pr_{\boldsymbol{x},\boldsymbol{e},\boldsymbol{e}'}[\mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}) = \boldsymbol{0} \wedge \mathsf{S\text{-}Ber}(\boldsymbol{x}+\boldsymbol{e}') = \boldsymbol{0}],
\end{aligned}$$

where $\boldsymbol{0} = (0, \ldots, 0) \in \mathbb{F}_2^{m\ell}$. Considering chunks of length ℓ separately, we obtain

$$= \Pr_{\boldsymbol{x},\boldsymbol{e},\boldsymbol{e}'}[\boldsymbol{x}_{(i)} \neq \boldsymbol{e}_{(i)} \wedge \boldsymbol{x}_{(i)} \neq \boldsymbol{e}'_{(i)}, i \in [m]].$$

Since $(\boldsymbol{e}, \boldsymbol{e}')$ is independent of $\boldsymbol{x}$,

$$= \sum_{e,e' \in \mathbb{F}_2^{m\ell}} \Pr_{\boldsymbol{e},\boldsymbol{e}'}[(\boldsymbol{e},\boldsymbol{e}') = (e,e')] \cdot \Pr_{\boldsymbol{x}}[\boldsymbol{x}_{(i)} \neq e_{(i)} \wedge \boldsymbol{x}_{(i)} \neq e'_{(i)}, i \in [m]]$$

Since the $\boldsymbol{x}_{(1)}, \ldots, \boldsymbol{x}_{(n)}$ are independently and identically distributed as uniform random variables,

$$
\begin{aligned}
&= \sum_{e,e' \in \mathbb{F}_2^{m\ell}} \Pr_{\boldsymbol{e},\boldsymbol{e}'}[(\boldsymbol{e}, \boldsymbol{e}') = (e, e')] \cdot \prod_{i=1}^{m} \Pr\left[\boldsymbol{x}_{(1)} \neq e_{(1)} \wedge \boldsymbol{x}_{(1)} \neq e'_{(1)}\right] \\
&\geq \sum_{e,e' \in \mathbb{F}_2^{m\ell}} \Pr_{\boldsymbol{e},\boldsymbol{e}'}[(\boldsymbol{e}, \boldsymbol{e}') = (e, e')] \cdot \Big((1 - 2 \cdot 2^{-\ell})\Big)^m \\
&= (1 - 2\mu)^m.
\end{aligned}
$$

□

Remark 4. The above derandomization has the same flavor as the "rounding" function (LWR) used to derandomize LWE samples, wherein the probability of $\lfloor x \rceil_p = \lfloor x + e \rceil_p$ is larger over $x \leftarrow \mathbb{Z}_q$, and a small error $e \in [B]$ where $B < q$. We note two key differences:

1. In LWR, the error e must be small (e.g., $B/q = 1/\mathsf{poly}(n)$), whereas, here $\boldsymbol{e}, \boldsymbol{e}'$ can follow an arbitrary distribution (not necessarily sparse).
2. In LWR, the derandomization effect still holds even if the error e depends on x. However, here, we require the error to be independent of $\boldsymbol{x}$.

Remark 5. Jumping ahead, the Bernoulli sampling function $\mathsf{S\text{-}Ber}$ plays the role of the S-box in our PRF construction when viewed through the lens of the SPN paradigm. Recall that in AES, the S-box is defined as S-box$(x) = x^{-1}$ over the extended finite field, and is designed to be invertible to ensure that the block cipher realizes a (pseudorandom) permutation. In contrast, since our goal is to construct a pseudorandom function (which is not necessarily a permutation), we are not constrained to use an invertible S-box—similar to modern hash functions such as SHA3, which employ non-invertible nonlinear components. Conceptually, the S-box in an SPN serves to introduce nonlinearity and confusion. Our function $\mathsf{S\text{-}Ber}$ fulfills this role in the simplest possible way: by computing the product of input bits. Remarkably, this minimal form of nonlinearity suffices to achieve pseudorandomness in our constructions.

6 Weak PRFs in NC^1 from LPN

6.1 Construction

In this section, we present our construction of weak PRFs.

Construction 1 (Weak PRFs from LPN). The weak PRF function is defined with respect to the following public parameters pp:

1. LPN parameter (n, μ), where $\mu = 2^{-\ell}$ for $\ell \in \mathbb{N}$.
2. Recursion depth $\tau \in \mathbb{N}$.

Denote $M_\alpha = \ell^{\tau-\alpha}$ for $0 \le \alpha \le \tau$, and $M = M_0 = \ell^\tau$. A member of the function family

$$\mathcal{F} = \mathcal{F}_{n,\mu,\tau} = \{F_{\boldsymbol{S}} : \{0,1\}^n \to \{0,1\}, \boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}\}. \tag{11}$$

is indexed by the secret key

$$\boldsymbol{S} = (\boldsymbol{s}_{(i)}^{(\alpha)})_{\alpha \in [\tau], i \in [M]} \in \mathbb{F}_2^{M\tau \times n}.$$

On input $x \in \{0,1\}^n$, $F_{\boldsymbol{S}}$ operates as follows:

1. $\boldsymbol{a} \leftarrow x \in \mathbb{F}_2^n$.
2. $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}_{(1)}^{(\alpha)}, \dots, \boldsymbol{s}_{(M_\alpha)}^{(\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$ for $\alpha \in [\tau]$ *in parallel.*
3. $\boldsymbol{r}^{(\alpha)} \leftarrow \boldsymbol{S}^{(\alpha)} \cdot \boldsymbol{a} \in \mathbb{F}_2^{M_\alpha}$ for $\alpha \in [\tau]$ *in parallel.*
4. $\boldsymbol{y}^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{F}_2^M$.
5. For $\alpha = 1, \dots, \tau$ *sequentially*:
   (a) $\boldsymbol{e}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$ (Definition 14).
   (b) $\boldsymbol{y}^{(\alpha)} \leftarrow \boldsymbol{r}^{(\alpha)} + \boldsymbol{e}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.
6. Output $y \leftarrow y^{(\tau)} \in \mathbb{F}_2$.

Theorem 5 (Security). *Let n be a security parameter. Let $\mathcal{F} = \mathcal{F}_{n,\mu,\tau}$ be the function family defined in Construction 1. Let $\ell = -\log \mu$, $M = \ell^\tau$. Then, for any efficient weak PRF adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, Q) \le M\tau \cdot \mathsf{Adv}_{\mathsf{LPN}}(n, \mu, Q) + Q \cdot \left((\mu M)^\tau + 2^{-(n-M\tau-1)}\right).$$

In particular, assuming LPN (Definition 9) is hard, and setting

$$\begin{cases} \mu = n^{-c} \text{ for any } c \in (0,1) \\ \tau = \omega(1) \cap O(\log n / \log\log n) \end{cases}$$

imply that $\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, Q) = \mathsf{negl}(n)$.

Proof sketch: We first show that constructed PRF F will have the same input-output behavior as an ideal version $\tilde{F}$ (where we replace $\boldsymbol{y}^{(0)}$ with a truly random value) except with negligible probability. We give a more detailed proof sketch below.

Consider the execution of $F(\boldsymbol{S}, x)$ and $\tilde{F}(\boldsymbol{S}, x)$ for $\boldsymbol{S} \leftarrow \mathbb{F}_2^{(\tau+1)\times n}, x \leftarrow \mathbb{F}_2^{m \times n}$. For $\alpha \in [\tau]$, denote by GOOD_α the event that the Bernoulli error of the α-th layer, which is extracted from the $(\alpha+1)$-th layer via

$$\boldsymbol{e}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha},$$

is identical across F and F'. Obviously, $F(\boldsymbol{S}, x) \neq \tilde{F}(\boldsymbol{S}, x)$ if and only if $\neg\mathsf{GOOD}_0$. Observe that $\mathsf{GOOD}_\tau, \mathsf{GOOD}_{\tau-1}, \dots, \mathsf{GOOD}_0$ is a monotone condition; once it is satisfied, it remains so for all future iterations. Together, $F(\boldsymbol{S}, x) \neq \tilde{F}(\boldsymbol{S}, x)$ if and only if $\neg\mathsf{GOOD}_\tau \wedge \dots \wedge \neg\mathsf{GOOD}_0$.

We claim that $\neg\mathsf{GOOD}_\tau \wedge \neg\mathsf{GOOD}_{\tau-1} \wedge \ldots \wedge \neg\mathsf{GOOD}_0$ with negligible probability $O(p^\tau)$, where $p = (1-2^{-\ell})^{2M_\alpha}$. Indeed, for $\alpha \in [\tau]$, the event GOOD_α occurs if the output of $\mathsf{S\text{-}Ber}$ at the α-th layer is a zero vector in both F and F'. Since $\mathsf{S\text{-}Ber}$ is a highly biased function, where $\Pr_{x \leftarrow \mathbb{F}_2^{M_{\alpha+1}}}[\mathsf{S\text{-}Ber}(x) = 0^{M_\alpha}] = (1-2^{-\ell})^{M_\alpha}$, we can prove that GOOD_α occurs with probability p.

The security of $\tilde{F}$ can be proven from the LPN assumption via a hybrid argument on round iteration. We refer to Appendix C for the full proof.

Remark 6. (Concrete security) In terms of the concrete security (T, Q, ϵ) of PRFs (where any adversary making at most Q queries and running in time T has an advantage of at most ϵ), our weak PRFs are only *provably secure* under (slightly) superpolymial T, Q, and $1/\epsilon$. Note that this is true even if we assume subexponential hardness for LPN problem. However, we are not aware of any quasipolynomial-time attacks that break the security of our weak PRFs.

Theorem 6 **(AKH structure).** *Under the parameter setting in Theorem 5, Construction 1 has γ-AKH structure (Definition 13) where $\gamma = \mu + \mathsf{negl}(n)$.*

Proof sketch: Observe that the output of the PRF takes the form $\langle \boldsymbol{s}^{(\tau)}, \boldsymbol{a}_x \rangle + e^{(\tau)}$. We need to show that $e^{(\tau)}$ is zero except with probability $\mu + \mathsf{negl}(n)$ over random key and random input. This follows as a byproduct of the security proof of Theorem 5. We refer to Appendix D for the full proof.

Claim 4 (**Efficiency**). *Under the parameter setting in Theorem 5, Construction 1 can be computed by a circuit of depth $O(\log n)$.*

Proof. Step 2 has depth $O(\log n)$. Since $\mathsf{S\text{-}Ber}$ is computable in $O(\log\log n)$ by Claim 2, Step 5 has depth $O(\tau \log\log n) = O(\log n)$. Other steps are all constant depth. Therefore, the overall evaluation depth of Construction 1 is $O(\log n)$. $\square$

Finally, by combining Theorem 5 (security), Theorem 6 (AKH structure), Claim 4 (efficiency), and Theorem 4 (AKH structure implies AKH-PRFs), we reach the following conclusion.

Corollary 1. *Let n be a security parameter and let $\mu = n^{-c}$ for any constant $c \in (0,1)$. Assume $\mathsf{LPN}_{n,\mu,m}$ (Definition 9) is hard for any m polynomial in n. Then, there exists a key-homomorphic weak PRF (Definition 12) $F \colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(n)}$ that can be computed by a circuit in NC^1.*

6.2 Our Construction as an SPN

As discussed in the introduction, our construction nicely fit into the substitution-permutation network (SPN) design framework used in modern block ciphers. In this section, we present Construction 1 from the perspective of SPN. We instantiate the SPN template by first specifying format of its inputs and outputs, followed by the specification of $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{Linear}, \mathsf{S\text{-}Box})$.

On input $x \in \{0,1\}^n$, the weak PRF $F_{\boldsymbol{S}}$ with secret key $\boldsymbol{S} \leftarrow \mathbb{F}_2^{M_\tau \times n}$ is defined as:

1. **Key schedule:**

(a) Derive diffusion matrix for round $\alpha \in [\tau]$ in parallel:

$$(\mathsf{sta}^{(\alpha)}, D^{(\alpha)}) \leftarrow \mathsf{DMgen}(x, \alpha),$$

where $\mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^n$, and $D^{(\alpha)} \in \mathbb{F}_2^{M_{\alpha-1} \times M_{\alpha-1}}$.

(b) Derive private round key for round $\alpha \in [\tau]$ in parallel:

$$\mathsf{srk}^{(\alpha)} \leftarrow \mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha) \in \mathbb{F}_2^{M_\alpha}$$

2. **Round iteration**:
   (a) Initialize $\boldsymbol{y}^{(0)} \leftarrow \mathbf{0} \in \mathbb{F}_2^M$.
   (b) For round $\alpha = 1, \ldots, \tau$ in sequential:
      i. *Confusion*: $\boldsymbol{w}^{(\alpha)} \leftarrow \mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$.
      ii. *Diffusion*: $\boldsymbol{z}^{(\alpha)} \leftarrow \mathsf{Linear}(\boldsymbol{w}^{(\alpha)}; D^{(\alpha)}) \in \mathbb{F}_2^{M_\alpha}$.
      iii. *Key mixing*: $\boldsymbol{y}^{(\alpha)} \leftarrow \mathsf{srk}^{(\alpha)} \oplus \boldsymbol{z}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.
   (c) Set $y \leftarrow \boldsymbol{y}_{(1)}^{(\tau)} \in \{0, 1\}$.
3. Output $y \in \{0, 1\}$.

Now we define the unspecified $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{Linear}, \mathsf{S\text{-}Box})$ as follows.

- $\mathsf{DMgen}(x, \alpha)$: On input $x \in \{0, 1\}^n$, $\alpha \in [\tau]$, set

$$\mathsf{sta}^{(\alpha)} = x \in \mathbb{F}_2^n,$$
$$M^{(\alpha)} = \mathbf{I}_{M_\alpha} \in \mathbb{F}_2^{M_\alpha \times M_\alpha}.$$

  where $\mathbf{I}_{M_\alpha}$ is the identity matrix. Output the status $\mathsf{sta}^{(\alpha)}$, and diffusion matrix $M^{(\alpha)}$.

- $\mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha)$: On input $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$, $\mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^n$, $\alpha \in [\tau]$,
  1. Parse $\boldsymbol{S} = (\boldsymbol{s}_{(i)}^{(\alpha)})_{\alpha \in [\tau], i \in [M]} \in \mathbb{F}_2^{M\tau \times n}$.
  2. Set $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}_{(1)}^{(\alpha)}, \ldots, \boldsymbol{s}_{(M_\alpha)}^{(\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$.
  3. Compute

$$\mathsf{srk}^{(\alpha)} = \boldsymbol{S}^{(\alpha)} \cdot \mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha},$$

  4. Output private round key $\mathsf{srk}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.

- $\mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)})$: On input $\boldsymbol{y}^{(\alpha)} \in \mathbb{F}_2^{M_{\alpha-1}}$, output

$$\boldsymbol{w}^{(\alpha)} = \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha},$$

  where $\mathsf{S\text{-}Ber}(\cdot)$ is defined in Definition 14.

- $\mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)})$: On input $\boldsymbol{w}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$, $M^{(\alpha)} \in \mathbb{F}_2^{M_\alpha \times M_\alpha}$, output

$$\boldsymbol{z}^{(\alpha)} = \boldsymbol{w}^{(\alpha)} \cdot M^{(\alpha)} = \boldsymbol{w}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha},$$

  where the last equation follows since $M^{(\alpha)}$ is the identity matrix.

Remark 7. For our weak PRFs, the diffusion matrix for each layer is simply an identity matrix.

7 EI-PRFs in NC^1 from Sparse LPN

7.1 Full Rank Sparse LPN ($\mathsf{FRsparseLPN}$)

For the purpose of modularity, we introduce the following variant of sparse LPN ($\mathsf{sparseLPN}$) assumption, which we call *full rank sparse LPN* ($\mathsf{FRsparseLPN}$) assumption. This assumption additionally requires that each square chunk of the public matrix $\boldsymbol{A}$ is full rank. We will later show this variant is implied by original sparse LPN assumption. This assumption is stated as follows.

Definition 15 (**Full rank sparse LPN**). Let n be a security parameter. Let $m = \mathsf{poly}(n)$ and $m = m' \cdot n$ for an integer m'. Let $\mu = n^{-c}$ for $c \in (0, 1)$. Let the sparsity parameter $t = \omega(1)$ be an odd integer. The $\mathsf{FRsparseLPN}_{n,\mu,t,m}$ assumption states that

$$(\boldsymbol{A}, \boldsymbol{s} \cdot \boldsymbol{A} + \boldsymbol{e}) \overset{c}{\approx} (\boldsymbol{A}, U_m), \tag{12}$$

where $\boldsymbol{s} \leftarrow \mathbb{F}_2^n$ and $\boldsymbol{e} \leftarrow \mathsf{Ber}_\mu^m$, $\boldsymbol{A} = (\boldsymbol{A}_1, \dots \boldsymbol{A}_{m'}) \leftarrow \mathbb{F}_2^{n\times m}$ where $\boldsymbol{A}_i \in \mathbb{F}_2^{n\times n}$ for $i \in [m']$, such that (1) every column of $\boldsymbol{A}$ has exactly t non-zero elements, and (2) $\boldsymbol{A}_i$ is full rank for $i \in [m']$.

Remark 8. It is necessary that the sparsity parameter t is an odd number. Otherwise, there is no chance for $\boldsymbol{A}_i$ to be full rank. In particular, if t is even, any unit vector is not in the space spanned by columns of $\boldsymbol{A}_i$.

7.1.1 Reduction from $\mathsf{sparseLPN}$ to $\mathsf{FRsparseLPN}$

We now show that the introduced full rank version of sparse LPN problem, i.e., $\mathsf{FRsparseLPN}$ is as hard as the original sparse LPN problem, i.e., $\mathsf{sparseLPN}$.

Theorem 7 (**Reduction from $\mathsf{sparseLPN}$ to $\mathsf{FRsparseLPN}$**). *Let n, m' be integers, $m = m'n$. Let $0 < c < 1, \mu = n^{-c}$. Let t be an odd number, which is the sparsity parameter. There exists some constant $c' \in (0, 7)$, such that for any efficient adversary $\mathcal{A}$, we have*

$$\mathsf{Adv}_{\mathsf{FRsparseLPN}}(n, \mu, t, m) \leq \mathsf{Adv}_{\mathsf{sparseLPN}}(n, \mu, t, m'n^{c'}) + \mathsf{negl}(n)$$

Proof. It suffices to prove that a "wide", random sparse matrix $\boldsymbol{A} \in \mathbb{F}_2^{n\times n^{c'}}$ (such that each column has exactly t ones) has rank n except with negligible probability.

A random matrix $\boldsymbol{A}' \in \mathbb{F}_2^{n\times n^2}$ has rank n except with negligible probability (see Lemma 14). Our proof strategy is then to "simulate" a random vector in $\mathbb{F}_2^n$ (serving as one column of $\boldsymbol{A}'$) by taking the summation of, say n^5, random sparse vectors. We "simulate" each of n^2 columns of $\boldsymbol{A}'$ one by one. Consequently, a random sparse matrix $\boldsymbol{A} \in \mathbb{F}_2^{n\times n^7}$ will have rank n except with negligible probability.

All in all, it suffices to prove the following lemma.

Lemma 4. *Let n be an integer, t be an odd integer, $m = n^5$, $\boldsymbol{v}_1, \dots, \boldsymbol{v}_m \in \mathbb{F}_2^n$ be random sparse vectors such that each vector has exactly t ones. Define the random variable m' as follows.*

$$m' = \begin{cases} m & w.p.\ 0.5, \\ m-1 & w.p.\ 0.5. \end{cases}$$

Define the random variable $S = \boldsymbol{v}_1 + \boldsymbol{v}_2 + \cdots + \boldsymbol{v}_{m'} \in \mathbb{F}_2^n$. Then it holds that

$$\mathsf{SD}(S, U_n) \leq \mathsf{negl}(n).$$

Proof sketch: The intuition is that if each $\boldsymbol{v}_i$ is a sparse vector in the sense that $\boldsymbol{v}_i \sim \mathsf{Ber}_{t/n}^n$. By Piling-up lemma (Lemma 17), we immediately have $\mathsf{SD}(S, U_n) \leq \mathsf{negl}(n)$. But here each $\boldsymbol{v}_i \in \mathbb{F}_2^n$ has exactly t ones, therefore each coordinate of $\boldsymbol{v}_i$ is no longer independent. We use discrete Fourier analysis to deal with this issue. The full proof is in Appendix B.

□

7.2 Construction

Notation. To simplify notation, throughout this section we use $x_i \in \{0,1\}^{\log n}$ to denote the i-th $\log n$ chunk of $x \in \{0,1\}^{k \log n}$ for $i \in [k]$, i.e., x_i starts at position $(i-1)\log n + 1$, and ends at position $i \log n$ of x. We use $x_{j \ldots i}$ to denote $x_j x_{j+1} \cdots x_i \in \{0,1\}^{(j-i+1)\log n}$.

Definition 16 (Column sparse matrix). Let n, m, t be integers. Denote the set of matrices in $\mathbb{F}_2^{n\times m}$ that has exactly t ones in each column by $\mathbb{S}^{n\times m}[t]$. Denote $\mathbb{S}^{n\times m}[\leq t] = \bigcup_{t' \leq t} \mathbb{S}^{n \times m}[t']$

7.2.1 Input Encoding

Before we present our input encoder, we first introduce the SSMM function that will be used.

Definition 17 (Selective Sparse Matrix Multiplication). Let n, m, k, t be integers. Let $\boldsymbol{A}_0, \ldots, \boldsymbol{A}_m \in \mathbb{S}^{n\times n}[t]$. Selective sparse matrix multiplication function $\mathsf{SSMM}_{(\boldsymbol{A}_0,\ldots,\boldsymbol{A}_m)}$ with respect to $(\boldsymbol{A}_0, \ldots, \boldsymbol{A}_m)$ takes as input $x \in [m]^k$, and outputs $\prod_{i=1}^k \boldsymbol{A}_{x_i}$.

Remark 9. A related problem is "iterated matrix multiplication" (IMM), which takes as input k matrices and output their product. Besides working with sparse matrix, the key difference between SSMM and IMM is that in SSMM all matrices are fixed in advance.

We now define the input encoder, which is considered as part of public parameters of the PRF constructed later.

Definition 18 (Input Encoder). The input encoder is defined with respect to the following public parameters pp:

1. Full rank sparse LPN (Definition 15) parameter (n, t).
2. t-sparse full-rank random matrices $\boldsymbol{A}_0, \ldots, \boldsymbol{A}_n \in \mathbb{S}^{n\times n}[t]$.
3. Input length $k \log n$, where $k = \omega(1)$.

The *input encoder* $\mathsf{Encode} = \mathsf{Encode}_{n,t,k,\{\boldsymbol{A}\}} : \{0,1\}^{k \log n} \to \mathbb{F}_2^{k(k+1)\times n \times n}$ is defined as

$$\mathsf{Encode}(x) = (\mathbf{A}_{(i,j)}(x))_{i\in[k], j\in[k+1]}, \tag{13}$$

where

$$\mathbf{A}_{(i,j)}(x) = \mathsf{SSMM}_{(\boldsymbol{A}_0,\ldots,\boldsymbol{A}_n)}(x_{j\ldots i}), \tag{14}$$

and where SSMM is defined in Definition 17.

For example, let input $x = x_1x_2x_3x_4 \in \{0,1\}^{4\log n}$, where $x_1, x_2, x_3, x_4 \in \{0,1\}^{\log n}$. Then, $\mathbf{A}_{(4,2)}(x) = \boldsymbol{A}_{x_2} \cdot \boldsymbol{A}_{x_3} \cdot \boldsymbol{A}_{x_4}$.

Sparsity. We now show that $\mathbf{A}_{(i,j)}(x)$ (a product of column sparse matrices) retains certain column sparsity, which is the crucial property for achieving low-depth PRFs in our construction.

Lemma 5 (**Sparsity of $\mathbf{A}_{(i,j)}$**)**.** *For $x \in \{0,1\}^{k\log n}$, $i, j \in [k]$, $\mathbf{A}_{(i,j)}(x) \in \mathbb{S}^{n\times n}[\leq t^k]$.*

Proof. Without loss of generality, we prove the case for $i = n, j = 1$. We proceed by induction on k.

- For $k = 1$, we have $\mathbf{A}_{(k,1)}(x) = \boldsymbol{A}_x$. By definition of $\boldsymbol{A}_x$, each column of $A(x)$ has at most $t^k = t$ non-zero entries.
- For $k > 1$. $\mathbf{A}_{(k,1)}(x) = \boldsymbol{A}_{x_1} \cdot \mathbf{A}_{(k-1,1)}(x_{2\ldots k})$. By the inductive assumption, each column of $\mathbf{A}_{(k-1,1)}(x_{2\ldots k})$ has at most t^{k-1} non-zero elements. Therefore, each column of $\mathbf{A}_{(k,1)}(x)$ is formed by summation of at most t^{k-1} number of columns in $\boldsymbol{A}_{x_1}$. Therefore, each column of $\mathbf{A}_{(k,1)}(x)$ has at most $t \cdot t^{k-1} = t^k$ non-zero entries.

$\square$

Efficiency. For input $x \in \{0,1\}^{k\log n}$ where $k = \omega(1)$, the depth of the input encoder boils down to the depth efficiency of the SSMM function.

The following theorem states that computing SSMM with respect to *sub-polynomial sparse* matrices (i.e., matrices in $\mathbb{S}^{n\times n}[n^{o(1)}]$) in depth $O(\log n)$ reduces to computing SSMM with respect to *constant sparse* matrices (i.e., in $\mathbb{S}^{n\times n}[O(1)]$) in depth $O(\log n)$.

Theorem 8 (**Reduction from SSMM over $\mathbb{S}[n^{o(1)}]$ to SSMM over $\mathbb{S}[O(1)]$**)**.** *Let n be an integer, $t = t(n), k = k(n)$, $m = \mathsf{poly}(n)$. Let $\boldsymbol{A}_0, \ldots, \boldsymbol{A}_m \in \mathbb{S}^{n\times n}[t]$ be any t-sparse matrices. Let $d_{t,k} = d_{t,k}(n)$ be the depth required to compute $\mathsf{SSMM}_{\{\boldsymbol{A}_1,\ldots,\boldsymbol{A}_m\}}$. It holds that, if there exist an odd number $c = O(1)$ and $k = \omega(1)$ such that $d_{c,k} = O(\log n)$, then for any odd number $t = n^{o(1)}$, there exists $k' = \omega(1)$ such that $d_{t,k'} = O(\log n)$.*

Proof. Let $t = n^{o(1)}$, $\boldsymbol{A}_i \in \mathbb{S}^{n\times n}[t]$ for $i \in [m]$ be any t-sparse matrices. We first make the following important observation.

Claim 5. *Let $c = O(1)$ be any odd number. For any t, there exists $t' = O(t)$, such that each matrix in $\mathbb{S}^{n\times n}[t]$ can be written as the summation of t' matrices in $\mathbb{S}^{n\times n}[c]$.*

Proof. It suffices to consider each column independently.

If $c = 1$, then the claim follows trivially, since each vector in $\mathbb{S}^n[t]$ can be written as the sum of $t' = t$ unit vectors (i.e., in $\mathbb{S}^n[1]$).

If $c \geq 3$, it suffices to show that each unit vector in $\mathbb{S}^n[1]$ can be written as the sum of 3 vectors in $\mathbb{S}^n[c]$. Then it follows that each vector in $\mathbb{S}^n[t]$ can be written as the sum of $t' = 3t = O(t)$ vectors in $\mathbb{S}^n[c]$.

Claim 6. *Let $c \geq 3$ be an odd constant number. Any unit vector in $\mathbb{S}^n[1]$ can be written as the sum of 3 vectors in $\mathbb{S}^n[c]$.*

Proof. Without loss of generality, let the unit vector be $\boldsymbol{e}_1 = (1, 0, \ldots, 0) \in \mathbb{S}^n[1]$. We construct $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3 \in \mathbb{S}^n[c]$ as follows:

$$\begin{cases} \boldsymbol{a}_1 = (1, \overbrace{1, \ldots, 1}^{k}, \overbrace{1, \ldots, 1}^{k}, \overbrace{0, \ldots, 0}^{k+1}, 0, \ldots, 0) \\ \boldsymbol{a}_2 = (0, \overbrace{1, \ldots, 1}^{k}, \overbrace{0, \ldots, 0}^{k}, \overbrace{1, \ldots, 1}^{k+1}, 0, \ldots, 0) \\ \boldsymbol{a}_3 = (0, \overbrace{0, \ldots, 0}^{k}, \overbrace{1, \ldots, 1}^{k}, \overbrace{1, \ldots, 1}^{k+1}, 0, \ldots, 0), \end{cases}$$

where $c = 2k + 1$ for some k. Therefore, $\boldsymbol{e}_1 = \boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{a}_3$.

□

□

By Claim 5, for each $\boldsymbol{A}_i \in \mathbb{S}^{n\times n}[t]$, $i \in [m]$, there exist $\boldsymbol{B}_{i,1}, \ldots, \boldsymbol{B}_{i,t'} \in \mathbb{S}^{n\times n}[c]$, such that $\boldsymbol{A}_i = \sum_{j=1}^{t'} \boldsymbol{B}_{i,j}$.

Let $k' = \omega(1)$ be a parameter to be determined later. Let $\mathbf{C}' = \mathbf{C}'_{(\boldsymbol{B}_{i,j})_{i\in[m],j\in[t']}}$ be the circuit computing the SSMM function with respect to $(\boldsymbol{B}_{i,j})_{i\in[m],j\in[t']}$: On input $x \in ([m] \times [t'])^{k'}$, $\mathbf{C}'(x) = \prod_{i=1}^{k'} \boldsymbol{B}_{x_i}$. We construct a circuit $\mathbf{C} = \mathbf{C}_{(\boldsymbol{A}_1,\ldots,\boldsymbol{A}_m)}$ computing SSMM with respect to $(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m)$ as follows.

On input $x \in [m]^{k'}$:

1. Parse $x = (x_1, \ldots, x_{k'})$ where $x_i \in [m]$.
2. For each $\boldsymbol{j} \in [t']^{k'}$ do in parallel:
   (a) Set $x'_i \leftarrow (x_i, \boldsymbol{j}_i) \in [m] \times [t']$, for $i \in [k']$.
   (b) Set $x^{\boldsymbol{j}} \leftarrow (x'_1, \ldots, x'_{k'}) \in ([m] \times [t'])^{k'}$.
   (c) Set $\boldsymbol{P}_{\boldsymbol{j}} \leftarrow \mathbf{C}'(x^{\boldsymbol{j}}) \in \mathbb{F}_2^{n\times n}$.
3. Set $\boldsymbol{P} = \sum_{\boldsymbol{j}\in[t']^{k'}} \boldsymbol{P}_{\boldsymbol{j}}$. Output $\boldsymbol{P} \in \mathbb{F}_2^{n\times n}$.

We now analyze the correctness and efficiency of $\mathbf{C}$:

- **Correctness**: By definition, for $x \in [m]^{k'}$, we have

$$\begin{aligned}\mathsf{SSMM}_{(\boldsymbol{A}_1,\ldots,\boldsymbol{A}_m)}(x) &= \prod_{i\in[k']} \boldsymbol{A}_{x_i} = \prod_{i\in[k']}\sum_{j\in[t']} \boldsymbol{B}_{x_i,j} = \sum_{\boldsymbol{j}\in[t']^{k'}} \prod_{i\in[k']} \boldsymbol{B}_{x_i,\boldsymbol{j}_i} \\ &= \sum_{\boldsymbol{j}\in[t']^{k'}} \mathsf{SSMM}_{(\boldsymbol{B}_{i,j})_{i\in[m],j\in[t']}}(x^{\boldsymbol{j}}) \\ &= \sum_{\boldsymbol{j}\in[t']^{k'}} \mathbf{C}'_{(\boldsymbol{B}_{i,j})_{i\in[m],j\in[t']}}(x^{\boldsymbol{j}}) = \mathbf{C}_{(\boldsymbol{A}_1,\ldots,\boldsymbol{A}_m)}(x).\end{aligned} \tag{15}$$

- **Efficiency**: The size of $\mathbf{C}$ is $|\mathbf{C}| = O((t')^{k'} \cdot |\mathbf{C}'|)$. The depth of C is $d_{\mathbf{C}} = O(d_{\mathbf{C}'} + \log(t')^{k'})$. If $|\mathsf{C}'| = \mathsf{poly(n)}$ and $d_{\mathbf{C}'} = O(\log n)$, then $|C| = \mathsf{poly}(n)$ and $d_{\mathbf{C}} = O(\log n)$ for $k' = \omega(1) \cap O(\frac{\log n}{\log t'})$.

$\square$

Remark 10. The input encoding (i.e., SSMM function) is computable in depth $O(\log k \cdot \log n) = \omega(\log n)$ by naive iterated matrix multiplication. Jumping ahead, it is the most expensive component in our PRF construction to be presented later (all other operations are in logarithmic depth). However, we observe that the computation does not require access to any secret key, and thus can be done by public computation.

Remark 11. To date, it is not known that whether SSMM can be computed in depth $O(\log n)$ (even for constant sparsity matrices). The above theorem implies that SSMM with any sub-polynomial (i.e., $n^{o(1)}$) sparsity is no harder than SSMM with constant sparsity. Therefore, it allows us to use $n^{o(1)}$-sparsity matrices "for free": we can base our PRFs on sparse LPN with $n^{o(1)}$ sparsity (more conservative assumption than constant sparsity) without sacrificing its potential to be computed in logarithmic depth.

7.2.2 Preprocessing Input Encoding

We now introduce our two trimming functions, i.e., CTrim and RTrim, which are used to leverage the sparsity property for low-depth computation in our construction.

Definition 19 (Column Trim function). Let n, m, m' be integers, $m' \leq m$. We define the Column Trim function $\mathsf{CTrim}_{m'} : \mathbb{Z}_2^{n\times m} \to \mathbb{Z}_2^{n\times m'}$. On input $\boldsymbol{A} \in \mathbb{Z}_2^{n\times m}$, $\mathsf{CTrim}_{m'}(\boldsymbol{A})$ outputs the first m' columns of $\boldsymbol{A}$.

It follows by definition, CTrim has the following property.

Lemma 6 (Property of CTrim). *Let n, m, m' be integers. $m' \leq m$ For any $\boldsymbol{s} \in \mathbb{F}_2^n$, $\boldsymbol{A} \in \mathbb{F}_2^{n\times m}$, it holds that $\boldsymbol{s} \cdot \mathsf{CTrim}_{m'}(\boldsymbol{A}) = \mathsf{CTrim}_{m'}(\boldsymbol{s} \cdot \boldsymbol{A})$.*

Lemma 7 (Depth of CTrim). *$\mathsf{CTrim}_{m'}$ can be implemented by a circuit in NC^1.*

Definition 20 (**Row Trim function**)**.** Let n, m, t be integers. We define the Row Trim function $\mathsf{RTrim} : \mathbb{S}^{n\times m}[\leq t] \to \mathbb{S}^{mt\times m}[\leq t]$ (Definition 16). On input $\boldsymbol{A} \in \mathbb{S}^{n\times m}[\leq t]$, $\mathsf{RTrim}(\boldsymbol{A})$ does the following: from top to bottom, delete zero rows in $\boldsymbol{A}$ until there are only mt rows left, and output the resulting matrix in $\mathbb{S}^{mt\times m}[\leq t]$.

Definition 21 (**Scatter function**)**.** Let n, m, t be integers, $\mu > 0$. Define the (randomized) function $\mathsf{Scatter} : \mathbb{F}_2^t \times \mathbb{S}^{n\times m}[\leq t] \to \mathbb{F}_2^n$ as follows: on input $\boldsymbol{r} \in \mathbb{F}_2^t$, $\boldsymbol{A} \in \mathbb{S}^{n\times m}[\leq t]$,

```
1. Sample $\boldsymbol{e} = (e_1, \dots, e_n) \leftarrow \mathsf{Ber}_\mu^n$.
2. For $1 \leq i \leq mt$:
   (a) If $\boldsymbol{A}$ has less than $i$ non-zero rows, end the loop.
   (b) Find $j \in [n]$, such that the $j$-th row of $\boldsymbol{A}$ is its $i$-th non-zero row.
   (c) Set $e_j \leftarrow \boldsymbol{r}_i$.
3. Return $\boldsymbol{e} = (e_1, \dots, e_n) \in \mathbb{F}_2^n$.
```

It follows by definition, RTrim has the following property.

Lemma 8 (**Property of Row Trim**)**.** *For any $\boldsymbol{r} \in \mathbb{F}_2^t$, $\boldsymbol{A} \in \mathbb{S}^{n\times m}[\leq t]$ (Definition 16), it holds that $\boldsymbol{r} \cdot \mathsf{RTrim}(\boldsymbol{A}) = \mathsf{Scatter}(\boldsymbol{r}, \boldsymbol{A}) \cdot \boldsymbol{A}$, where $\mathsf{Scatter}$ is defined in Definition 21.*

We now show that RTrim can also be implemented in NC^1.

Lemma 9 (**Depth of RTrim**)**.** *RTrim can be implemented by a circuit in NC^1.*

Proof. We construct the circuit $\mathbf{C}_{\mathsf{RTrim}}$ computing RTrim as follows. On input $\boldsymbol{A} \in \mathbb{S}^{n\times m}[\leq t]$:

```
1. $a_i \leftarrow \vee_{j=1}^m \boldsymbol{A}_{i,j} \in \mathbb{Z}_2$ for $i \in [n]$ in parallel.
2. $b_i \leftarrow \sum_{j=1}^i a_i \in \mathbb{Z}$ (addition over integer) for $i \in [n]$ in parallel.
3. $\boldsymbol{B} \leftarrow \mathbf{0} \in \mathbb{F}_2^{mt\times m}$.
4. For $j \in [mt]$ in parallel:
   (a) For $i \in [n]$ in parallel:
       $$c_i = \begin{cases} \mathbb{I}[b_i = j], & j = 1, \\ \mathbb{I}[b_i = j \wedge \boldsymbol{b}_{i-1} = j-1] & j > 1. \end{cases}$$
   (b) Compute $p \leftarrow \sum_{i=1}^n c_i \cdot i$.
   (c) Row copy $\boldsymbol{B}_j \leftarrow \boldsymbol{A}_p$.
5. Output $\boldsymbol{B} \in \mathbb{S}^{mt\times m}[\leq t]$.
```

Since each step can be implemented in NC^1, thus $\mathbf{C}_{\mathsf{RTrim}} \in \mathsf{NC}^1$. □

7.2.3 Putting it All Together

Construction 2 (PRF from sparse full rank LPN). The PRF function is defined with respect to the following public parameters pp:

1. Full rank sparse LPN (Definition 15) parameter (n, μ, t, m), where $\mu = 2^{-\ell}$ for some $\ell \in \mathbb{N}$, $m = n^2$.
2. t-sparse full-rank random matrices $\boldsymbol{A}_0, \ldots, \boldsymbol{A}_n \in \mathbb{S}^{n \times n}[t]$ (Definition 16).
3. Recursion depth $\tau \in \mathbb{N}$.
4. Input length $k \log n$, where $k = \omega(1)$.

Let $\mathsf{Encode} = \mathsf{Encode}_{n,t,k,\{\boldsymbol{A}\}}$ be the input encoder defined in Definition 18. Denote $M_\alpha = (t^k \cdot k \cdot \ell)^{\tau-\alpha}$ for $0 \le \alpha \le \tau$, and $M = M_0 = (t^k \cdot k \cdot \ell)^\tau$. A member of the function family

$$\mathcal{F} = \mathcal{F}_{n,\mu,t,\{\boldsymbol{A}_0,\ldots,\boldsymbol{A}_n\},\tau,k} = \{F_{\boldsymbol{S}} \colon \{0,1\}^{k \log n} \to \{0,1\}, \boldsymbol{S} \in \mathbb{F}_2^{\tau k \times n}\} \tag{16}$$

is indexed by the secret key

$$\boldsymbol{S} = \left(\boldsymbol{s}_{(i)}^{(\alpha)}\right)_{\alpha \in [\tau], i \in [k]} \in \mathbb{F}_2^{\tau k \times n}.$$

On input $x \in \{0,1\}^{k \log n}$:

1. $\mathbf{A}_{(i,j)} \leftarrow \mathsf{Encode}(x)_{(i,j)} = \mathsf{SSMM}_{\{\boldsymbol{A}_0,\ldots,\boldsymbol{A}_n\}}(x_{j\ldots i}) \in \mathbb{F}_2^{n \times n}$ (Definition 18) for $i \in [k], j \in [i+1]$ in parallel.
2. $\overline{\mathbf{A}}_{(i,1)}^{(\alpha)} \leftarrow \mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,1)}) \in \mathbb{F}_2^{n \times M_\alpha}$ (Definition 19) for $i \in [k], \alpha \in [\tau]$ in parallel.
3. $\overline{\overline{\mathbf{A}}}_{(i,j)}^{(\alpha)} \leftarrow \mathsf{RTrim}(\mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,j)})) \in \mathbb{F}_2^{M_\alpha t^k \times M_\alpha}$ (Definition 20) for $i \in [k], j \in [2, i+1], \alpha \in [\tau]$ in parallel.
4. $\boldsymbol{r}_{(i)}^{(\alpha)} \leftarrow \boldsymbol{s}_{(i)}^{(\alpha)} \cdot \overline{\mathbf{A}}_{(i,1)}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$ for $\alpha \in [\tau], i \in [k]$ in parallel.
5. $\boldsymbol{y}_{(i)}^{(0)} \leftarrow \mathbf{0} \in \mathbb{F}_2^{M}$ for $i \in [k]$ in parallel.
6. For $\alpha = 1, \ldots, \tau$ *in sequential*:
   (a) $\boldsymbol{e}_{(i)}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}_\mu\left(\boldsymbol{y}_{(i)}^{(\alpha-1)}\right) \in \mathbb{F}_2^{M_\alpha \cdot k \cdot t^k}$ (Definition 14) for $i \in [k]$ in parallel
   (b) Parse $\boldsymbol{e}_{(i)}^{(\alpha)} = (\boldsymbol{e}_{(i\to 1)}^{(\alpha)}, \ldots, \boldsymbol{e}_{(i \to k)}^{(\alpha)})$, where $\boldsymbol{e}_{(i\to j)}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha t^k}$ for $j \in [k]$.
   (c) $\boldsymbol{z}_{(i)}^{(\alpha)} \leftarrow \sum_{j=1}^{i} \boldsymbol{e}_{(j \to i)}^{(\alpha)} \cdot \overline{\overline{\mathbf{A}}}_{(i,j+1)}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$ for $i \in [k]$ in parallel.
   (d) $\boldsymbol{y}_{(i)}^{(\alpha)} \leftarrow \boldsymbol{r}_{(i)}^{(\alpha)} + \boldsymbol{z}_{(i)}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$ for $i \in [k]$ in parallel.
7. Output $y \leftarrow \boldsymbol{y}_{(k)}^{(\tau)} \in \mathbb{F}_2$.

Theorem 9 (**Security**)**.** *Let n be a security parameter. Let $\mathcal{F} = \mathcal{F}_{n,\mu,t,\{\boldsymbol{A}_0,\ldots,\boldsymbol{A}_n\},\tau,k}$ be the function family defined in Construction 2. Let $\ell = -\log\mu$, $M = (t^k \cdot k \cdot \ell)^\tau$. Then, for any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, t, k, Q) \le Q \cdot \tau \cdot k^2 \cdot \mathsf{Adv}_{\mathsf{FRsparseLPN}}(n, \mu, t, n^2) + n^k \cdot (\mu k M)^{\tau-1}.$$

In particular, assuming $\mathsf{sparseLPN}$ *(Definition 10) is hard and setting*

$$\begin{cases} \mu = n^{-c}, \text{ for any } c \in (0,1) \\ t = \omega(1) \cap O(n^{o(1)}) \\ \tau = \omega(1) \cap O(\log n/\log\log n) \\ k = \omega(1) \cap O(\log n/(\tau \log t)), \end{cases}$$

imply $\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, t, k, Q) = \mathsf{negl}(n)$ *by Theorem 7.*

Proof sketch: We show that constructed PRF will have the same input-output behavior as its noisy version except with negligible probability, where the security of the latter can be proven from the sparse full rank LPN assumption via a hybrid argument on round iteration. The full proof is in Appendix G.

Remark 12. (Concrete security) In terms of the concrete security (T, Q, ϵ) of PRFs (where any adversary making at most Q queries and running in time T has an advantage of at most ϵ), our PRFs are only *provably secure* under (slightly) superpolymial T, Q, and $1/\epsilon$. Note that this is true even if we assume subexponential hardness for sparse LPN problem. However, we are not aware of any quasipolynomial-time attacks that break the security of our PRFs.

Remark 13. (On basing security on LPN) It is known that sparse LPN is hard assuming subexponential hardness of LPN problem [JLS24, BBTV24]. However, that reduction is meaningful only when the noise level μ is constant. Our construction requires μ to be inverse-polynomial, thus falls outside the meaningful parameter regime covered by the existing reduction.

Theorem 10 (**AKH structure**)**.** *Under the parameter setting in Theorem 9, Construction 2 has γ-AKH structure (Definition 13) where $\gamma = n^{-c+\delta}$ for any $\delta > 0$.*

Proof sketch: Observe that the output of the PRF takes the form $\left\langle \boldsymbol{s}_{(k)}^{(\tau)}, \boldsymbol{a}_x \right\rangle + z_{(k)}^{(\alpha)}$. We need show that $z_{(k)}^{(\alpha)}$ is zero except with probability $n^{-c+\delta}$ over random $\boldsymbol{S}$. This follows as a byproduct of the security proof of Theorem 9. We refer to Appendix H for the full proof.

Claim 7 (**Efficiency**)**.** *Under the parameter setting in Theorem 9, Construction 2 can be computed by a circuit of depth $\omega(\log n)$ for any constant $\epsilon > 0$. Ignoring the depth of input encoding (Step 1), it can be computed by a circuit of depth $O(\log n)$.*

Proof. Step 1 has depth $\omega(\log n)$. Step 2 and 3 have depth $O(\log n)$ by Lemma 7 and Lemma 9. Step 4 has depth $O(\log n)$ since it is matrix-vector multiplication with dimension n. Step 6 has depth $O(\log n)$ since $\mathsf{S\text{-}Ber}$ has depth $O(\log \ell)$ by Claim 2, and $\boldsymbol{e} \cdot \overline{\overline{\mathbf{A}}}$ (matrix-vector multiplication with dimension $O(Mt)$) has depth $O(\log Mt) = O(\log n/\tau)$. Others are all constant depth. $\square$

Combining Theorem 9 (security), Theorem 10 (AKH structure), Claim 7 (efficiency), and Theorem 4 (AKH structure implies AKH-PRFs), we have the following corollary.

Corollary 2. *Let n be a security parameter. Let $\mu = n^{-c}$ for any constant $c \in (0,1)$. Let sparsity parameter $t = O(n^{o(1)})$. Assuming $\mathsf{sparseLPN}_{n,\mu,t,m}$ (Definition 15) is hard for $m = n^{c'}$, $c' \in (0,9)$, then there exists a key-homomorphic PRF (Definition 2) $F \colon \mathcal{K} \times \{0,1\}^{\omega(\log n)} \to \{0,1\}^{\mathsf{poly}(n)}$ computable in $\mathsf{NC}^{1+\epsilon}$ for any $\epsilon > 0$, and a key-homomorphic encoded-input PRF (Definition 12) $F \colon \mathcal{K} \times \{0,1\}^{\omega(\log n)} \to \{0,1\}^{\mathsf{poly}(n)}$ computable in NC^1.*

7.3 Our Construction as an SPN

Our construction nicely fit into the substitution-permutation network (SPN) design framework used in modern block ciphers. In this section, we present Construction 2 from the perspective of SPN.

We instantiate the SPN template by first specifying format of its inputs and outputs, followed by the specification of $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{Linear}, \mathsf{S\text{-}Box})$.

On input $x \in \{0,1\}^{k \log n}$, the PRF $F_{\boldsymbol{S}}$ with secret key $\boldsymbol{S} \leftarrow \mathbb{F}_2^{\tau k \times n}$ is defined as:

1. **Key schedule**:
   (a) Derive diffusion matrix for round $\alpha \in [\tau]$ in parallel:
   $$(\mathsf{sta}^{(\alpha)}, M^{(\alpha)}) \leftarrow \mathsf{DMgen}(x, \alpha)$$
   (b) Derive private round key for round $\alpha \in [\tau]$ in parallel:
   $$\mathsf{srk}^{(\alpha)} \leftarrow \mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha) \in \mathbb{F}_2^{k \times M_\alpha}$$
2. **Round iteration**:
   (a) Initialize $\boldsymbol{y}^{(0)} \leftarrow \mathbf{0} \in \mathbb{F}_2^{k \times m}$.
   (b) For round $\alpha = 1, \ldots, \tau$ in sequential:
      i. *Confusion*: $\boldsymbol{w}^{(\alpha)} \leftarrow \mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha)}) \in \mathbb{F}_2^{k \times M_\alpha \cdot k \cdot t^k}$.
      ii. *Diffusion*: $\boldsymbol{z}^{(\alpha)} \leftarrow \mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)}) \in \mathbb{F}_2^{k \times M_\alpha}$.
      iii. *Key mixing*: $\boldsymbol{y}^{(\alpha)} \leftarrow \mathsf{srk}^{(\alpha)} \oplus \boldsymbol{z}^{(\alpha)} \in \mathbb{F}_2^{k \times M_\alpha}$.
   (c) Set $y \leftarrow \boldsymbol{y}^{(\tau)}_{(k)} \in \mathbb{F}_2$.
3. Output $y \in \mathbb{F}_2$.

The tuple $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{Linear}, \mathsf{S\text{-}Box})$ is specified as follows.

- $\mathsf{DMgen}(x, \alpha)$: On input $x \in \{0,1\}^{k \log n}$, $\alpha \in [\tau]$,
  1. Compute for $i \in [k], j \in [i+1]$ in parallel:
  $$\mathbf{A}_{(i,j)} \leftarrow \mathbf{A}_{(i,j)}(x) \in \mathbb{F}_2^{n \times n}$$

2. Compute for $i \in [k]$ in parallel:
$$\overline{\mathbf{A}}^{(\alpha)}_{(i,1)} \leftarrow \mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,1)}) \in \mathbb{F}_2^{n \times M_\alpha}.$$
Set $\mathsf{sta}^{(\alpha)} = \left(\overline{\mathbf{A}}^{(\alpha)}_{(i,1)}\right)_{i \in [k]} \in \mathbb{F}_2^{k \times n \times M_\alpha}$.

3. Compute for $i \in [k], j \in [2, i+1]$ in parallel:
$$\overline{\overline{\mathbf{A}}}^{(\alpha)}_{(i,j)} \leftarrow \mathsf{RTrim}(\mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,j)})) \in \mathbb{F}_2^{M_\alpha t^k \times M_\alpha}.$$
Set $M^{(\alpha)} = \left(\overline{\overline{\mathbf{A}}}^{(\alpha)}_{(i,j)}\right)_{i \in [k], j \in [2, i+1]}$.

Output status $\mathsf{sta}^{(\alpha)}$, and diffusion matrix $M^{(\alpha)}$.

- $\mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha)$: On input $\boldsymbol{S} \in \mathbb{F}_2^{n \times \tau}$, $\mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^{k \times n \times M_\alpha}$, $\alpha \in [\tau]$.
  1. Parse $\mathsf{sta}^{(\alpha)} = \left(\overline{\mathbf{A}}^{(\alpha)}_{(i,1)}\right)_{i \in [k]}$, where $\overline{\mathbf{A}}^{(\alpha)}_{(i,1)} \in \mathbb{F}_2^{n \times M_\alpha}$.
  2. Parse $\boldsymbol{S} = (\boldsymbol{s}^{(\alpha)}_{(i)})_{\alpha \in [\tau], i \in [k]}$, where $\boldsymbol{s}^{(\alpha)}_{(i)} \in \mathbb{F}_2^n$.
  3. Compute for $i \in [k]$ in parallel:
  $$\mathsf{srk}^{(\alpha)}_{(i)} \leftarrow \boldsymbol{s}^{(\alpha)}_{(i)} \cdot \overline{\mathbf{A}}^{(\alpha)}_{(i,1)} \in \mathbb{F}_2^{M_\alpha}.$$

  Output private round key $\mathsf{srk}^{(\alpha)} = (\mathsf{srk}^{(\alpha)}_{(1)}, \ldots, \mathsf{srk}^{(\alpha)}_{(k)}) \in \mathbb{F}_2^{k \times M_\alpha}$.

- $\mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)})$: On input $\boldsymbol{y}^{(\alpha-1)} \in \mathbb{F}_2^{k \times M_{\alpha-1}}$,
  1. Parse $\boldsymbol{y}^{(\alpha-1)} = (\boldsymbol{y}^{(\alpha-1)}_{(1)}, \ldots, \boldsymbol{y}^{(\alpha-1)}_{(k)})$, where $\boldsymbol{y}^{(\alpha-1)}_{(i)} \in \mathbb{F}_2^{M_{\alpha-1}}$ for $i \in [k]$.
  2. Compute for $i \in [k]$ in parallel:
  $$\boldsymbol{w}^{(\alpha)}_{(i)} \leftarrow \mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha-1)}_{(i)}) \in \mathbb{F}_2^{M_\alpha \cdot k \cdot t^k}.$$

  Output $\boldsymbol{w}^{(\alpha)} = (\boldsymbol{w}^{(\alpha)}_{(1)}, \ldots, \boldsymbol{w}^{(\alpha)}_{(k)}) \in \mathbb{F}_2^{k \times M_\alpha \cdot k \cdot t^k}$.

- $\mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)})$: On input $\boldsymbol{w}^{(\alpha)} \in \mathbb{F}_2^{k \times M_\alpha \cdot k \cdot t^k}$,
  1. Parse $\boldsymbol{w}^{(\alpha)} = (\boldsymbol{w}^{(\alpha)}_{(1)}, \ldots, \boldsymbol{w}^{(\alpha)}_{(k)})$, where $\boldsymbol{w}^{(\alpha)}_{(i)} \in \mathbb{F}_2^{M_\alpha \cdot k \cdot t^k}$ for $i \in [k]$.
  2. Parse $\boldsymbol{w}^{(\alpha)}_{(i)} = (\boldsymbol{w}^{(\alpha)}_{(i \to 1)}, \ldots, \boldsymbol{w}^{(\alpha)}_{(i \to k)})$, where $\boldsymbol{w}^{(\alpha)}_{(i \to j)} \in \mathbb{F}_2^{M_\alpha t^k}$ for $i, j \in [k]$.
  3. Parse $M^{(\alpha)} = \left(\overline{\overline{\mathbf{A}}}^{(\alpha)}_{(i,j)}\right)_{i \in [k], j \in [2, i+1]}$, where $\overline{\overline{\mathbf{A}}}^{(\alpha)}_{(i,j)} \in \mathbb{F}_2^{M_\alpha t^k \times M_\alpha}$.
  4. Compute for $i \in [k]$ in parallel:
  $$\boldsymbol{z}^{(\alpha)}_{(i)} \leftarrow \sum_{j=1}^{i} \boldsymbol{w}^{(\alpha)}_{(j \to i)} \cdot \overline{\overline{\mathbf{A}}}^{(\alpha)}_{(i,j+1)} \in \mathbb{F}_2^{M_\alpha}.$$

  Output $\boldsymbol{z}^{(\alpha)} = (\boldsymbol{z}^{(\alpha)}_{(1)}, \ldots, \boldsymbol{z}^{(\alpha)}_{(k)}) \in \mathbb{F}_2^{k \times M_\alpha}$.

Remark 14. In our construction, the diffusion matrix for each layer depends on the input x.

8 PRFs in NC^1 from Seeded LPN

8.1 Seeded LPN Assumption

Definition 22 (Seeded LPN). Let n be a security parameter. Let $0 < c < 1, \mu = n^{-c}$. Let m be an integer. Let $p > 2^n$ be a prime number, $2^n/p = \mathsf{negl}(n)$. Let $\mathcal{H}$ be a n-wise independent hash function family mapping from $[m]$ to $\mathbb{Z}_p$. Let $\mathsf{bit\text{-}decompose}\colon \mathbb{Z}_p \to \mathbb{F}_2^n$ be the bit decomposition function. The $\mathsf{seededLPN}_{n,\mu,m,p}$ assumption states that

$$(\boldsymbol{A}, \boldsymbol{s} \cdot \boldsymbol{A} + \boldsymbol{e}, H) \overset{c}{\approx} (\boldsymbol{A}, U_m, H) \tag{17}$$

where $\boldsymbol{s} \leftarrow \mathbb{F}_2^n$, $\boldsymbol{e} \leftarrow \mathsf{Ber}_\mu^m$, $H \leftarrow \mathcal{H}$, and $\boldsymbol{A} \in \mathbb{F}_2^{n\times m}$ and the i-th column of $\boldsymbol{A}$ is $\boldsymbol{A}_i = \mathsf{bit\text{-}decompose}(H(i))$.

8.1.1 Security Against Linear Test

To give some formal confidence in the validity of the assumption, we perform initial cryptanalysis showing that it is secure within the "linear test framework". Within this framework, the adversary only tries to detect a bias in the observed samples by computing linear functions of these samples. (The choice of the linear function itself can depend arbitrarily on the code matrix.) As argued by [CRR21, BCG⁺22], this framework captures most known attacks on LPN-style assumptions (Information-Set Decoding [Pra62], BKW [BKW00], Gaussian Elimination [EKM17], Statistical Decoding [Al 01], and more).

Below we provide some background on linear test. We first introduce the notion of bias of a distribution.

Definition 23 (Bias of a Distribution). Given a distribution $\mathcal{D}$ over $\mathbb{F}^n$ and a vector $\boldsymbol{u} \in \mathbb{F}^n$, the bias of $\mathcal{D}$ with respect to $\boldsymbol{u}$, denoted $\mathsf{bias}_{\boldsymbol{u}}(\mathcal{D})$, is equal to

$$\mathsf{bias}_{\boldsymbol{u}}(\mathcal{D}) = |\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\langle \boldsymbol{u}, \boldsymbol{x}\rangle] - \mathbb{E}_{\boldsymbol{x}\sim U_n}[\langle \boldsymbol{u}, \boldsymbol{x}\rangle]| = \left| \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\langle \boldsymbol{u}, \boldsymbol{x}\rangle] - \frac{1}{|\mathbb{F}^n|} \right|,$$

where U_n denotes the uniform distribution over $\mathbb{F}^n$. The bias of $\mathcal{D}$, denoted bias($\mathcal{D}$), is the maximum bias of $\mathcal{D}$ with respect to any nonzero vector $\boldsymbol{u} \in \mathbb{F}^n \backslash \{\boldsymbol{0}\}$.

Definition 24 (Security against Linear Test). Let $\mathcal{D} = \{\mathcal{D}_{n,m}\}_{n,m\in\mathbb{N}}$ denote a family of noise distributions over $\mathbb{F}_2^m$. Let $\mathcal{C}$ be a probabilistic code generation algorithm such that $\mathcal{C}(n,m)$ outputs a matrix $A \in \mathbb{F}_2^{n\times m}$. Let $\epsilon, \delta : \mathbb{N} \to [0,1]$ be two functions. We say that the $(\mathcal{D}, \mathcal{C}, \mathbb{F}_2)$-LPN$(m,n)$ assumption with dimension n and $m = m(n)$ samples is (ϵ, δ)-secure against linear tests if for any (possibly inefficient) adversary $\mathcal{A}$ which, on input a matrix $A \in \mathbb{F}_2^{n\times m}$, outputs a nonzero $\mathbf{v} \in \mathbb{F}_2^m$, it holds that

$$\Pr[\boldsymbol{A} \leftarrow \mathcal{C}(n,m), \mathbf{v} \leftarrow \mathcal{A}(\boldsymbol{A}) : \mathsf{bias}_{\mathbf{v}}(\mathcal{D}_{\boldsymbol{A}}) \geq \epsilon(n)] \leq \delta(n),$$

where $\mathcal{D}_{\boldsymbol{A}}$ denotes the distribution induced by sampling $\boldsymbol{s} \leftarrow \mathbb{F}_2^n$, $\boldsymbol{e} \leftarrow \mathcal{D}_{n,m}$, and outputting the LPN samples $\boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} \in \mathbb{F}_2^m$.

Following [ADI+17, CRR21], we call *dual distance* of a matrix M, and write $\mathsf{dd}(M)$, the largest integer d such that every subset of d columns of M is linearly independent. The name "dual distance" stems from the fact that the $\mathsf{dd}(M)$ is also the minimum distance of the dual of the code generated by M (i.e., the code generated by the right null space of M).

Lemma 10 (**[CRR21]**). *Let $\mathcal{D} = \{\mathcal{D}_{n,m}\}_{n,m\in\mathbb{N}}$ denote a family of noise distributions over $\mathbb{F}_2^m$. Let $\mathcal{C}$ be a probabilistic code generation algorithm s.t. $\mathcal{C}(n,m) \to A \in \mathbb{F}_2^{n\times m}$. Then for any $d \in \mathbb{N}$, the $(\mathcal{D},\mathcal{C},\mathbb{F}_2)$-LPN$(n,m)$ assumption with dimension n and $m = m(n)$ samples is $(\varepsilon_d, \delta_d)$-secure against linear tests, where*

$$\varepsilon_d = \max_{|\vec{v}|>d} \mathsf{bias}_{\vec{v}}(\mathcal{D}_{n,m}), \quad \textit{and} \quad \delta_d = \Pr_{\boldsymbol{A}\leftarrow\mathcal{C}(n,m)}[\mathsf{dd}(\boldsymbol{A}) \le d].$$

Lemma 11 (**Security of seeded LPN against linear test**)**.** *Let $d < n$, $\mathsf{seededLPN}_{n,\mu,m}$ are $(\varepsilon_d, \delta_d)$-secure against linear test, where*

$$\varepsilon_d = e^{-2\mu d}, \quad \textit{and} \quad \delta_d = 2^{(2+\log m)d-n}.$$

In particular, setting

$$\begin{cases} m = 2^{o(n)} \\ d = O(n/\log m) \end{cases}$$

implies that $\varepsilon_d = \mathsf{negl}(n)$, and $\delta_d = \mathsf{negl}(n)$.

Proof. Let Bernoulli noise distribution $\mathcal{D}_{n,m} = \mathsf{Ber}_\mu^m$, for some noise rate μ. The probability that d random vectors over $\mathbb{F}_2^n$ are linearly independent is at least

$$\prod_{i=0}^{d-1} \frac{2^n - 2^i}{2^n} \ge (1 - 2^{d-1-n})^d \ge 1 - 2^{2d-n}.$$

Therefore, by a union bound, the probability that a random matrix $A \leftarrow \mathbb{F}_2^{n\times m}$ satisfies $\mathsf{dd}(A) < d$ is at most $\binom{m}{d} \cdot 2^{2d-n} \le 2^{(2+\log m)d-n}$. On the other hand, for any d and any $\vec{v}$ with $|\vec{v}| > d$, we have by Piling-up lemma (Lemma 17):

$$\Pr[\vec{e} \leftarrow \mathsf{Ber}_\mu^n : \vec{v}^T \cdot \vec{e} = 1] = \frac{1 - (1-2\mu)^d}{2},$$

hence $\mathsf{bias}_{\vec{v}}(\mathsf{Ber}_\mu^n) = (1-2\mu)^d \le e^{-2\mu d}$. In particular, setting $d = O(n/\log m)$ suffices to guarantee that with probability at least $\delta_d = 1 - 2^{-o(n)}$, the LPN samples will have bias (with respect to any possible nonzero vector $\vec{v}$) ε_d at most $e^{-O(n\mu/\log m)}$. $\square$

We note that security against linear test is not sufficient for its security. However, the following two scenarios are the only exceptions currently known [BCG+22].

1. The code is strongly algebraic. We use the bit-decomposition heuristic to destroy the underlying algebraic structure, since modulus switching corresponds to a high-degree polynomial.
2. The noise is structured (which is the case, for example, for regular noise) and the adversary can see enough samples. We use Bernoulli random noise which is not structured.

Finally, we note that while we were not able to break our seeded LPN assumption, admittedly, it is a new and strong assumption, and as such a healthy dose of skepticism is warranted.

8.2 Construction

Construction 3 (PRF from seeded LPN). The PRF function is defined with respect to the following public parameters pp:

1. Seeded LPN (Definition 22) parameter (n, μ, m, p, H), where $\mu = 2^{-\ell}$ for some $\ell \in \mathbb{N}$.
2. Recursion depth $\tau \in \mathbb{N}$.

We denote $M_\alpha = \ell^{\tau-\alpha}$ for $0 \leq \alpha \leq \tau$, and $M = M_0 = \ell^\tau$. Let $\mathcal{F}' = \mathcal{F}'_{n,\mu,\tau}$ be the weak PRF family constructed in Construction 1. Let $\mathsf{bit\text{-}decompose}\colon \mathbb{Z}_p \to \mathbb{F}_2^n$ be the bit decomposition function.

A member of the function family

$$\mathcal{F} = \mathcal{F}_{n,\mu,m,H,\tau} : [m] \to \{0,1\}, \boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}\} \tag{18}$$

is indexed by the secret key

$$\boldsymbol{S} = (\boldsymbol{s}_{(i)}^{(\alpha)})_{\alpha \in [\tau], i \in [M]} \in \mathbb{F}_2^{M\tau \times n}.$$

On input $x \in [m]$, $F_{\boldsymbol{S}}$ is defined as

1. Set $x' \leftarrow \mathsf{bit\text{-}decompose}(H(x)) \in \mathbb{F}_2^n$.
2. Output $F'_{\boldsymbol{S}}(x') \in \{0,1\}$, where $F'_{\boldsymbol{S}}$ is a member of weak PRF family $\mathcal{F}'$ indexed by $\boldsymbol{S}$.

Remark 15. Note the our construction is not a generic transformation from weak PRFs intro strong ones. We present it in such a way only to reuse the same construction that was presented earlier in the context of the weak PRF, and our security proof is non-black box.

Theorem 11 (Security). *Let n be a security parameter. Let $\mathcal{F} = \mathcal{F}_{n,\mu,m,H,\tau}$ be the function family defined in Construction 3. Let $M = (-\log \mu)^\tau$. Then, for any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, m, p) \leq M\tau \cdot \mathsf{Adv}_{\mathsf{seededLPN}}(n, \mu, m, p) + m \cdot \left((\mu M)^\tau + 2^{-(n-M\tau-1)} + 2^n/p\right).$$

In particular, assuming seeded LPN (Definition 22) is hard, and setting

$$\begin{cases} \mu = n^{-c}, \text{ for any } c \in (0,1) \\ \tau = \omega(1) \cap O(\log n/ \log\log n) \\ 2^n/p = \mathsf{negl}(n) \\ m = 2^{\omega(\log n)} \cap 2^{O(\tau \log n)} \end{cases}$$

imply $\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, m, p) = \mathsf{negl}(n)$.

Proof sketch: We show that constructed PRF will have the same input-output behavior as a variant where F' is replaced by its ideal version. The security of this variant can be proven from the seeded LPN assumption via a hybrid argument on round iteration. We refer to Appendix E for the full proof.

Remark 16. (Concrete security) In terms of the concrete security (T, Q, ϵ) of PRFs (where any adversary making at most Q queries and running in time T has an advantage of at most ϵ), our PRFs are only *provably secure* under (slightly) superpolymial T, Q, and $1/\epsilon$. Note that this is true even if we assume subexponential hardness for seeded LPN problem. However, we are not aware of any quasipolynomial-time attacks that break the security of our PRFs.

Theorem 12 (**AKH structure**). *Under the parameter setting in Theorem 11, Construction 3 has γ-AKH structure (Definition 13) where $\gamma = \mu + \mathsf{negl}(n)$.*

Proof sketch: Observe that the output of the PRF takes the form $\langle \boldsymbol{s}^{(\tau)}, \boldsymbol{a}_x \rangle + e^{(\tau)}$. We need to show that $e^{(\tau)}$ is zero except with probability $\mu + \mathsf{negl}(n)$ over random key. This follows from a byproduct of the security proof of Theorem 11. We refer to Appendix F for the full proof.

Claim 8 (**Efficiency**). *Under the parameter setting in Theorem 11, Construction 3 can be computed by a circuit of depth $O(\log n)$.*

Proof. It follows from Claim 4 that Step 2 can be computed in depth $O(\log n)$. In addition, bit decomposition and n-wise independent hash can all be computed in depth $O(\log n)$. Therefore, the overall depth is $O(\log n)$. □

Combining Theorem 11 (security), Theorem 12 (AKH structure), Claim 8 (efficiency), and Theorem 4 (AKH structure implies AKH-PRFs), we have the following corollary.

Corollary 3. *Let n be a security parameter. Let $\mu = n^{-c}$ for any constant $c \in (0, 1)$. Assume that $\mathsf{seededLPN}_{n,\mu,m,p}$ (Definition 22) is hard for some $m = 2^{\omega(\log n)}$. Then, there exists a key-homomorphic pseudorandom function (Definition 12) $F : \mathcal{K} \times \{0,1\}^{\omega(\log n)} \to \{0,1\}^{\mathsf{poly}(n)}$ that can be computed by a circuit in NC^1.*

8.3 Extension to Large Input Domain

We give two pathways to extend Construction 3 to large input domain. The first approach is black-box. In particular, we use generic domain extension techniques to extend its input domain. The second approach is non-black box, where we settle for non-adaptive security and prove that Construction 3 with an large input domain is secure against non-adaptive attackers.

Combining Theorem 11 (security), Claim 8 (efficiency), and Lemma 1 (domain extension), we have the following corollary.

Corollary 4. *Let n be a security parameter. Let $\mu = n^{-c}$ for any constant $c \in (0, 1)$. Assume that $\mathsf{seededLPN}_{n,\mu,m,H}$ (Definition 22) is hard for some $m = 2^{\omega(\log n)}$. Then, there exists a pseudorandom function (Definition 2) $F : \mathcal{K} \times \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^{\mathsf{poly}(n)}$ that can be computed by a circuit in NC^1.*

Theorem 13 (**Security against non-adaptive attacker**). *Let n be a security parameter. Let $\mathcal{F}$ be the function family defined in Construction 3. Let $M = (-\log \mu)^{\tau}$. Then, for any efficient adversary $\mathcal{A}_{\mathsf{na}}$ making at most Q non-adaptive queries, we have*

$$\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}_{\mathsf{na}}, n, \mu, \tau, m, p) \leq M\tau \cdot \mathsf{Adv}_{\mathsf{seededLPN}}(n, \mu, m, p) + Q \cdot \left((\mu M)^{\tau} + 2^{-(n - M\tau - 1)} + 2^n / p \right).$$

In particular, assuming seeded LPN (Definition 22) is hard, and setting

$$\begin{cases} \mu = n^{-c}, \text{ for any } c \in (0,1) \\ \tau = \omega(1) \cap O(\log n / \log\log n) \\ 2^n/p = \mathsf{negl}(n) \\ m = 2^{o(n)} \end{cases}$$

imply $\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}_{\mathsf{na}}, n, \mu, \tau, m, p) = \mathsf{negl}(n)$.

Proof. The proof is exactly the same as that of Theorem 11 except that we apply a union bound over all Q queries made by the non-adaptive attacker rather applying a union bound over all input domain. □

Combining Theorem 13 (non-adaptive security), Theorem 12 (AKH structure), Claim 8 (efficiency), and Theorem 4 (AKH structure implies AKH-PRFs), we have the following corollary.

Corollary 5. *Let n be a security parameter. Let $\mu = n^{-c}$ for any constant $c \in (0,1)$. Assume that* $\mathsf{seededLPN}_{n,\mu,m,p}$ *(Definition 22) is hard for any $m = 2^{o(n)}$. Then, there exists a key-homomorphic non-adaptive pseudorandom function (Definition 12) $F \colon \mathcal{K} \times \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^{\mathsf{poly}(n)}$ that can be computed by a circuit in* NC^1.

8.4 Our Construction as an SPN

As discussed in the introduction, our construction nicely fit into the substitution-permutation network (SPN) design framework used in modern block ciphers. In this section, we present Construction 3 from the perspective of SPN.

On input $x \in \mathbb{Z}_p$, the PRF $F_{\boldsymbol{S}}$ with secret key $\boldsymbol{S} \leftarrow \mathbb{F}_2^{M\tau \times n}$ is defined as:

1. **Key schedule:**
   (a) Derive diffusion matrix for round $\alpha \in [\tau]$ in parallel:
   $$(\mathsf{sta}^{(\alpha)}, D^{(\alpha)}) \leftarrow \mathsf{DMgen}(x, \alpha),$$
   where $\mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^n$, and $D^{(\alpha)} \in \mathbb{F}_2^{M_{\alpha-1} \times M_{\alpha-1}}$.
   (b) Derive private round key for round $\alpha \in [\tau]$ in parallel:
   $$\mathsf{srk}^{(\alpha)} \leftarrow \mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha) \in \mathbb{F}_2^{M_\alpha}$$
2. **Round iteration**:
   (a) Initialize $\boldsymbol{y}^{(0)} \leftarrow \mathbf{0} \in \mathbb{F}_2^M$.
   (b) For round $\alpha = 1, \ldots, \tau$ in sequential:
      i. *Confusion*: $\boldsymbol{w}^{(\alpha)} \leftarrow \mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$.
      ii. *Diffusion*: $\boldsymbol{z}^{(\alpha)} \leftarrow \mathsf{Linear}(\boldsymbol{w}^{(\alpha)}; D^{(\alpha)}) \in \mathbb{F}_2^{M_\alpha}$.

iii. *Key mixing*: $\boldsymbol{y}^{(\alpha)} \leftarrow \mathsf{srk}^{(\alpha)} \oplus \boldsymbol{z}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.

(c) Set $y \leftarrow \boldsymbol{y}^{(\tau)}_{(1)} \in \{0,1\}$.

3. Output $y \in \{0,1\}$.

Now we define the unspecified $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{Linear}, \mathsf{S\text{-}Box})$ as follows.

- $\mathsf{DMgen}(x, \alpha)$: On input $x \in \mathbb{Z}_p$, $\alpha \in [\tau]$, set

$$\mathsf{sta}^{(\alpha)} = \mathsf{bit\text{-}decompose}(H(x)) \in \mathbb{F}_2^n,$$

$$M^{(\alpha)} = \mathbf{I}_{M_\alpha} \in \mathbb{F}_2^{M_\alpha \times M_\alpha}.$$

where H is n-wise independent hash function, $\mathbf{I}_{M_\alpha}$ is the identity matrix. Output the status $\mathsf{sta}^{(\alpha)}$, and diffusion matrix $M^{(\alpha)}$.

- $\mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha)$: On input $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$, $\mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^n$, $\alpha \in [\tau]$,

  1. Parse $\boldsymbol{S} = (\boldsymbol{s}^{(\alpha)}_{(i)})_{\alpha \in [\tau], i \in [M]} \in \mathbb{F}_2^{M\tau \times n}$.
  2. Set $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}^{(\alpha)}_{(1)}, \ldots, \boldsymbol{s}^{(\alpha)}_{(M_\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$.
  3. Compute
  $$\mathsf{srk}^{(\alpha)} = \boldsymbol{S}^{(\alpha)} \cdot \mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha},$$
  4. Output private round key $\mathsf{srk}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.

- $\mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)})$: On input $\boldsymbol{y}^{(\alpha)} \in \mathbb{F}_2^{M_{\alpha-1}}$, output

$$\boldsymbol{w}^{(\alpha)} = \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha},$$

where $\mathsf{S\text{-}Ber}(\cdot)$ is defined in Definition 14.

- $\mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)})$: On input $\boldsymbol{w}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$, $M^{(\alpha)} \in \mathbb{F}_2^{M_\alpha \times M_\alpha}$, output

$$\boldsymbol{z}^{(\alpha)} = \boldsymbol{w}^{(\alpha)} \cdot M^{(\alpha)} = \boldsymbol{w}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha},$$

where the last equation follows since $M^{(\alpha)}$ is the identity matrix.

Acknowledgements

We thank Geoffroy Couteau for helpful discussions. Ding and Komargodski were supported in part by a grant from the Israel Science Foundation (ISF Grant No. 1774/20), and by the European Union (ERC, SCALE,101162665). Jain was supported by an NSF Career Award CNS 2441647, a Google faculty research scholar award, a gift from Stellar Foundation and internal grants from CMU. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

A Well-known Facts, Lemmas and Inequalities

Lemma 12 (Chernoff bound). *For any $n \in \mathbb{N}$, let $X_1, \ldots, X_n$ be independent random variables and let $\bar{X} = \sum_{i=1}^{n} X_i$, where $\Pr[0 \leq X_i \leq 1] = 1$ holds for every $1 \leq i \leq n$. Then, for any $\Delta_1 > 0$ and $0 < \Delta_2 < 1$,*

$$\Pr\left[\bar{X} > (1+\Delta_1)\cdot\mathbb{E}[\bar{X}]\right] < \exp\left(-\frac{\min(\Delta_1, \Delta_1^2)}{3}\mathbb{E}[\bar{X}]\right),$$

$$\Pr\left[\bar{X} < (1-\Delta_2)\cdot\mathbb{E}[\bar{X}]\right] < \exp\left(-\frac{\Delta_2^2}{2}\mathbb{E}[\bar{X}]\right).$$

Lemma 13 (The Hoeffding bound). *Let $q \in \mathbb{N}$, and let $\xi_1, \xi_2, \ldots, \xi_q$ be independent random variables such that for each $1 \leq i \leq q$ it holds that $\Pr[a_i \leq \xi_i \leq b_i] = 1$. Then, for any $t > 0$ we have*

$$\Pr\left[\left|\sum_{i=1}^{q}\xi_i - \mathbb{E}\left[\sum_{i=1}^{q}\xi_i\right]\right| \geq t\right] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{q}(b_i - a_i)^2}\right).$$

Lemma 14 (Lemma A.3 in [BLMR13]). *Let $n \in \mathbb{N}$, the probability that a random matrix $M \leftarrow \mathbb{F}_2^{n\times n}$ is full rank (i.e., rank n) is at least 0.288.*

Lemma 15. *Let q be prime, $m, n \in \mathbb{N}$, $m < n$, the probability that a random matrix $M \leftarrow \mathbb{Z}_q^{n\times m}$ is full rank (i.e., rank m) is at least $1 - q^{-(n-m-1)}$.*

Proof. Consider matrix M being sampled column by column, and denote $\mathcal{E}_i$ to be the event that "column i is non-zero and neither is it any linear combination of the preceding columns (i.e., columns 1 to $i-1$)".

$$\begin{aligned}\Pr[M \text{ has full rank }] &= \Pr[\mathcal{E}_1]\cdot\Pr[\mathcal{E}_2|\mathcal{E}_1]\cdots\Pr[\mathcal{E}_m|\mathcal{E}_{m-1}] \\ &= (1-q^{-n})\cdot(1-q^{-n+1})\cdots(1-q^{-n+m-1}) \\ &> \exp(-(q^{-n+1}+q^{-n+2}+\cdots+q^{-n+m})) \\ &> \exp(-q^{-(n-m-1)}) \\ &> 1-q^{-(n-m-1)},\end{aligned}$$

where the first inequality is due to Fact 2 and the last follows from Fact 1. □

Lemma 16. *If $\boldsymbol{S} \in \mathbb{Z}_q^{n\times m}$ is a full-rank matrix, then $\boldsymbol{AS}$ is distributed uniformly over $\mathbb{Z}_q^{n'\times m}$.*

Proof. Let $\boldsymbol{S}$ be viewed as $\boldsymbol{S} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n]$ for row vectors $\boldsymbol{s}_i\mathbb{Z}_q^m$. As $\mathrm{Rk}(\boldsymbol{S}) = m$, there are m rows that are linearly independent. Let $\boldsymbol{S}^* \in \mathbb{Z}_q^{m\times m}$ denote the submatrix of these m linearly independent rows. Consider fixing $n-m$ columns of $\boldsymbol{S}$ of $\boldsymbol{A}$ corresponding to the remaining $m-n$ rows of $\boldsymbol{S}$ and only consider a $n' \times m$ submatrix $\boldsymbol{A}^*$ that correspond to $\boldsymbol{S}^*$. The matrix $\boldsymbol{A}^*$ has row vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{n'} \in \mathbb{Z}_q^m$.

Then $\boldsymbol{AS} = [\boldsymbol{a}_1^T\boldsymbol{S}^* + \boldsymbol{u}_1, \ldots, \boldsymbol{a}_{n'}^T\boldsymbol{S}^* + \boldsymbol{u}_{n'}]$ where $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n'} \in \mathbb{Z}_q^m$ are arbitrary vectors that depend on the values of the fixed rows. For any $i \in [n']$, as $\boldsymbol{S}^*$ is full-rank, there is a bijection between vectors $\boldsymbol{a}_i$ and $\boldsymbol{a}_i^T\boldsymbol{S}^*+\boldsymbol{u}_i$. As $\boldsymbol{a}_i$ is distributed uniformly over $\mathbb{Z}_q^m$, so is $\boldsymbol{a}_i^T\boldsymbol{S}^*$ and $\boldsymbol{a}_i^T\boldsymbol{S}^*+\boldsymbol{u}_i$. This in turn implies that $\boldsymbol{AS}$ is distributed uniformly over all possible $n' \times m$ matrices.

The above result holds true for every possible fixing of the $n-m$ columns of $\boldsymbol{A}$ corresponding to the rows not in $\boldsymbol{S}^*$ and therefore holds true for the uniform distribution over these values as well. $\square$

Fact 1. *For any $x > 0$ it holds that $\exp(-x) > 1-x$.*

Fact 2. *Let q be a prime. For any $0 < x < 1/q$ it holds that $1-x > \exp(-qx)$.*

Lemma 17 (**Piling-up lemma**)**.** *For any $0 < \mu < 1/2$ and any integer n, given n random variables $X_1, \cdots, X_n$ i.i.d. to Ber_μ, it holds that $\Pr\left(\bigoplus_{i=1}^n X_i = 0\right) = 1/2 + (1-2\mu)^n/2$.*

Lemma 18 (**Statistical Distance from Uniform**)**.** *Let X be a random variable with support on $\{0,1\}^n$. For $\boldsymbol{a} \in \{0,1\}^n$, let $\hat{f}(\boldsymbol{a})$ be its Fourier coefficient, defined by $\hat{f}(\boldsymbol{a}) = \mathbb{E}_X[(-1)^{\langle X, \boldsymbol{a}\rangle}]$. It holds that*

$$\mathsf{SD}(X, U_n) \le 2^{n-1}\left(\sum_{\boldsymbol{a}\in\{0,1\}^n, \boldsymbol{a}\neq 0} \hat{f}(\boldsymbol{a})^2\right)^{1/2} \tag{19}$$

B Proof of Lemma 4

Lemma 19 (**Restating Lemma 4**)**.** *Let n be an integer, t be an odd integer, $m = n^5$, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in \mathbb{F}_2^n$ be random sparse vectors such that each vector has exactly t ones. Define the random variable m' as follows.*

$$m' = \begin{cases} m & w.p.\ 0.5, \\ m-1 & w.p.\ 0.5. \end{cases}$$

Define the random variable $S = \boldsymbol{v}_1 + \boldsymbol{v}_2 + \cdots + \boldsymbol{v}_{m'} \in \mathbb{F}_2^n$. Then it holds that

$$\mathsf{SD}(U_n, S) \le \mathsf{negl}(n).$$

Proof. To analyze the distribution of S, we leverage tools from discrete Fourier analysis, which help us express the behavior of S in terms of its Fourier coefficients. This approach allows us to bound its statistical distance from the uniform distribution by Lemma 18. Therefore, our task is to bound the summation of its Fourier coefficient.

The Fourier coefficient for $\boldsymbol{a} \in \mathbb{F}_2^n$ is

$$\begin{aligned} \phi(\boldsymbol{a}) &= \mathbb{E}_S[(-1)^{\langle \boldsymbol{a}, S\rangle}] \\ &= \mathbb{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m,m'}[(-1)^{\langle \boldsymbol{a}, \boldsymbol{v}_1+\cdots+\boldsymbol{v}_{m'}\rangle}] \\ &= \mathbb{E}_{m'}\left[\prod_{i=1}^{m'} \mathbb{E}_{\boldsymbol{v}_i}[(-1)^{\langle \boldsymbol{a}, \boldsymbol{v}_i\rangle}]\right] \\ &= \mathbb{E}_{m'}\left[\mathbb{E}_{\boldsymbol{v}}[(-1)^{\langle \boldsymbol{a}, \boldsymbol{v}\rangle}]^{m'}\right], \end{aligned} \tag{20}$$

where the second to last line is due to the fact that $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ are independent and the fact that $\mathbb{E}[XY] = \mathbb{E}[X]\cdot\mathbb{E}[Y]$ if X and Y are independent, and the last line is true since $v_1, \ldots, v_m$ are identically distributed.

If $\boldsymbol{a} = \mathbf{1}$, we have

$$\phi(\boldsymbol{a}) = \mathbb{E}_{m'}[(-1)^{m'}] = 0.5 \cdot (-1) + 0.5 \cdot 1 = 0. \tag{21}$$

If $\boldsymbol{a} = \mathbf{0}$, we have

$$\phi(\boldsymbol{a}) = \mathbb{E}_{m'}[1] = 1. \tag{22}$$

If $\boldsymbol{a} \neq 1$ and $\boldsymbol{a} \neq 0$, by Claim 9, we have

$$\phi(\boldsymbol{a}) = \mathbb{E}_{m'}[(1 - 1/n^3)^{m'}] < (1 - 1/n^3)^{m-1} = 1/2^{\Omega(n^2)} \tag{23}$$

By Lemma 18, it holds that

$$\mathsf{SD}(U_n, S) \leq 2^{n-1} \left(\sum_{\boldsymbol{a} \in \{0,1\}^n, \boldsymbol{a} \neq 0} \phi(\boldsymbol{a})^2 \right)^{1/2} = \mathsf{negl}(n) \tag{24}$$

Claim 9. *For $\boldsymbol{a} \neq 1$ and $\boldsymbol{a} \neq 0$, $\mathbb{E}_{\boldsymbol{v}}[(-1)^{\langle \boldsymbol{a}, \boldsymbol{v} \rangle}] \leq 1 - 1/n^3$.*

Proof. Let $w \in [1, n-1]$ be weight of the binary vector $\boldsymbol{a}$, we have

$$\begin{aligned}
\left| \mathbb{E}_{\boldsymbol{v}}[(-1)^{\langle \boldsymbol{a}, \boldsymbol{v} \rangle}] \right| &= \left| \frac{1}{\binom{n}{t}} \cdot \sum_{i=0}^{w} (-1)^i \cdot \binom{w}{i} \cdot \binom{n-w}{t-i} \right| \\
&= \left| \frac{1}{\binom{n}{t}} \cdot \left(\sum_{i=0}^{w} \binom{w}{2i} \cdot \binom{n-w}{t-2i} - \sum_{i=0}^{w} \binom{w}{2i+1} \cdot \binom{n-w}{t-(2i+1)} \right) \right| \\
&\leq \frac{1}{\binom{n}{t}} \cdot \max \left(\sum_{i=0}^{w} \binom{w}{2i} \cdot \binom{n-w}{t-2i}, \sum_{i=0}^{w} \binom{w}{2i+1} \cdot \binom{n-w}{t-(2i+1)} \right) \\
&\leq 1 - 1/n^2.
\end{aligned} \tag{25}$$

Note that $\binom{w}{i} \cdot \binom{n-w}{t-i}$ is nonzero if and only if $\max(0, w+t-n) \leq i \leq \min(w, t)$.

Case 1: $\min(w,t) - \max(0, w+t-n) + 1$ is even. Denote

$$A = 1/\binom{n}{t} \cdot \sum_{i=0}^{w} \binom{w}{2i} \cdot \binom{n-w}{t-2i}$$

$$B = 1/\binom{n}{t} \cdot \sum_{i=0}^{w} \binom{w}{2i+1} \cdot \binom{n-w}{t-(2i+1)}.$$

Note that there is an equal number of non-zero terms in A and in B, since the total number of non-zero terms is even and we allocate them evenly to A and B. By Claim 10 and Claim 11, we have

$$\begin{cases} A, B > 0 \\ A + B = 1 \\ \frac{1}{n^2} \leq \frac{A}{B} \leq n^2 \end{cases} \tag{26}$$

Assume without loss of generality that $A > B$. Plugging $A = n^2 B$ into $A + B = 1$, we have $B = \frac{1}{n^2}$. Therefore, $\max(A, B) = A = 1 - \frac{1}{n^2+1}$, and

$$\left|\mathbb{E}_{\boldsymbol{v}}[(-1)^{\langle \boldsymbol{a},\boldsymbol{v}\rangle}]\right| \leq \max(A, B) \leq 1 - \frac{1}{n^2+1}. \tag{27}$$

Case 2: $\min(w,t) - \max(0, w+t-n) + 1$ is odd. Without loss of generality, assume $\min(w,t)$ is odd. We define A in the same way as in Case 1. For B, we exclude the last non-zero term, i.e., C, to make it have the same number of non-zero terms as A. Formally, denote

$$A = 1/\binom{n}{t} \cdot \sum_{i\in[\max(0,w+t-n),\min(w,t)],i \text{ even}} \binom{w}{i} \cdot \binom{n-w}{t-i}$$

$$B = 1/\binom{n}{t} \cdot \sum_{i\in[\max(0,w+t-n),\min(w,t)-1],i \text{ odd}} \binom{w}{i} \cdot \binom{n-w}{t-i}$$

$$C = 1/\binom{n}{t} \cdot \binom{w}{\min(w,t)} \cdot \binom{n-w}{t-\min(w,t)}$$

By Claim 10, Claim 11 and Claim 12, we have

$$\begin{cases} A, B, C > 0 \\ A + (B + C) = 1 \\ \frac{1}{n^2} \leq \frac{A}{B} \leq n^2 \\ \frac{1}{2} \leq C \leq \frac{n-1}{n} \end{cases} \tag{28}$$

Assume without loss of generality that $A > B$. Plugging $A = n^2 B$ into $A + B + C = 1$, we have $B = \frac{1}{n^2+1} \cdot (1 - C)$. Therefore, $\max(A, B) = A = (1 - C) \cdot (1 - \frac{1}{n^2+1})$, and

$$\begin{aligned} \left|\mathbb{E}_{\boldsymbol{v}}[(-1)^{\langle \boldsymbol{a},\boldsymbol{v}\rangle}]\right| &\leq \max(A, B + C) \\ &\leq \max(A, B) + C \\ &\leq (1 - C) \cdot \left(1 - \frac{1}{n^2}\right) + C \\ &= 1 - \frac{1}{n^2+1} + \frac{C}{n^2+1} \\ &\leq 1 - \frac{1}{n^3} \end{aligned} \tag{29}$$

Claim 10.

$$\sum_{i=0}^{w} \cdot \binom{w}{2i} \cdot \binom{n-w}{t-2i} + \sum_{i=0}^{w} \cdot \binom{w}{2i+1} \cdot \binom{n-w}{t-(2i+1)} = \binom{n}{t}$$

Proof. It follows from the identity $\binom{n}{t} = \sum_{i=0}^{w} \binom{w}{i} \cdot \binom{n-w}{t-i}$, by partitioning i into even and odd cases. □

Claim 11. *For* $\max(1, w+t-n+1) \le i \le \min(w,t)$*, it holds that*

$$\frac{1}{n^2} \le \frac{\binom{w}{i} \cdot \binom{n-w}{t-i}}{\binom{w}{i-1} \cdot \binom{n-w}{t-(i-1)}} \le n^2$$

Proof. We have

$$\frac{\binom{w}{i} \cdot \binom{n-w}{t-i}}{\binom{w}{i-1} \cdot \binom{n-w}{t-(i-1)}} = \frac{w-i+1}{i} \cdot \frac{t-i+1}{n-w-t+i} \tag{30}$$

It holds that for $\max(1, w+t-n+1) \le i \le \min(w,t)$,

$$\frac{1}{n} \le \frac{w-i+1}{i} \le n \tag{31}$$

It holds that for $\max(1, w+t-n+1) \le i \le \min(w,t)$,

$$\frac{1}{n} \le \frac{t-i+1}{n-w-t+i} \le n \tag{32}$$

Combining the above inequalities concludes the proof. □

Claim 12. *Let* $t = \omega(1)$ *be slightly super constant,* $1 \le w \le n-1$*, it holds that for sufficiently large* n*,*

$$\frac{1}{2} \le \frac{\binom{w}{\min(w,t)} \cdot \binom{n-w}{t-\min(w,t)}}{\binom{n}{t}} \le \frac{n-1}{n}$$

Proof. If $w < t$, we have

$$\binom{w}{\min(w,t)} \cdot \binom{n-w}{t-\min(w,t)} = \binom{n-w}{t-w} \le \binom{n-1}{t-w} \le \binom{n-1}{t}, \tag{33}$$

where the last inequality is due to $t < (n-1)/2$ for sufficiently large n.

If $w \ge t$, we have

$$\binom{w}{\min(w,t)} \cdot \binom{n-w}{t-\min(w,t)} = \binom{w}{t} \le \binom{n-1}{t}. \tag{34}$$

It also holds that

$$\frac{1}{2} \le \binom{n-1}{t} \Big/ \binom{n}{t} = \frac{n-t}{n} \le \frac{n-1}{n} \tag{35}$$

Combining the above inequalities concludes the proof.

□

□

□

C Proof of Theorem 5

Theorem 14 (**Restating Theorem 5**)**.** *Let n be a security parameter. Let $\mathcal{F} = \mathcal{F}_{n,\mu,\tau}$ be the function family defined in Construction 1. Let $M = (-\log \mu)^\tau$. Then, for any efficient weak PRF adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, Q) \leq M\tau \cdot \mathsf{Adv}_{\mathsf{LPN}}(n, \mu, Q) + Q \cdot \Big((\mu M)^\tau + 2^{-(n-M\tau-1)}\Big).$$

Proof. To aid the proof we first define a sequence of auxiliary functions $\tilde{\mathcal{F}}^{(\alpha^*)}$ for $1 \leq \alpha^* \leq \tau$. The function $\tilde{\mathcal{F}}^{(\alpha^*)}$ is defined identically as the weak PRF function $\mathcal{F}$ except that it set $\boldsymbol{y}^{(\alpha^*)} \leftarrow U_{m_{\alpha^*-1}}$ and starts from the α^*-th round.

Formally, on input $x \in \{0,1\}^n$, $\tilde{F}^{(\alpha^*)}_{\boldsymbol{S}}$ is defined as:

1. $\boldsymbol{a} \leftarrow x \in \mathbb{F}_2^n$.
2. $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}^{(\alpha)}_{(1)}, \ldots, \boldsymbol{s}^{(\alpha)}_{(M_\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$ for $\alpha \in [\tau]$ *in parallel.*
3. $\boldsymbol{r}^{(\alpha)} \leftarrow \boldsymbol{S}^{(\alpha)} \cdot \boldsymbol{a} \in \mathbb{F}_2^{M_\alpha}$ for $\alpha \in [\tau]$ *in parallel.*
4. $\underline{\boldsymbol{y}^{(\alpha^*-1)} \leftarrow U_{M_{\alpha^*-1}} \in \mathbb{F}_2^{M_{\alpha^*-1}}}$.
5. For $\underline{\alpha = \alpha^*, \ldots, \tau}$ *in sequential*:
   (a) $\boldsymbol{e}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$ (Definition 14).
   (b) $\boldsymbol{y}^{(\alpha)} \leftarrow \boldsymbol{r}^{(\alpha)} + \boldsymbol{e}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.
6. Output $y \leftarrow \boldsymbol{y}^{(\tau)} \in \mathbb{F}_2$.

We proceed via a sequence of games.

Game H_0. This is the real attack game: we choose $\boldsymbol{S} \leftarrow \mathbb{F}_2^{M\tau \times n}$ and upon request sample $x \leftarrow \{0,1\}^n$ and output $V(x, F_{\boldsymbol{S}})$.

Game H_1. This is the intermediate game where we choose $\boldsymbol{S} \leftarrow \mathbb{F}_2^{M\tau \times n}$ and upon request sample $x \leftarrow \{0,1\}^n$ and output $V(x, \tilde{F}^{(1)}_{\boldsymbol{S}})$.

For $1 \leq \alpha^* \leq \tau + 1$, we define the following games.

Game H_{2,α^*}. This is the intermediate game where we choose $\boldsymbol{S} \leftarrow \mathbb{F}_2^{M\tau \times n}$ and upon request sample $x \leftarrow \{0,1\}^n$ and output $V(x, \tilde{F}^{(\alpha^*)}_{\boldsymbol{S}})$.

Game H_3. This is the ideal attack game where upon request we sample $x \leftarrow \{0,1\}^n$, $u \leftarrow \mathbb{F}_2$ and output (x, u).

Lemma 20. *For any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_0, H_1) \leq Q \cdot \Big((\mu M)^\tau + 2^{-(n-M\tau-1)}\Big) \tag{36}$$

Proof. We consider the statistical distance between the output of H_0 and H_1.

$$\begin{aligned}
&\mathsf{SD}\left(V\left(\boldsymbol{x}^{H_0}, F_{\boldsymbol{S}^{H_0}}\right), V\left(\boldsymbol{x}^{H_1}, \tilde{F}^{(1)}_{\boldsymbol{S}^{H_1}}\right)\right)\\
=&\frac{1}{2}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathbb{F}_2^{Q\times n}\times\mathbb{F}_2^Q}\left|\Pr_{\boldsymbol{x}^{H_0}}\left[\boldsymbol{x}^{H_0}=\boldsymbol{x}\right]\cdot\left(\Pr_{\boldsymbol{S}^{H_0}}\left[V(\boldsymbol{x},F_{\boldsymbol{S}^{H_0}})=(\boldsymbol{x},\boldsymbol{y})\right]-\Pr_{\boldsymbol{S}^{H_1},\tilde{\boldsymbol{y}}^{(0)}}\left[V(\boldsymbol{x},\tilde{F}^{(1)}_{\boldsymbol{S}^{H_1}})=(\boldsymbol{x},\boldsymbol{y})\right]\right)\right|\\
=&\frac{1}{2}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathbb{F}_2^{Q\times n}\times\mathbb{F}_2^Q}\Pr_{\boldsymbol{x}^{H_0}}\left[\boldsymbol{x}^{H_0}=\boldsymbol{x}\right]\\
&\cdot\left|\sum_{\boldsymbol{S}\in\mathbb{F}_2^{M\tau\times n}}\Pr_{\boldsymbol{S}^{H_0}}\left[\boldsymbol{S}^{H_0}=\boldsymbol{S}\right]\cdot\left(\Pr[V(\boldsymbol{x},F_{\boldsymbol{S}})=(\boldsymbol{x},\boldsymbol{y})]-\Pr_{\tilde{\boldsymbol{y}}^{(0)}}[V(\boldsymbol{x},F^{(1)}_{\boldsymbol{S}})=(\boldsymbol{x},\boldsymbol{y})]\right)\right|\\
=&\frac{1}{2}\sum_{(\boldsymbol{x},\boldsymbol{S})\in\mathbb{F}_2^{Q\times n}\times\mathbb{F}_2^{M\tau\times n}}\Pr_{\boldsymbol{x}^{H_0}}\left[\boldsymbol{x}^{H_0}=\boldsymbol{x}\right]\cdot\Pr_{\boldsymbol{S}^{H_0}}\left[\boldsymbol{S}^{H_0}=\boldsymbol{S}\right]\\
&\cdot\sum_{\boldsymbol{y}\in\mathbb{F}_2^Q}\left|\left(\Pr[V(\boldsymbol{x},F_{\boldsymbol{S}})=(\boldsymbol{x},\boldsymbol{y})]-\Pr_{\tilde{\boldsymbol{y}}^{(0)}}[V(\boldsymbol{x},\tilde{F}^{(1)}_{\boldsymbol{S}})=(\boldsymbol{x},\boldsymbol{y})]\right)\right|
\end{aligned}\tag{37}$$

Since $F_{\boldsymbol{S}}$ is a deterministic function, we have $\Pr[V(\boldsymbol{x},F_{\boldsymbol{S}})=(\boldsymbol{x},\boldsymbol{y})]=1$ if $F_{\boldsymbol{S}}(\boldsymbol{x})=\boldsymbol{y}$, otherwise 0. We then continue as follows.

$$\begin{aligned}
=&\frac{1}{2}\sum_{(\boldsymbol{x},\boldsymbol{S})\in\mathbb{F}_2^{Q\times n}\times\mathbb{F}_2^{M\tau\times n}}\Pr_{\boldsymbol{x}^{H_0}}\left[\boldsymbol{x}^{H_0}=\boldsymbol{x}\right]\cdot\Pr_{\boldsymbol{S}^{H_0}}\left[\boldsymbol{S}^{H_0}=\boldsymbol{S}\right]\\
&\quad\cdot\left(1-\Pr_{\tilde{\boldsymbol{y}}^{(0)}}[V(\boldsymbol{x},\tilde{F}^{(1)}_{\boldsymbol{S}})=V(\boldsymbol{x},F_{\boldsymbol{S}})]\right)+\Pr_{\tilde{\boldsymbol{y}}^{(0)}}[V(\boldsymbol{x},\tilde{F}^{(1)}_{\boldsymbol{S}})\neq V(\boldsymbol{x},F_{\boldsymbol{S}})])\\
=&\sum_{(\boldsymbol{x},\boldsymbol{S})\in\mathbb{F}_2^{Q\times n}\times\mathbb{F}_2^{M\tau\times n}}\Pr_{\boldsymbol{x}^{H_0}}\left[\boldsymbol{x}^{H_0}=\boldsymbol{x}\right]\cdot\Pr_{\boldsymbol{S}^{H_0}}\left[\boldsymbol{S}^{H_0}=\boldsymbol{S}\right]\cdot\Pr_{\tilde{\boldsymbol{y}}^{(0)}}\left[V(\boldsymbol{x},\tilde{F}^{(1)}_{\boldsymbol{S}})\neq V(\boldsymbol{x},F_{\boldsymbol{S}})\right]\\
=&\Pr_{(\boldsymbol{x},\boldsymbol{S},\tilde{\boldsymbol{y}}^{(0)})\leftarrow\mathbb{F}_2^{Q\times n}\times\mathbb{F}_2^{M\tau\times n}\times\mathbb{F}_2^{Q\times M}}[V(\boldsymbol{x},F_{\boldsymbol{S}})\neq V(\boldsymbol{x},\tilde{F}^{(1)}_{\boldsymbol{S}})]\\
\leq&Q\cdot\Pr_{(x,\boldsymbol{S},\tilde{\boldsymbol{y}}^{(0)})\leftarrow\mathbb{F}_2^{n}\times\mathbb{F}_2^{M\tau\times n}\times\mathbb{F}_2^{M}}[F_{\boldsymbol{S}}(x)\neq\tilde{F}^{(1)}_{\boldsymbol{S}}(x)]\\
\leq&Q\cdot\left((\mu M)^\tau+2^{-(n-M\tau-1)}\right)
\end{aligned}\tag{38}$$

where the second-to-last line follows from the union bound, and the last inequality is due to Claim 13.

Claim 13.

$$\Pr_{(x,\boldsymbol{S},\boldsymbol{y}^{(0)})\leftarrow\{0,1\}^n\times\mathbb{F}_2^{M\tau\times n}\times\mathbb{F}_2^{M}}[F_{\boldsymbol{S}}(x)\neq\tilde{F}^{(1)}_{\boldsymbol{S}}(x)]\leq(\mu M)^\tau+2^{-(n-M\tau-1)}$$

Proof. We consider the transcript of the intermediate results during the evaluation of $F_{\boldsymbol{S}}(x)$ and $\tilde{F}^{(1)}_{\boldsymbol{S}}(x)$. We use X to denote the random variable during the evaluation of $F_{\boldsymbol{S}}(x)$ (e.g., $\boldsymbol{y}$, $\boldsymbol{z}$). We use $\tilde{X}$ to denote the random variable during the evaluation of $\tilde{F}_{\boldsymbol{S}}(x)$ (e.g., $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{z}}$).

Claim 14. *Conditioned on $\boldsymbol{S}\in\mathbb{F}_2^{M\tau\times n}$ being full rank, it holds that*

$$(\boldsymbol{r}^{(1)},\ldots,\boldsymbol{r}^{(\tau)})$$

is distributed uniformly at random for random x.

Proof. Since $x \in \mathbb{F}_2^n$ is uniformly at random and $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$ is full rank, it follows that $\boldsymbol{R} = \boldsymbol{S} \cdot x \in \mathbb{F}_2^{M\tau}$ is also uniformly at random By Lemma 16, so is $(\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(\tau)})$, which is a subset of $\boldsymbol{R}$. □

Claim 15. *Conditioned on $\boldsymbol{S} \in \mathbb{F}_2^{M\tau\times n}$ being full rank, for $\alpha \in [\tau]$, $\boldsymbol{r}^{(\alpha)}$ is independent of $(\boldsymbol{e}^{(\alpha')}, \tilde{\boldsymbol{e}}^{(\alpha')})$ for $\alpha' \leq \alpha$.*

Proof. For $\alpha' \leq \alpha$, $(\boldsymbol{e}^{(\alpha')}, \tilde{\boldsymbol{e}}^{(\alpha')})$ only depends on $\boldsymbol{y}^{(0)}$, $\boldsymbol{r}^{(\alpha'')}$ for $\alpha'' < \alpha'$. Since $\boldsymbol{r}^{(\alpha)}$ is independent of $\boldsymbol{y}^{(0)}$, and $\boldsymbol{r}^{(\alpha'')}$ by Claim 14. Therefore, $\boldsymbol{r}^{(\alpha)}$ is independent of $(\boldsymbol{e}^{(\alpha')}, \tilde{\boldsymbol{e}}^{(\alpha')})$.

□

Observe that if $F_{\boldsymbol{S}}(x) \neq \tilde{F}_{\boldsymbol{S}}^{(1)}(x)$, then it must hold that for all $\alpha \in [\tau]$, $\boldsymbol{y}^{(\alpha)} \neq \tilde{\boldsymbol{y}}^{(\alpha)}$ since round iteration is deterministic. For $\alpha = 1, \ldots, \tau$, we consider the probability that $\boldsymbol{y}^{(\alpha)} = \tilde{\boldsymbol{y}}^{(\alpha)}$ conditioned on $\boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}$. Conditioned on $\boldsymbol{S} \in \mathbb{F}_2^{M\tau\times n}$ being full rank, which happens except with probability $2^{-(n-M\tau-1)}$ by Lemma 15, we have

$$\begin{aligned}
&\Pr_{x,\boldsymbol{S},\boldsymbol{y}^{(0)}}[\boldsymbol{y}^{(\alpha)} = \tilde{\boldsymbol{y}}^{(\alpha)} \mid \boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}, \boldsymbol{S} \text{ full rank}] \\
&= \Pr[\mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha-1)}) = \mathsf{S\text{-}Ber}(\tilde{\boldsymbol{y}}^{(\alpha-1)}) \mid \boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}, \boldsymbol{S} \text{ full rank}] \\
&\geq \Pr[\mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha-1)}) = \mathsf{S\text{-}Ber}(\tilde{\boldsymbol{y}}^{(\alpha-1)}) \mid \boldsymbol{S} \text{ full rank}] \\
&= \Pr[\mathsf{S\text{-}Ber}(\boldsymbol{r}^{(\alpha-1)} + \boldsymbol{e}^{(\alpha-1)}) = \mathsf{S\text{-}Ber}(\boldsymbol{r}^{(\alpha-1)} + \tilde{\boldsymbol{e}}^{(\alpha-1)}) \mid \boldsymbol{S} \text{ full rank}] \\
&\geq (1-2\mu)^{M_{\alpha-1}/\ell} \geq (1-2\mu)^{M/2} \geq 1 - \mu M,
\end{aligned} \tag{39}$$

where the last line is due to Lemma 3, Claim 15, and Claim 14. Therefore, it holds that

$$\begin{aligned}
&\Pr_{x,\boldsymbol{S},\boldsymbol{y}^{(0)}}[F_{\boldsymbol{S}}(x) \neq \tilde{F}_{\boldsymbol{S}}^{(1)}(x) \mid \boldsymbol{S} \text{ full rank}] \\
&= \Pr\left[\bigcap_{\alpha=1}^{\tau} \boldsymbol{y}^{(\alpha)} \neq \tilde{\boldsymbol{y}}^{(\alpha)} \mid \boldsymbol{S} \text{ full rank}\right] \\
&= \Pr[\boldsymbol{y}^{(0)} \neq \tilde{\boldsymbol{y}}^{(0)} \mid \boldsymbol{S} \text{ full rank}] \cdot \prod_{\alpha=2}^{\tau} \Pr[\boldsymbol{y}^{(\alpha)} \neq \tilde{\boldsymbol{y}}^{(\alpha)} \mid \boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}, \boldsymbol{S} \text{ full rank}] \\
&\leq (\mu M)^{\tau}
\end{aligned} \tag{40}$$

□

□

By definition, the hybrid game $H_{2,1}$ corresponds to the game H_1, and the hybrid game $H_{2,\tau+1}$ corresponds to the game H_3. We show that each consecutive hybrid experiments H_{2,α^*} and H_{2,α^*+1} for $\alpha^* = 1, 2, \ldots, \tau$ are computationally indistinguishable.

Lemma 21. *For any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_{2,\alpha^*}, H_{2,\alpha^*+1}) \leq M \cdot \mathsf{Adv}_{\mathsf{LPN}}(n, \mu, Q) \tag{41}$$

for all $1 \leq \alpha^* \leq \tau$.

Proof. Let $\mathcal{A}$ be a distinguisher for the games H_{2,α^*} and H_{2,α^*+1}. We construct an algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ to break the $\mathsf{LPN}_{n,\mu,Q}$ problem (up to a multiplicative factor of M in the advantage). Algorithm $\mathcal{A}'$ proceeds as follows:

1. At the start of the game, algorithm $\mathcal{A}'$ samples secret key $\boldsymbol{S} = (\boldsymbol{s}_{(i)}^{(\alpha)})_{\alpha \in [\tau], i \in [M]} \leftarrow \mathbb{F}_2^{M\tau \times n}$.
2. Whenever $\mathcal{A}$ requests an output, Algorithm $\mathcal{A}'$ does the following.
   (a) It queries the LPN problem oracle with M secrets to receive $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}_2^n \times \mathbb{F}_2^M$.
   (b) It set $\boldsymbol{b} \leftarrow (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{M_{\alpha^*}}) \in \mathbb{F}_2^{M_{\alpha^*}}$.
   (c) $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}_{(1)}^{(\alpha)}, \ldots, \boldsymbol{s}_{(M_\alpha)}^{(\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$ for $\alpha \in [\tau]$ *in parallel.*
   (d) $\boldsymbol{r}^{(\alpha)} \leftarrow \boldsymbol{S}^{(\alpha)} \cdot \boldsymbol{a} \in \mathbb{F}_2^{M_\alpha}$ for $\alpha \in [\tau]$ *in parallel.*
   (e) $\underline{\boldsymbol{y}^{(\alpha^*)} \leftarrow \boldsymbol{b} \in \mathbb{F}_2^{M_{\alpha^*}}}$.
   (f) For $\underline{\alpha = \alpha^* + 1, \ldots, \tau}$ *in sequential*:
      i. $\boldsymbol{e}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$ (Definition 14).
      ii. $\boldsymbol{y}^{(\alpha)} \leftarrow \boldsymbol{r}^{(\alpha)} + \boldsymbol{e}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.
   (g) Sends $y \leftarrow \boldsymbol{y}^{(\tau)} \in \mathbb{F}_2$ to $\mathcal{A}$.
3. At the end of the game, when $\mathcal{A}$ returns its guess $\beta \in \{0, 1\}$, algorithm $\mathcal{A}'$ returns β.

We now show that depending on whether $\mathcal{A}'$ is receiving LPN samples or uniform samples from the $\mathsf{LPN}_{n,\mu,Q}$ oracle with m secrets, it simulates either H_{2,α^*} or H_{2,α^*+1}.

- Suppose that $\mathcal{A}'$ is receiving LPN samples for M secrets $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_M) \leftarrow \mathbb{F}_2^{M \times n}$. Observe that in H_{2,α^*}, we have

$$\begin{aligned} \boldsymbol{y}^{(\alpha^*)} &= \boldsymbol{r}^{(\alpha^*)} \oplus \boldsymbol{e}^{(\alpha^*)} \\ &= \boldsymbol{S}^{(\alpha^*)} \cdot x \oplus \boldsymbol{e}^{(\alpha^*)} \end{aligned} \tag{42}$$

  where $\boldsymbol{S}^{(\alpha^*)} \in \mathbb{F}_2^{M_{\alpha^*} \times n}$ is uniformly at random, $\boldsymbol{e}^{(\alpha^*)} = \mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha^*)}) = \mathsf{S\text{-}Ber}(U_{M_{\alpha^*-1}})$, which is distributed as $\mathsf{Ber}_{M_{\alpha^*}}$ by Claim 3.

  We then study the behavior of $\mathcal{A}'$. For each of $\mathcal{A}$'s request, algorithm $\mathcal{A}'$ receives $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{S} \cdot \boldsymbol{a} + \boldsymbol{e}) \in \mathbb{F}_2^n \times \mathbb{F}_2^M$ for $\boldsymbol{a} \leftarrow \mathbb{F}_2^n, \boldsymbol{e} \leftarrow \mathsf{Ber}_\mu^M$. It then sets $\boldsymbol{y}^{(\alpha^*)} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{M_{\alpha^*}}) = \boldsymbol{S}' \cdot \boldsymbol{a} + \boldsymbol{e}$, where $\boldsymbol{S}' = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{M_{\alpha^*}}) \in \mathbb{F}_2^{M_{\alpha^*}}$. Therefore, it perfectly simulates $\boldsymbol{y}^{(\alpha^*)}$ in game H_{2,α^*}. The response of $\mathcal{A}'$ for each of $\mathcal{A}$'s oracle queries is distributed exactly as in H_{2,α^*}.

- Suppose that $\mathcal{A}'$ is receiving uniformly random samples. Then, for each of $\mathcal{A}$'s request, algorithm $\mathcal{A}'$ receives $(\boldsymbol{a}, \boldsymbol{u}) \in \mathbb{F}_2^n \times \mathbb{F}_2^{M_{\alpha^*}}$ for $\boldsymbol{u} \leftarrow \mathbb{F}_2^{M_{\alpha^*}}$. It then sets $\boldsymbol{y}^{(\alpha^*)} = \boldsymbol{u}$, which perfectly simulates $\boldsymbol{y}^{(\alpha^*)}$ in game oracle queries is distributed exactly as in H_{2,α^*+1}. Then the response of $\mathcal{A}'$ for each of $\mathcal{A}$'s oracle queries is distributed exactly as in H_{2,α^*+1}.

We have shown that depending on whether $\mathcal{A}'$ is interacting with LPN samples or uniform samples, it simulates the distributions of either H_{2,α^*} or H_{2,α^*+1}. Since $\mathcal{A}$ makes at most Q queries, $\mathcal{A}'$ will receive at most Q samples from the LPN oracle with M secrets. Therefore, with the same distinguishing advantage of $\mathcal{A}$ (up to a multiplicative factor of M), algorithm $\mathcal{A}'$ solves the $\mathsf{LPN}_{n,\mu,Q}$ problem. □

This concludes the proof of Theorem 5. □

D Proof of Theorem 6

Theorem 15 (**Restating Theorem 6**). *Under the parameter setting in Theorem 5, Construction 1 has γ-AKH structure (Definition 13) where $\gamma = \mu + \mathsf{negl}(n)$.*

Proof. Denote $\mathcal{X} = \{0,1\}^n$, $\mathcal{K} = \mathbb{F}_2^{M\tau \times n}$. The required (f, h, z) in AKH structure (Definition 13) is specified as follows.

- Let $f \colon \mathcal{X} \to \mathbb{F}_2^n$ be the identity function.
- Let $h \colon \mathcal{K} \to \mathbb{F}_2^n$ be defined as follows. On input $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$:
  1. Parse $\boldsymbol{S} = (\boldsymbol{s}_{(i)}^{(\alpha)})_{\alpha \in [\tau], i \in [M]}$, where $\boldsymbol{s}_{(i)}^{(\alpha)} \in \mathbb{F}_2^n$.
  2. Output $\boldsymbol{s}_{(1)}^{(\tau)} \in \mathbb{F}_2^n$.

  It is easy to see that h is a group homomorphism, as required.
- Let $z \colon \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2$ is the function outputting $e^{(\tau)} \in \mathbb{F}_2$, an intermediate value during the execution of F in Construction 1. By definition, z is as efficient as F.

It follows by definition that for any $\boldsymbol{S} \in \mathcal{K}, x \in \{0,1\}^n$, F can be represented as

$$F(\boldsymbol{S}, x) = \langle f(x), h(\boldsymbol{S}) \rangle + z(\boldsymbol{S}, x). \tag{43}$$

It remains to prove the following lemma.

Lemma 22 (**Vanishing of z**). *It holds that*

$$\Pr_{\boldsymbol{S} \leftarrow \mathcal{K}, x \leftarrow \mathcal{X}}[z(\boldsymbol{S}, x) = 1] \leq \mu + \mathsf{negl}(\lambda).$$

Proof. For random x, conditioned on $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$ (which happens except with probability $2^{-(n-M\tau-1)}$ by Lemma 15), it holds that

$$\begin{aligned} & \boldsymbol{r}^{(\tau-1)} \text{ is uniformly at random (Claim 14) and independent of } \boldsymbol{e}^{(\tau-1)} \text{ (Claim 15)} \\ \implies & \boldsymbol{y}^{(\tau-1)} = \boldsymbol{r}^{(\tau-1)} + \boldsymbol{e}^{(\tau-1)} \text{ is uniformly at random} \\ \implies & e^{(\tau)} = \mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\tau-1)}) \sim \mathsf{Ber}_\mu \text{ (Claim 3)} \end{aligned} \tag{44}$$

It then follows that

$$\begin{aligned} \Pr_{\boldsymbol{S}\leftarrow\mathcal{K}, x\leftarrow\mathcal{X}}[z(\boldsymbol{S}, x) = 1] &\le \Pr[\mathsf{Ber}_\mu = 1] + \Pr_{\boldsymbol{S}\leftarrow\mathcal{K}}[\boldsymbol{S} \text{ not full rank}] \\ &\le \mu + \mathsf{negl}(n). \end{aligned} \tag{45}$$

□

□

E Proof of Theorem 11

Theorem 16 (Restating Theorem 11). *Let n be a security parameter. Let $\mathcal{F} = \mathcal{F}_{n,\mu,m,H,\tau}$ be the function family defined in Construction 3. Let $M = (-\log\mu)^\tau$. Then, for any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, m) \le M\tau \cdot \mathsf{Adv}_{\mathsf{seededLPN}}(n, \mu, m) + m \cdot \left((\mu M)^\tau + 2^{-(n-M\tau-1)} + 2^n/p\right).$$

Proof. Since our proof makes non-black box use of the underlying weak PRF F'. We first provide an unroll version of Construction 3 as follows.

On input $x \in [m]$, $\tilde{F}_{\boldsymbol{S}}^{(\alpha^*)}$ is defined as:

1. Set $x' \leftarrow \mathsf{bit\text{-}decompose}(H(x)) \in \{0,1\}^n$.
2. $\boldsymbol{a} \leftarrow x' \in \mathbb{F}_2^n$.
3. $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}_{(1)}^{(\alpha)}, \dots, \boldsymbol{s}_{(M_\alpha)}^{(\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$ for $\alpha \in [\tau]$ *in parallel.*
4. $\boldsymbol{r}^{(\alpha)} \leftarrow \boldsymbol{S}^{(\alpha)} \cdot \boldsymbol{a} \in \mathbb{F}_2^{M_\alpha}$ for $\alpha \in [\tau]$ *in parallel.*
5. $\boldsymbol{y}^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{F}_2^{M_{\alpha^*-1}}$.
6. For $\alpha = 1, \dots, \tau$ *in sequential*:
   (a) $\boldsymbol{e}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$ (Definition 14).
   (b) $\boldsymbol{y}^{(\alpha)} \leftarrow \boldsymbol{r}^{(\alpha)} + \boldsymbol{e}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.
7. Output $y \leftarrow \boldsymbol{y}^{(\tau)} \in \mathbb{F}_2$.

We then define a sequence of auxiliary functions $\tilde{\mathcal{F}}^{(\alpha^*)}$ for $1 \leq \alpha^* \leq \tau$.

On input $x \in [m]$, $\tilde{F}_{\boldsymbol{S}}^{(\alpha^*)}$ is defined as:

1. Set $x' \leftarrow \mathsf{bit\text{-}decompose}(H(x)) \in \{0,1\}^n$.
2. $\boldsymbol{a} \leftarrow x' \in \mathbb{F}_2^n$.
3. $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}_{(1)}^{(\alpha)}, \ldots, \boldsymbol{s}_{(M_\alpha)}^{(\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$ for $\alpha \in [\tau]$ *in parallel.*
4. $\boldsymbol{r}^{(\alpha)} \leftarrow \boldsymbol{S}^{(\alpha)} \cdot \boldsymbol{a} \in \mathbb{F}_2^{M_\alpha}$ for $\alpha \in [\tau]$ *in parallel.*
5. $\underline{\boldsymbol{y}^{(\alpha^*-1)} \leftarrow U_{M_{\alpha^*-1}} \in \mathbb{F}_2^{M_{\alpha^*-1}}}$.
6. For $\underline{\alpha = \alpha^*, \ldots, \tau}$ *in sequential*:
   (a) $\boldsymbol{e}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$ (Definition 14).
   (b) $\boldsymbol{y}^{(\alpha)} \leftarrow \boldsymbol{r}^{(\alpha)} + \boldsymbol{e}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.
7. Output $y \leftarrow \boldsymbol{y}^{(\tau)} \in \mathbb{F}_2$.

We proceed via a sequence of games.

Game H_0. This is the real PRF attack game: the challenger chooses the public parameters pp, and samples an $F_{\boldsymbol{S}} \leftarrow \mathcal{F}$. The attacker gets the public parameters pp, and oracle access to $F_{\boldsymbol{S}}(\cdot)$.

Game H_1. This game is identical to Game H_0 except that the attacker $\mathcal{A}$ has oracle access to $\tilde{F}_{\boldsymbol{S}}^{(1)}(\cdot)$.

For $1 \leq \alpha^* \leq \tau$, we define the following games.

Game H_{2,α^*}. This game is identical to Game H_0 except that the attacker $\mathcal{A}$ has oracle access to $\tilde{F}_{\boldsymbol{S}}^{(\alpha^*)}(\cdot)$.

Game H_3. This is the ideal PRF game. The challenger chooses the public parameter pp as in Game H_1 and gives it to the attacker. The challenger (inefficiently) samples a truly random function $R : [m] \to \{0,1\}$ and gives the attacker oracle access to $R(\cdot)$.

Lemma 23. *For any adversary $\mathcal{A}$, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_0, H_1) \leq m \cdot \Big((\mu M)^\tau + 2^{-(n - M\tau - 1)} + 2^n/p \Big) \tag{46}$$

Proof. Without loss of generality, we assume $\mathcal{A}$ is deterministic. If for all queries made by $\mathcal{A}$, $F_{\boldsymbol{S}}(\cdot)$ and $\tilde{F}_{\boldsymbol{S}}^{(1)}(\cdot)$ have the same output except with some small probability (over the randomness of secret key $\boldsymbol{S}$, n-wise independent hash function h, and the randomness of $\tilde{F}_{\boldsymbol{S}}^{(1)}(\cdot)$), then the distinguishing advantage of $\mathcal{A}$ is information-theoretically upper bounded by that probability.

Therefore, it suffices to prove the following claim. The lemma then follows by applying a union bound over all $x \in [m]$. It is important to note that we cannot directly apply a union bound only over the number of queries made by $\mathcal{A}$, as the queries are made adaptively, which could depend on previous answers.

Claim 16. *For any $x \in [m]$, it holds that*

$$\Pr[F_{\boldsymbol{S}}(x) \neq \tilde{F}^{(1)}_{\boldsymbol{S}}(x)] \leq (\mu M)^{\tau} + 2^{-(n-M\tau-1)} + 2^n/p$$

where the probability is over the randomness of the secret key $\boldsymbol{S}$, the hash function $h \leftarrow \mathcal{H}$, and the randomness in the evaluation of $\tilde{F}^{(1)}_{\boldsymbol{S}}(\cdot)$.

Proof. We consider the transcript of the intermediate results during the evaluation of $F_{\boldsymbol{S}}(x)$ and $\tilde{F}^{(1)}_{\boldsymbol{S}}(x)$. We use X to denote the random variable during the evaluation of $F_{\boldsymbol{S}}(x)$ (e.g., $\boldsymbol{y}$, $\boldsymbol{z}$). We use $\tilde{X}$ to denote the random variable during the evaluation of $\tilde{F}_{\boldsymbol{S}}(x)$ (e.g., $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{z}}$).

Claim 17. *For any $x \in [m]$, it holds that*

$$\Pr[H(x) \geq \lfloor p/2^n \rfloor \cdot 2^n] \leq 2^n/p$$

over $H \leftarrow \mathcal{H}$.

Proof. The claim follows from $H(x)$ is distributed uniformly over $\mathbb{Z}_p$ by n-wise independence of hash function H. $\square$

Claim 18. *Conditioned on $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$ being full rank, and $H(x) < \lfloor p/2^n \rfloor \cdot 2^n$, it holds that*

$$(\boldsymbol{r}^{(1)}, \dots, \boldsymbol{r}^{(\tau)})$$

is distributed uniformly at random for $H \leftarrow \mathcal{H}$.

Proof. By n-wise independence of hash function H, it holds that for any $x \in [m]$, any $y \leq \lfloor p/2^n \rfloor \cdot 2^n$,

$$\Pr_{H \leftarrow \mathcal{H}}[H(x) = y \mid H(x) \leq \lfloor p/2^n \rfloor \cdot 2^n] = 1/(\lfloor p/2^n \rfloor \cdot 2^n).$$

Therefore, $x' = \textsf{bit-decompose}(H(x))$ is uniformly at random. Since $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$ is full rank, it follows that $\boldsymbol{R} = \boldsymbol{S} \cdot x \in \mathbb{F}_2^{M\tau}$ is also uniformly at random By Lemma 16, so is $(\boldsymbol{r}^{(1)}, \dots, \boldsymbol{r}^{(\tau)})$, which is a subset of $\boldsymbol{R}$. $\square$

Claim 19. *Conditioned on $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$ being full rank and $H(x) < \lfloor p/2^n \rfloor \cdot 2^n$, for $\alpha \in [\tau]$, $\boldsymbol{r}^{(\alpha)}$ is independent of $(\boldsymbol{e}^{(\alpha')}, \tilde{\boldsymbol{e}}^{(\alpha')})$ for $\alpha' \leq \alpha$.*

Proof. For $\alpha' \leq \alpha$, $(\boldsymbol{e}^{(\alpha')}, \tilde{\boldsymbol{e}}^{(\alpha')})$ only depends on $\boldsymbol{y}^{(0)}$, $\boldsymbol{r}^{(\alpha'')}$ for $\alpha'' < \alpha'$. Since $\boldsymbol{r}^{(\alpha)}$ is independent of $\boldsymbol{y}^{(0)}$, and $\boldsymbol{r}^{(\alpha'')}$ by Claim 18. Therefore, $\boldsymbol{r}^{(\alpha)}$ is independent of $(\boldsymbol{e}^{(\alpha')}, \tilde{\boldsymbol{e}}^{(\alpha')})$. $\square$

Observe that if $F_{\boldsymbol{S}}(x) \neq \tilde{F}^{(1)}_{\boldsymbol{S}}(x)$, then it must hold that for all $\alpha \in [\tau]$, $\boldsymbol{y}^{(\alpha)} \neq \tilde{\boldsymbol{y}}^{(\alpha)}$ since round iteration is deterministic. For $\alpha = 1, \dots, \tau$, we consider the probability that $\boldsymbol{y}^{(\alpha)} = \tilde{\boldsymbol{y}}^{(\alpha)}$ conditioned on $\boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}$. Conditioned on $\boldsymbol{S} \in \mathbb{F}_2^{M\tau \times n}$ being full rank (which happens except with probability $2^{-(n-M\tau-1)}$ by Lemma 15) and $H(x) < \lfloor p/2^n \rfloor \cdot 2^n$ (which happens except with probability $2^n/p$ by Claim 16), we have

$$\begin{aligned}
&\Pr_{x,\boldsymbol{S},\boldsymbol{y}^{(0)}}[\boldsymbol{y}^{(\alpha)} = \tilde{\boldsymbol{y}}^{(\alpha)} \mid \boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}] \\
=&\Pr[\mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha-1)}) = \mathsf{S\text{-}Ber}(\tilde{\boldsymbol{y}}^{(\alpha-1)}) \mid \boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}] \\
\geq&\Pr[\mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha-1)}) = \mathsf{S\text{-}Ber}(\tilde{\boldsymbol{y}}^{(\alpha-1)})] \\
=&\Pr[\mathsf{S\text{-}Ber}(\boldsymbol{r}^{(\alpha-1)} + \boldsymbol{e}^{(\alpha-1)}) = \mathsf{S\text{-}Ber}(\boldsymbol{r}^{(\alpha-1)} + \tilde{\boldsymbol{e}}^{(\alpha-1)})] \\
\geq&(1-\mu)^{2M_{\alpha-1}/\ell} \geq (1-\mu)^M \geq 1 - \mu M,
\end{aligned} \tag{47}$$

where the last line is due to Lemma 3, Claim 18, and Claim 19. Therefore, it holds that

$$\begin{aligned}
&\Pr_{x,\boldsymbol{S},\boldsymbol{y}^{(0)}}[F_{\boldsymbol{S}}(x) \neq \tilde{F}^{(1)}_{\boldsymbol{S}}(x)] \\
=&\Pr\left[\bigcap_{\alpha=1}^{\tau} \boldsymbol{y}^{(\alpha)} \neq \tilde{\boldsymbol{y}}^{(\alpha)}\right] \\
=&\Pr[\boldsymbol{y}^{(0)} \neq \tilde{\boldsymbol{y}}^{(0)}] \cdot \prod_{\alpha=2}^{\tau} \Pr[\boldsymbol{y}^{(\alpha)} \neq \tilde{\boldsymbol{y}}^{(\alpha)} \mid \boldsymbol{y}^{(\alpha-1)} \neq \tilde{\boldsymbol{y}}^{(\alpha-1)}] \\
\leq&(\mu M)^\tau
\end{aligned} \tag{48}$$

□

This concludes the proof of *Lemma* 23. □

By definition, the hybrid game $H_{2,1}$ corresponds to the game H_1, and the hybrid game $H_{2,\tau+1}$ corresponds to the game H_3. We show that each consecutive hybrid experiments H_{2,α^*} and H_{2,α^*+1} for $\alpha^* = 1, 2, \ldots, \tau$ are computationally indistinguishable.

Lemma 24. *For any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_{2,\alpha^*}, H_{2,\alpha^*+1}) \leq M \cdot \mathsf{Adv}_{\mathsf{seededLPN}}(n, \mu, m) \tag{49}$$

for all $1 \leq \alpha^* \leq \tau$.

Proof. Let $\mathcal{A}$ be a distinguisher for the games H_{2,α^*} and H_{2,α^*+1}. We construct an algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ to break the $\mathsf{seededLPN}_{n,\mu,m}$ problem (up to a multiplicative factor of m in the advantage). Algorithm $\mathcal{A}'$ proceeds as follows:

1. At the start of the game, algorithm $\mathcal{A}'$ samples secret key $\boldsymbol{S} = (\boldsymbol{s}^{(\alpha)}_{(i)})_{\alpha\in[\tau],i\in[M]} \leftarrow \mathbb{F}_2^{M\tau\times n}$.
2. Whenever $\mathcal{A}$ requests an output with input $x \in [m]$, Algorithm $\mathcal{A}'$ does the following.
   (a) It sends x to the $\mathsf{seededLPN}$ problem oracle with M secrets to receive the the x-th sample $(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{F}_2^n \times \mathbb{F}_2^M$, where $\boldsymbol{a} = \mathsf{bit\text{-}decompose}(H(x))$.
   (b) It set $\boldsymbol{b} \leftarrow (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{M_{\alpha^*}}) \in \mathbb{F}_2^{M_{\alpha^*}}$.

(c) $\boldsymbol{S}^{(\alpha)} \leftarrow (\boldsymbol{s}^{(\alpha)}_{(1)}, \ldots, \boldsymbol{s}^{(\alpha)}_{(M_\alpha)}) \in \mathbb{F}_2^{M_\alpha \times n}$ for $\alpha \in [\tau]$ *in parallel.*

(d) $\boldsymbol{r}^{(\alpha)} \leftarrow \boldsymbol{S}^{(\alpha)} \cdot \boldsymbol{a} \in \mathbb{F}_2^{M_\alpha}$ for $\alpha \in [\tau]$ *in parallel.*

(e) $\underline{\boldsymbol{y}^{(\alpha^*)} \leftarrow \boldsymbol{b} \in \mathbb{F}_2^{M_{\alpha^*}}}$.

(f) For $\underline{\alpha = \alpha^* + 1, \ldots, \tau}$ *in sequential*:

  i. $\boldsymbol{e}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}_\mu(\boldsymbol{y}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha}$ (Definition 14).

  ii. $\boldsymbol{y}^{(\alpha)} \leftarrow \boldsymbol{r}^{(\alpha)} + \boldsymbol{e}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha}$.

(g) Sends $y \leftarrow \boldsymbol{y}^{(\tau)} \in \mathbb{F}_2$ to $\mathcal{A}$.

3. At the end of the game, when $\mathcal{A}$ returns its guess $\beta \in \{0, 1\}$, algorithm $\mathcal{A}'$ returns β.

We now show that depending on whether $\mathcal{A}'$ is receiving $\mathsf{seededLPN}$ samples or uniform samples from the $\mathsf{seededLPN}_{n,\mu,N}$ oracle with M secrets, it simulates either H_{2,α^*} or H_{2,α^*+1}.

- Suppose that $\mathcal{A}'$ is receiving $\mathsf{seededLPN}$ samples for M secrets $\boldsymbol{S} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_M) \leftarrow \mathbb{F}_2^{M \times n}$. Observe that in H_{2,α^*}, we have

$$\begin{aligned} \boldsymbol{y}^{(\alpha^*)} &= \boldsymbol{r}^{(\alpha^*)} \oplus \boldsymbol{e}^{(\alpha^*)} \\ &= \boldsymbol{S}^{(\alpha^*)} \cdot \mathsf{bit\text{-}decompose}(H(x)) \oplus \boldsymbol{e}^{(\alpha^*)} \end{aligned} \tag{50}$$

  where $\boldsymbol{S}^{(\alpha^*)} \in \mathbb{F}_2^{M_{\alpha^*} \times n}$ is uniformly at random, $\boldsymbol{e}^{(\alpha^*)} = \mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\alpha^*)}) = \mathsf{S\text{-}Ber}(U_{M^{\alpha^*-1}})$, which is distributed as $\mathsf{Ber}_{M^{\alpha^*}}$ by Claim 3.

  We then study the behavior of $\mathcal{A}'$. For each of $\mathcal{A}$'s request, algorithm $\mathcal{A}'$ receives $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{S} \cdot \boldsymbol{a} + \boldsymbol{e}) \in \mathbb{F}_2^n \times \mathbb{F}_2^M$ for $\boldsymbol{a} = \mathsf{bit\text{-}decompose}(H(x)), \boldsymbol{e} \leftarrow \mathsf{Ber}_\mu^M$. It then sets $\boldsymbol{y}^{(\alpha^*)} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{M_{\alpha^*}}) = \boldsymbol{S}' \cdot \boldsymbol{a} + \boldsymbol{e}$, where $\boldsymbol{S}' = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{M_{\alpha^*}}) \in \mathbb{F}_2^{M_{\alpha^*}}$. Therefore, it perfectly simulates $\boldsymbol{y}^{(\alpha^*)}$ in game H_{2,α^*}. The response of $\mathcal{A}'$ for each of $\mathcal{A}$'s oracle queries is distributed exactly as in H_{2,α^*}.

- Suppose that $\mathcal{A}'$ is receiving uniformly random samples. Then, for each of $\mathcal{A}$'s request, algorithm $\mathcal{A}'$ receives $(\boldsymbol{a}, \boldsymbol{u}) \in \mathbb{F}_2^n \times \mathbb{F}_2^{M_{\alpha^*}}$ for $\boldsymbol{u} \leftarrow \mathbb{F}_2^{M_{\alpha^*}}$. It then sets $\boldsymbol{y}^{(\alpha^*)} = \boldsymbol{u}$, which perfectly simulates $\boldsymbol{y}^{(\alpha^*)}$ in game oracle queries is distributed exactly as in H_{2,α^*+1}. Then the response of $\mathcal{A}'$ for each of $\mathcal{A}$'s oracle queries is distributed exactly as in H_{2,α^*+1}.

We have shown that depending on whether $\mathcal{A}'$ is interacting with $\mathsf{seededLPN}$ samples or uniform samples, it simulates the distributions of either H_{2,α^*} or H_{2,α^*+1}. Since $\mathcal{A}$ makes at most Q queries, $\mathcal{A}'$ will receive at most Q samples from the $\mathsf{seededLPN}$ oracle with M secrets. Therefore, with the same distinguishing advantage of $\mathcal{A}$ (up to a multiplicative factor of M), algorithm $\mathcal{A}'$ solves the $\mathsf{seededLPN}_{n,\mu,m}$ problem. □

This concludes the proof of Theorem 11. □

F Proof of Theorem 12

Theorem 17 (**Restating Theorem 12**)**.** *Under the parameter setting in Theorem 11, Construction 3 has γ-AKH structure (Definition 13) where $\gamma = \mu + \mathsf{negl}(n)$.*

Proof. Denote $\mathcal{X} = [m]$, $\mathcal{K} = \mathbb{F}_2^{M\tau \times n}$. The required (f, h, z) in AKH structure (Definition 13) is specified as follows.

- Let $f\colon \mathcal{X} \to \mathbb{F}_2^n$ be defined as $f(x) = \mathsf{bit\text{-}decompose}(H(x))$.
- Let $h\colon \mathcal{K} \to \mathbb{F}_2^n$ be defined as follows. On input $\boldsymbol{S} \in \mathbb{F}_2^{M\tau\times n}$:
  1. Parse $\boldsymbol{S} = (\boldsymbol{s}_{(i)}^{(\alpha)})_{\alpha\in[\tau], i\in[M]}$, where $\boldsymbol{s}_{(i)}^{(\alpha)} \in \mathbb{F}_2^n$.
  2. Output $\boldsymbol{s}_{(1)}^{(\tau)} \in \mathbb{F}_2^n$.

  It is easy to see that h is a group homomorphism, as required.
- Let $z\colon \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2$ is the function outputting $e^{(\tau)} \in \mathbb{F}_2$, an intermediate value during the execution of F (see Construction 1). By definition, z is as efficient as F.

It follows by definition that for any $\boldsymbol{S} \in \mathcal{K}, x \in \mathcal{X}$, F can be represented as

$$F(\boldsymbol{S}, x) = \langle f(x), h(\boldsymbol{S})\rangle + z(\boldsymbol{S}, x). \tag{51}$$

It remains to prove the following lemma.

Lemma 25 (**Vanishing of z**)**.** *It holds that*

$$\Pr_{\boldsymbol{S}\leftarrow\mathcal{K}}[z(\boldsymbol{S}, x) = 1] \le \mu + \mathsf{negl}(\lambda).$$

Proof. Conditioned on $\boldsymbol{S} \in \mathbb{F}_2^{M\tau\times n}$ (which happens except with negligible probability by Lemma 15), it holds that

$$\begin{aligned}
&\boldsymbol{r}^{(\tau-1)} \text{ is uniformly at random (Claim 19) and independent of } \boldsymbol{e}^{(\tau-1)} \text{ (Claim 19)}\\
\Longrightarrow\ &\boldsymbol{y}^{(\tau-1)} = \boldsymbol{r}^{(\tau-1)} + \boldsymbol{e}^{(\tau-1)} \text{ is uniformly at random}\\
\Longrightarrow\ &e^{(\tau)} = \mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\tau-1)}) \sim \mathsf{Ber}_\mu \text{ (Claim 3)}
\end{aligned} \tag{52}$$

It then follows that

$$\begin{aligned}
\Pr_{\boldsymbol{S}\leftarrow\mathcal{K}}[z(\boldsymbol{S}, x) = 1] &\le \Pr[\mathsf{Ber}_\mu = 1] + \Pr_{\boldsymbol{S}\leftarrow\mathcal{K}}[\boldsymbol{S} \text{ not full rank}]\\
&\le \mu + \mathsf{negl}(n).
\end{aligned} \tag{53}$$

□

□

G Proof of Theorem 9

Theorem 18 (restating Theorem 9). *Let n be a security parameter. Let $\mathcal{F} = \mathcal{F}_{n,\mu,t,\{\boldsymbol{A}_0,\dots,\boldsymbol{A}_n\},\tau,k}$ be the function family defined in Construction 2. Let $M = (t^k \cdot k \cdot (-\log \mu))^\tau$. Then, for any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{F}}(\mathcal{A}, n, \mu, \tau, t, k, Q) \leq Q \cdot \tau \cdot k^2 \cdot \mathsf{Adv}_{\mathsf{FRsparseLPN}}(n, \mu, t, n^2) + n^k \cdot (\mu k M)^{\tau-1}.$$

Proof. We will use the SPN version of our construction (Section 7.3) throughout the security proof due to its modularity.

We define our hybrid games as follows.

Game H_0. This is the real PRF attack game: the challenger chooses the public parameters pp, and samples an $F_{\boldsymbol{S}} \leftarrow \mathcal{F}$. The attacker gets the public parameters pp, and oracle access to $F_{\boldsymbol{S}}(\cdot)$.

Game H_1. This game is identical to Game H_0 except that the attacker $\mathcal{A}$ has oracle access to $F^{(1)}_{\boldsymbol{S}}(\cdot)$. $F^{(1)}_{\boldsymbol{S}}(\cdot)$ is identical to $F_{\boldsymbol{S}}(\cdot)$ except that it changes the order of computation and trimming. Specifically, the tuple $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{Linear}, \mathsf{S\text{-}Box})$ is specified as follows.

- $\mathsf{DMgen}(x, \alpha)$: On input $x \in \{0,1\}^{k \log n}$, $\alpha \in [\tau]$,
  1. Compute for $i \in [k], j \in [i+1]$ in parallel:
  $$\underline{\mathbf{A}_{(i,j)} \leftarrow \mathbf{A}_{(i,j)}(x) \in \mathbb{F}_2^{n\times n}}$$
  2. Set $\mathsf{sta}^{(\alpha)} = \left(\mathbf{A}^{(\alpha)}_{(i,1)}\right)_{i\in[k]} \in \mathbb{F}_2^{k\times n\times n}$.
  3. Set $M^{(\alpha)} = \left(\mathbf{A}^{(\alpha)}_{(i,j)}\right)_{i\in[k], j\in[2,i+1]}$.

  Output the public round key of round α, $\mathsf{prk}^{(\alpha)} = (\mathsf{sta}^{(\alpha)}, M^{(\alpha)})$.
- $\mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha)$: On input $\boldsymbol{S} \in \mathbb{F}_2^{n\times\tau}$, $\mathsf{sta}^{(\alpha)} \in \mathbb{F}_2^{k\times n\times n}$, $\alpha \in [\tau]$.
  1. Parse $\mathsf{sta}^{(\alpha)} = \left(\mathbf{A}^{(\alpha)}_{(i,1)}\right)_{i\in[k]}$, where $(\mathbf{A}^{(\alpha)}_{(i,1)} \in \mathbb{F}_2^{n\times n}$.
  2. Parse $\boldsymbol{S} = (\boldsymbol{s}^{(\alpha)}_{(i)})_{\alpha\in[\tau], i\in[k]}$, where $\boldsymbol{s}^{(\alpha)}_{(i)} \in \mathbb{F}_2^n$.
  3. Compute for $i \in [k]$ in parallel:
  $$\underline{\mathsf{srk}^{(\alpha)}_{(i)} \leftarrow \boldsymbol{s}^{(\alpha)}_{(i)} \cdot \mathbf{A}^{(\alpha)}_{(i,1)} \in \mathbb{F}_2^n.}$$

  Output the private round key of round α, $\mathsf{srk}^{(\alpha)} = (\mathsf{srk}^{(\alpha)}_{(1)}, \dots, \mathsf{srk}^{(\alpha)}_{(k)}) \in \mathbb{F}_2^{k\times n}$.
- $\mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)})$: On input $\boldsymbol{y}^{(\alpha-1)} \in \mathbb{F}_2^{k\times M_{\alpha-1}}$,
  1. Parse $\boldsymbol{y}^{(\alpha-1)} = (\boldsymbol{y}^{(\alpha-1)}_{(1)}, \dots, \boldsymbol{y}^{(\alpha-1)}_{(k)})$, where $\boldsymbol{y}^{(\alpha-1)}_{(i)} \in \mathbb{F}_2^{M_{\alpha-1}}$ for $i \in [k]$.

2. Compute for $i \in [k]$ in parallel:

$$\boldsymbol{w}_{(i)}^{(\alpha)} \leftarrow \mathsf{S\text{-}Ber}(\boldsymbol{y}_{(i)}^{(\alpha-1)}) \in \mathbb{F}_2^{M_\alpha \cdot k \cdot t^k}.$$

Output $\boldsymbol{w}^{(\alpha)} = (\boldsymbol{w}_{(1)}^{(\alpha)}, \dots, \boldsymbol{w}_{(k)}^{(\alpha)}) \in \mathbb{F}_2^{k \times M_\alpha \cdot k \cdot t^k}$.

- $\mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)})$: On input $\boldsymbol{w}^{(\alpha)} \in \mathbb{F}_2^{k \times M_\alpha \cdot k \cdot t^k}$,

  1. Parse $\boldsymbol{w}^{(\alpha)} = (\boldsymbol{w}_{(1)}^{(\alpha)}, \dots, \boldsymbol{w}_{(k)}^{(\alpha)})$, where $\boldsymbol{w}_{(i)}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha \cdot k \cdot t^k}$ for $i \in [k]$.
  2. Parse $\boldsymbol{w}_{(i)}^{(\alpha)} = (\boldsymbol{w}_{(i \to 1)}^{(\alpha)}, \dots, \boldsymbol{w}_{(i \to k)}^{(\alpha)})$, where $\boldsymbol{w}_{(i \to j)}^{(\alpha)} \in \mathbb{F}_2^{M_\alpha t^k}$ for $i, j \in [k]$.
  3. Parse $M^{(\alpha)} = \left(\mathbf{A}_{(i,j)}^{(\alpha)}\right)_{i \in [k], j \in [2, i+1]}$, where $\mathbf{A}_{(i,j)}^{(\alpha)} \in \mathbb{F}_2^{n \times n}$.
  4. $\underline{\text{Set } \overline{\mathbf{A}}_{(i,j+1)}^{(\alpha)} \leftarrow \mathsf{CTrim}_{M_\alpha}\left(\mathbf{A}_{(i,j+1)}^{(\alpha)}\right) \in \mathbb{F}_2^n.}$
  5. $\underline{\text{Set } \boldsymbol{v}_{(j \to i)}^{(\alpha)} \leftarrow \mathsf{Scatter}\left(\boldsymbol{w}_{(j \to i)}^{(\alpha)}, \overline{\mathbf{A}}_{(i,j+1)}^{(\alpha)}\right) \in \mathbb{F}_2^n.}$
  6. Compute for $i \in [k]$ in parallel:

  $$\underline{\boldsymbol{z}_{(i)}^{(\alpha)} \leftarrow \sum_{j=1}^{i} (\boldsymbol{v}_{(j \to i)}^{(\alpha)}) \cdot \mathbf{A}_{(i,j+1)}^{(\alpha)} \in \mathbb{F}_2^n.}$$

  Output $\boldsymbol{z}^{(\alpha)} = (\boldsymbol{z}_{(1)}^{(\alpha)}, \dots, \boldsymbol{z}_{(k)}^{(\alpha)}) \in \mathbb{F}_2^{k \times n}$.

On input $x \in \{0,1\}^{k \log n}$, the PRF $F_{\boldsymbol{S}}^{(1)}$ is therefore defined as:

1. Key schedule:
   (a) Compute $(\mathsf{sta}^{(\alpha)}, M^{(\alpha)}) \leftarrow \mathsf{DMgen}(x, \alpha)$ for $\alpha \in [\tau]$ in parallel.
   (b) Compute $\mathsf{srk}^{(\alpha)} \leftarrow \mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha) \in \mathbb{F}_2^{k \times n}$ for $\alpha \in [\tau]$ in parallel.
2. Round iteration:
   (a) Set $\boldsymbol{y}^{(0)} \leftarrow \mathbf{0} \in \mathbb{F}_2^{k \times M}$.
   (b) For round $\alpha = 1, \dots, \tau$ in sequential:
      i. $\boldsymbol{w}^{(\alpha)} \leftarrow \mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha)}) \in \mathbb{F}_2^{k \times M_\alpha \cdot k \cdot t^k}$.
      ii. $\boldsymbol{z}^{(\alpha)} \leftarrow \mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)}) \in \mathbb{F}_2^{k \times n}$.
      iii. $\underline{\boldsymbol{y}^{(\alpha)} \leftarrow \mathsf{CTrim}_{M_\alpha}(\mathsf{srk}^{(\alpha)} \oplus \boldsymbol{z}^{(\alpha)}) \in \mathbb{F}_2^{k \times M_\alpha}}$.
   (c) Set $y \leftarrow \boldsymbol{y}_{(k)}^{(\tau)} \in \mathbb{F}_2$.
3. Output $y \in \mathbb{F}_2$.

Game H_2. This game is identical to Game H_1 except that the attacker $\mathcal{A}$ has oracle access to another function, $\tilde{F}_{\boldsymbol{S}}^{(2)}(\cdot)$, where $\tilde{F}_{\boldsymbol{S}}^{(2)}(\cdot)$ is identical to $\tilde{F}_{\boldsymbol{S}}^{(1)}(\cdot)$ except that it samples $\boldsymbol{y}^{(0)}$ from a random function instead of setting it to zero. In particular, before the attacker begins making queries, for $i \in [k]$, the challenger (inefficiently) samples a random function $R_i : \{0,1\}^{k \log n} \to \mathbb{F}_2^m$. Then for $1 \le i \le k$, it first sets $\boldsymbol{y}_{(i)}^{(0)} = R_i(x_{1\ldots i}) \in \mathbb{F}_2^m$, then sets $\boldsymbol{y}^{(0)} = (\boldsymbol{y}_{(1)}^{(0)}, \ldots, \boldsymbol{y}_{(k)}^{(0)}) \in \mathbb{F}_2^{k \cdot m}$.

Game H_3. This game to Game H_0 is the same as Game H_2 to Game H_1. Specifically, this game is identical to Game H_0 except that the attacker $\mathcal{A}$ has oracle access to another function, $\tilde{F}_{\boldsymbol{S}}^{(3)}(\cdot)$, where $\tilde{F}_{\boldsymbol{S}}^{(3)}(\cdot)$ is identical to $F_{\boldsymbol{S}}(\cdot)$ except that it samples $\boldsymbol{y}^{(0)}$ from a random function instead of setting it to zero. In particular, before the attacker begins making queries, for $i \in [k]$, the challenger (inefficiently) samples a random function $R_i : \{0,1\}^{k \log n} \to \mathbb{F}_2^m$. Then for $1 \le i \le k$, it first sets $\boldsymbol{y}_{(i)}^{(0)} = R_i(x_{1\ldots i}) \in \mathbb{F}_2^m$, then sets $\boldsymbol{y}^{(0)} = (\boldsymbol{y}_{(1)}^{(0)}, \ldots, \boldsymbol{y}_{(k)}^{(0)}) \in \mathbb{F}_2^{k \cdot m}$.

Game H_4. This is the ideal PRF game. The challenger chooses the public parameter pp as in Game H_1 and gives it to the attacker. The challenger (inefficiently) samples a truly random function $R : \{0,1\}^{k \log n} \to \mathbb{F}_2$ and gives the attacker oracle access to $R(\cdot)$.

It is easy to verify that Game H_0 and Game H_1 are identical. Formally, we have the following lemma.

Lemma 26. *For any adversary $\mathcal{A}$, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_0, H_1) = 0 \tag{54}$$

Proof. Let us unroll the computation the computation in H_1. By Lemma 6 and Lemma 8, it holds that

$$\begin{aligned}
\boldsymbol{y}_{(i)}^{(\alpha)} &= \mathsf{CTrim}_{M_\alpha}\left(\mathsf{srk}_{(i)}^{(\alpha)} \oplus \boldsymbol{z}_{(i)}^{(\alpha)}\right) \\
&= \mathsf{CTrim}_{M_\alpha}\left(\mathsf{srk}_{(i)}^{(\alpha)}\right) \oplus \mathsf{CTrim}_{M_\alpha}\left(\boldsymbol{z}_{(i)}^{(\alpha)}\right) \\
&= \mathsf{CTrim}_{M_\alpha}\left(\boldsymbol{s}_{(i)}^{(\alpha)} \cdot \mathbf{A}_{(i,1)}^{(\alpha)}\right) \oplus \mathsf{CTrim}_{M_\alpha}\left(\sum_{j=1}^{i} (\boldsymbol{w}_{(j\to i)}^{(\alpha)}) \cdot \mathbf{A}_{(i,j+1)}^{(\alpha)}\right) \\
&= \boldsymbol{s}_{(i)}^{(\alpha)} \cdot \mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,1)}^{(\alpha)}) \oplus \sum_{j=1}^{i} \left(\mathsf{Scatter}\left(\boldsymbol{w}_{(j\to i)}^{(\alpha)}, \overline{\mathbf{A}}_{(i,j+1)}^{(\alpha)}\right)\right) \cdot \overline{\mathbf{A}}_{(i,j+1)}^{(\alpha)} \\
&= \boldsymbol{s}_{(i)}^{(\alpha)} \cdot \mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,1)}^{(\alpha)}) \oplus \sum_{j=1}^{i} \left(\boldsymbol{w}_{(j\to i)}^{(\alpha)}\right) \cdot \mathsf{RTrim}(\overline{\mathbf{A}}_{(i,j+1)}^{(\alpha)}),
\end{aligned} \tag{55}$$

Let us unroll the computation the computation in H_0. By Lemma 6, it holds that

$$\begin{aligned}
\boldsymbol{y}^{(\alpha)} &= \mathsf{srk}^{(\alpha)} \oplus \boldsymbol{z}^{(\alpha)} \\
&= \boldsymbol{s}_{(i)}^{(\alpha)} \cdot \overline{\mathbf{A}}_{(i,1)}^{(\alpha)} \oplus \sum_{j=1}^{i} (\boldsymbol{w}_{(j\to i)}^{(\alpha)})^T \cdot \overline{\overline{\mathbf{A}}}_{(i,j+1)}^{(\alpha)} \\
&= \boldsymbol{s}_{(i)}^{(\alpha)} \cdot \mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,1)}^{(\alpha)}) \oplus \sum_{j=1}^{i} \left(\boldsymbol{w}_{(j\to i)}^{(\alpha)}\right) \cdot \mathsf{RTrim}(\overline{\mathbf{A}}_{(i,j+1)}^{(\alpha)})
\end{aligned} \tag{56}$$

Therefore, $F_{\boldsymbol{S}}$ and $F_{\boldsymbol{S}}^{(1)}$ computes the same function. Therefore, the view of the adversary $\mathcal{A}$ is the same. $\square$

Corollary 6. *For any adversary $\mathcal{A}$, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_2, H_3) = 0 \tag{57}$$

Lemma 27. *For any efficient adversary $\mathcal{A}$ making at most Q queries, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_2, H_4) \leq Q \cdot \tau \cdot k^2 \cdot \mathsf{Adv}_{\mathsf{FRsparseLPN}}(n, \mu, t, n^2). \tag{58}$$

Proof. We define a sequence of intermediate hybrid games H_{2,α^*,i^*,j^*} for $\alpha^* \in [\tau]$ and $i^* \in [k], 0 \leq j^* \leq k$ as follows:

Game H_{2,α^*,i^*,j^*}. This game is identical to Game H_1 except that the attacker $\mathcal{A}$ has oracle access to another function, $\tilde{F}_{\boldsymbol{S}}^{(2,\alpha^*,i^*,j^*)}(\cdot)$, which we now define. Before the attacker begins making queries, the challenger (inefficiently) samples for $i \in [k]$ a random function $R_i : \{0,1\}^{k \log n} \to \mathbb{F}_2^{k \times n}$ and a random error function $E_i : \{0,1\}^{k \log n} \to \mathbb{F}_2^{k \times n}$ whose output is distributed as Ber_μ. The tuple $(\mathsf{DMgen}, \mathsf{srkgen}, \mathsf{Linear}, \mathsf{S\text{-}Box})$ is specified as in Game H_1. We need another function init, which is defined as follows.

- $\mathsf{init}(x, \alpha)$: On input $x \in \{0,1\}^{k \log n}$, $\alpha \in [\tau]$,
  1. Compute $\boldsymbol{u}_{(i)} \in \mathbb{F}_2^n$ for $i \in [k]$:
  $$\boldsymbol{u}_{(i)} = \begin{cases} R_i(x_{1\ldots i}) & i < i^* \\ R_i(x_{1\ldots j^*}) & i = i^* \end{cases} \tag{59}$$
  2. Compute $\boldsymbol{e}_{(i)} \in \mathbb{F}_2^{k \times n}$ for $1 \leq i \leq k$:
  $$\boldsymbol{e}_{(i)} \leftarrow E_i(x_{1\ldots i}) \in \mathbb{F}_2^{k \times n},$$
  3. Parse $\boldsymbol{e}_{(i)} = (\boldsymbol{e}_{(i \to 1)}, \ldots, \boldsymbol{e}_{(i \to k)})$, where $\boldsymbol{e}_{(i \to j)} \in \mathbb{F}_2^n$ for $1 \leq j \leq k$.
  4. Compute $\boldsymbol{y}_{(i)} \in \mathbb{F}_2^n$ for $i \in [k]$:
  $$\boldsymbol{y}_{(i)} = \begin{cases} \boldsymbol{u}_{(i)} & i < i^*, \\ \boldsymbol{u}_{(i)} \cdot \mathbf{A}_{(i,j^*+1)} + \sum_{j=j^*+1}^{i} (\boldsymbol{e}_{(j \to i)}) \cdot \mathbf{A}_{(i,j+1)} & i = i^*, \\ \boldsymbol{s}_{(i)}^{(\alpha)} \cdot \mathbf{A}_{(i,1)} + \sum_{j=1}^{i} (\boldsymbol{e}_{(j \to i)}) \cdot \mathbf{A}_{(i,j+1)} & i > i^*. \end{cases} \tag{60}$$
  5. Set $\boldsymbol{y} = (\boldsymbol{y}_{(1)}, \ldots, (\boldsymbol{y}_{(k)}) \in \mathbb{F}_2^{k \times n}$.
  6. Set $\boldsymbol{y} \leftarrow \mathsf{CTrim}_{M_\alpha}(\boldsymbol{y}) \in \mathbb{F}_2^{k \times M_\alpha}$.

On input $x \in \{0,1\}^{k \log n}$, $\tilde{F}_{\boldsymbol{S}}^{(2,\alpha^*,i^*,j^*)}(x)$ is then defined as:

1. Key schedule:
   (a) Compute $(\mathsf{sta}^{(\alpha)}, M^{(\alpha)}) \leftarrow \mathsf{DMgen}(x, \alpha)$ for $\alpha \in [\tau]$ in parallel.
   (b) Compute $\mathsf{srk}^{(\alpha)} \leftarrow \mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha) \in \mathbb{F}_2^{k \times n}$ for $\alpha \in [\tau]$ in parallel.

2. Round iteration:
   (a) Set $\underline{\boldsymbol{y}^{(\alpha^*-1)} \leftarrow \mathsf{init}(x, \alpha^*-1) \in \mathbb{F}_2^{k\times M^{\alpha^*-1}}}$.
   (b) For round $\underline{\alpha = \alpha^*, \dots, \tau}$ in sequential:
      i. $\boldsymbol{w}^{(\alpha)} \leftarrow \mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha)}) \in \mathbb{F}_2^{k\times M_\alpha \cdot k \cdot t^k}$.
      ii. $\boldsymbol{z}^{(\alpha)} \leftarrow \mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)}) \in \mathbb{F}_2^{k\times n}$.
      iii. $\boldsymbol{y}^{(\alpha)} \leftarrow \mathsf{CTrim}_{M_\alpha}(\mathsf{srk}^{(\alpha)} \oplus \boldsymbol{z}^{(\alpha)}) \in \mathbb{F}_2^{k\times M_\alpha}$.
   (c) Set $y \leftarrow \boldsymbol{y}^{(\tau)}_{(k)} \in \mathbb{F}_2$.
3. Output $y \in \mathbb{F}_2$.

It is easy to see that $H_2 = H_{2,1,k,k}$ and $H_4 = H_{2,\tau+1,k,k}$. We now show that each of the consecutive games are computationally indistinguishable, which is sufficient to prove the lemma.

Lemma 28. *For any efficient adversary $\mathcal{A}$ making at most Q queries, for $1 \le \alpha^* \le \tau+1$, $0 \le i^* \le k$, $1 \le j^* \le k$, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_{2,\alpha^*,i^*,j^*-1}, H_{2,\alpha^*,i^*,j^*}) \le Q \cdot \mathsf{Adv}_{\mathsf{FRsparseLPN}}(n, \mu, t, n^2) \tag{61}$$

Proof. Let $\mathcal{A}$ be a distinguisher for the Games H_{2,α^*,i^*,j^*-1} and H_{2,α^*,i^*,j^*} that makes at most Q queries. We construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to solve the $\mathsf{FRsparseLPN}_{n,\mu,t,n^2}$ problem. Algorithm $\mathcal{B}$ proceeds as follows:

1. At the start of the game, algorithm $\mathcal{B}$ receives $\mathsf{FRsparseLPN}_{n,\mu,t,n^2}$ challenge with Q secrets $(\boldsymbol{A}, \boldsymbol{B}) \in \mathbb{F}_2^{n\times n^2} \times \mathbb{F}_2^{Q\times n^2}$. It parses $\boldsymbol{A} = \{\boldsymbol{A}_x\}$ for $x \in [n]$, where $\boldsymbol{A}_x \in \mathbb{F}_2^{n\times n}$. It parses $\boldsymbol{B} = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_Q)$ for $\boldsymbol{b}_i = (\boldsymbol{b}_{i,0}, \dots, \boldsymbol{b}_{i,n}) \in \mathbb{F}_2^{n\times n}$. It samples the public parameter pp as in H_1 except for the input encoder Encode, which is already instantiated by $\boldsymbol{A}$. It then gives pp to the adversary $\mathcal{A}$. For $i \in [k]$, the challenger (inefficiently) samples a random function $R_i : \{0,1\}^k \to \mathbb{F}_2^{k\times n}$, and a error function $E_i : \{0,1\}^{k\log n} \to \mathbb{F}_2^{k\times n}$ whose output is distributed as Ber_μ. It samples the secret key $\boldsymbol{S} = (\boldsymbol{s}^{(\alpha)}_{(i)})_{\alpha\in[\tau], i\in[k]} \in \mathbb{F}_2^{\tau k\times n}$.
2. The the adversary $\mathcal{A}$ makes Q adaptive evaluation queries. For the $\mathcal{A}$'s t-th query $x \in \{0,1\}^k$:
   (a) The challenger first checks if $\mathcal{A}$ previously made a query $x' \in \{0,1\}^k$ in its t'-th query for which $x'_{1\dots j^*-1} = x_{1\dots j^*-1}$, if this is not the case, it chooses a previously unused $\boldsymbol{b}_i$ where $1 \le i \le Q$ and sets $\boldsymbol{b}'_t \leftarrow \boldsymbol{b}_i \in \mathbb{F}_2^{n\times n}$, otherwise, it uses the $\boldsymbol{b}'_{t'}$ that was chosen when queried by x' in the t'-th query and sets $\boldsymbol{b}'_t \leftarrow \boldsymbol{b}'_{t'} \in \mathbb{F}_2^{n\times n}$. It parses $\boldsymbol{b}'_{t'} = (\boldsymbol{b}'_{t',0}, \boldsymbol{b}'_{t',1}, \dots, \boldsymbol{b}'_{t',n})$.
   (b) For $i < i^*$, it sets $\boldsymbol{u}_{(i)} = R_i(x_{1\dots i}) \in \mathbb{F}_2^n$.
   (c) For $i \in [k]$, it sets $\boldsymbol{e}_{(i)} \leftarrow E_i(x_{1\dots i}) \in \mathbb{F}_2^{k\cdot n}$.
   (d) Parse $\boldsymbol{e}_{(i)} = (\boldsymbol{e}_{(i\to 1)}, \dots, \boldsymbol{e}_{(i\to k)})$, where $\boldsymbol{e}_{(i\to j)} \in \mathbb{F}_2^n$ for $1 \le j \le k$.
   (e) For $i \in [k]$, set

$$\boldsymbol{y}^{(\alpha^*-1)}_{(i)} = \begin{cases} \boldsymbol{u}_{(i)} & i < i^*, \\ (\boldsymbol{b}'_{t',x_{j^*}})^T \cdot \mathbf{A}_{(i,j^*+1)} + \sum_{j=j^*+1}^{i} (\boldsymbol{e}_{(j\to i)}) \cdot \mathbf{A}_{(i,j+1)} & i = i^*, \\ \boldsymbol{s}^{(\alpha^*-1)}_{(i)} \cdot \mathbf{A}_{(i,1)} + \sum_{j=1}^{i} (\boldsymbol{e}_{(j\to i)}) \cdot \mathbf{A}_{(i,j+1)} & i > i^*. \end{cases} \tag{62}$$

(f) It sets $\boldsymbol{y}^{(\alpha^*-1)} = (\boldsymbol{y}_{(1)}, \ldots, (\boldsymbol{y}_{(k)}) \in \mathbb{F}_2^{k \times n}$.

(g) It sets $\boldsymbol{y}^{(\alpha^*-1)} \leftarrow \mathsf{Trim}_{m_{\alpha^*-1}}(\boldsymbol{y}) \in \mathbb{F}_2^{k \times m_{\alpha^*-1}}$.

If $\alpha^* = \tau$ and $i^* = k$ and $j^* = k$, it sets $y = \boldsymbol{y}_{(k)}^{(\tau)}$ and outputs $y \in \mathbb{F}_2$. Otherwise,

(a) Key schedule:
   i. Compute $(\mathsf{sta}^{(\alpha)}, M^{(\alpha)}) \leftarrow \mathsf{DMgen}(x, \alpha)$ for $\alpha \in [\tau]$ in parallel.
   ii. Compute $\mathsf{srk}^{(\alpha)} \leftarrow \mathsf{srkgen}(\boldsymbol{S}, \mathsf{sta}^{(\alpha)}, \alpha) \in \mathbb{F}_2^{k \times n}$ for $\alpha \in [\tau]$ in parallel.

(b) Round iteration:
   i. Set $\underline{\boldsymbol{y}^{(\alpha^*-1)} \leftarrow \boldsymbol{y}^{(\alpha^*-1)} \in \mathbb{F}_2^{k \times M^{\alpha^*-1}}}$.
   ii. For round $\alpha = \alpha^*, \ldots, \tau$ in sequential:
      A. $\boldsymbol{w}^{(\alpha)} \leftarrow \mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha)}) \in \mathbb{F}_2^{k \times M_\alpha \cdot k \cdot t^k}$.
      B. $\boldsymbol{z}^{(\alpha)} \leftarrow \mathsf{Linear}(\boldsymbol{w}^{(\alpha)}, M^{(\alpha)}) \in \mathbb{F}_2^{k \times n}$.
      C. $\boldsymbol{y}^{(\alpha)} \leftarrow \mathsf{CTrim}_{M_\alpha}(\mathsf{srk}^{(\alpha)} \oplus \boldsymbol{z}^{(\alpha)}) \in \mathbb{F}_2^{k \times M_\alpha}$.
   iii. Set $y \leftarrow \boldsymbol{y}_{(k)}^{(\tau)} \in \mathbb{F}_2$.

(c) Output $y \in \mathbb{F}_2$.

We now show that depending on whether the algorithm $\mathcal{B}$ is interacting with the real $\mathsf{FRsparseLPN}$ oracle or the ideal oracle, it perfectly simulates $\mathcal{A}$'s view of either H_{2,α^*,i^*,j^*-1} and H_{2,α^*,i^*,j^*}.

- If $\mathcal{B}$ is interacting with the real $\mathsf{FRsparseLPN}$ oracle, then each vector $\boldsymbol{b}_{i,x_{j^*}}$ is a valid $\mathsf{FRsparseLPN}$ sample $\boldsymbol{s}_i \cdot \boldsymbol{A}_{x_{j^*}} + \boldsymbol{e}_{i,x_{j^*}}$ for $\boldsymbol{s}_i \leftarrow \mathbb{F}_2^n$, $\boldsymbol{e}_{i,x_{j^*}} \leftarrow \mathsf{Ber}_\mu^n$. For each of $\mathcal{A}$'s t-th query $x \in \{0,1\}^{k \log n}$, let $\boldsymbol{s}_{x_{1\ldots j^*-1}}, \boldsymbol{e}_{x_{1\ldots j^*}} \in \mathbb{F}_2^n$ be the secret and noise corresponding to the $\boldsymbol{b}'_t$ that we choose. We have

$$\begin{aligned} \boldsymbol{y}_{(i^*)}^{(\alpha^*-1)} &= (\boldsymbol{b}'_{t',x_{j^*}})^T \cdot \mathbf{A}_{(i^*,j^*+1)} + \sum_{j=j^*+1}^{i^*} (\boldsymbol{e}_{(j \to i^*)}) \cdot \mathbf{A}_{(i^*,j+1)} \\ &= (\boldsymbol{s}_{x_{1\ldots j^*-1}} \boldsymbol{A}_{x_{j^*}} + \boldsymbol{e}_{x_{1\ldots j^*}})^T \cdot \mathbf{A}_{(i^*,j^*+1)} + \sum_{j=j^*+1}^{i^*} (\boldsymbol{e}_{(j \to i^*)}) \cdot \mathbf{A}_{(i^*,j+1)} \\ &= \boldsymbol{s}_{x_{1\ldots j^*-1}}^T \cdot \mathbf{A}_{(i^*,j^*)} + \boldsymbol{e}_{x_{1\ldots j^*}} \cdot \mathbf{A}_{(i^*,j^*+1)} + \sum_{j=j^*+1}^{i^*} (\boldsymbol{e}_{(j \to i^*)}) \cdot \mathbf{A}_{(i^*,j+1)}. \end{aligned} \tag{63}$$

  In game H_{2,α^*,i^*,j^*-1}, we have that

$$\begin{aligned} \boldsymbol{y}_{(i^*)}^{(\alpha^*-1)} &= R_{i^*}(x_{1\ldots x_{j^*-1}}) \cdot \mathbf{A}_{(i^*,j^*)} + \sum_{j=j^*}^{i^*} (\boldsymbol{e}_{(j \to i^*)}) \cdot \mathbf{A}_{(i^*,j+1)} \\ &= R_{i^*}(x_{1\ldots x_{j^*-1}}) \cdot \mathbf{A}_{(i^*,j^*)} + (\boldsymbol{e}_{(j^* \to i^*)}) \cdot \mathbf{A}_{(i^*,j^*+1)} + \sum_{j=j^*}^{i^*} (\boldsymbol{e}_{(j \to i^*)}) \cdot \mathbf{A}_{(i^*,j+1)} \end{aligned} \tag{64}$$

$\boldsymbol{s}_{x_{1\ldots x_{j^*-1}}}$ has the same distribution as $R_{i^*}(x_{1\ldots x_{j^*-1}})$ due to the way we choose $\boldsymbol{b}'_t$. Likewise, $\boldsymbol{e}_{x_{1\ldots j^*}}$ has the same distribution as $\boldsymbol{e}_{(j^* \to i^*)}$. Therefore, by definition, $\mathcal{B}$ perfectly simulates H_{2,α^*,i^*,j^*-1}.

- If $\mathcal{B}$ is interacting with the ideal oracle, then each vector $\boldsymbol{b}_i$ for $i \in [Q]$ is a random vector in $\mathbb{F}_2^{n^2}$ and therefore, for the t-th query $x_t \in \{0,1\}^k$, the way we choose $\boldsymbol{b}'_{t',x_{j^*}}$ is equivalent to obtaining it from a random function by $R(x_{1\ldots,j})$. Therefore, it has the same distribution as $\boldsymbol{u}_{(i^*)}$ in H_{2,α^*,i^*,j^*}. Hence, algorithm $\mathcal{B}$ perfectly simulates H_{2,α^*,i^*,j^*}.

We have shown that depending on whether $\mathcal{B}$ is interacting with the real $\mathsf{FRsparseLPN}$ or ideal oracle with Q secrets, it perfectly simulates the distributions of either H_{2,α^*,i^*,j^*-1} or H_{2,α^*,i^*,j^*}. Therefore, with the same distinguishing advantage of $\mathcal{A}$ (up to a multiplicative factor of Q), algorithm $\mathcal{B}$ solves the $\mathsf{FRsparseLPN}$ problem.

Lemma 29. *For any adversary $\mathcal{A}$, for $1 \le \alpha^* \le \tau + 1$, $0 \le i^* \le k$, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_{2,\alpha^*,i^*,k}, H_{2,\alpha^*,i^*+1,0}) = 0 \tag{65}$$

Proof. The only difference between Game $H_{2,\alpha^*,i^*,k}$ and Game $H_{2,\alpha^*,i^*+1,0}$ is the way we set $\boldsymbol{y}_{(i)}^{(\alpha^*)}$. It suffices to show that the distribution of $\boldsymbol{y}_{(i)}^{(\alpha^*)}$ are identical in $H_{2,\alpha^*,i^*,k}$ and $H_{2,\alpha^*,i^*+1,0}$.

In Game $H_{2,\alpha^*,i^*,k}$, for $i \in [k]$, since $\mathbf{A}_{(i,k^*+1)} = \mathbb{I}$, we have

$$\boldsymbol{y}_{(i)}^{(\alpha^*-1)} = \begin{cases} \boldsymbol{u}_{(i)} & i < i^*+1, \\ \boldsymbol{s}_{(i^*+1)}^{(\alpha^*-1)} \cdot \mathbf{A}_{(i,1)} + \sum_{j=1}^{i}(\boldsymbol{e}_{(j\to i)}) \cdot \mathbf{A}_{(i,j+1)} & i = i^*+1, \\ \boldsymbol{s}_{(i)}^{(\alpha^*-1)} \cdot \mathbf{A}_{(i,1)} + \sum_{j=1}^{i}(\boldsymbol{e}_{(j\to i)}) \cdot \mathbf{A}_{(i,j+1)} & i > i^*+1. \end{cases} \tag{66}$$

In Game $H_{2,\alpha^*,i^*+1,0}$, for $i \in [k]$, we have

$$\boldsymbol{y}_{(i)}^{(\alpha^*-1)} = \begin{cases} \boldsymbol{u}_{(i)} & i < i^*+1, \\ \boldsymbol{u}_{(i^*+1)} \cdot \mathbf{A}_{(i,1)} + \sum_{j=1}^{i}(\boldsymbol{e}_{(j\to i)}) \cdot \mathbf{A}_{(i,j+1)} & i = i^*+1, \\ \boldsymbol{s}_{(i)}^{(\alpha^*-1)} \cdot \mathbf{A}_{(i,1)} + \sum_{j=1}^{i}(\boldsymbol{e}_{(j\to i)}) \cdot \mathbf{A}_{(i,j+1)} & i > i^*. \end{cases} \tag{67}$$

Since $\boldsymbol{u}_{(i^*+1)} = R_{i^*+1}(x_{1\ldots 0}) = R_{i^*+1}(\perp)$, $\boldsymbol{u}_{(i^*+1)}$ is random and consistent for different input x. Therefore, $\boldsymbol{u}_{(i^*+1)}$ is distributed exactly as $\boldsymbol{s}_{(\alpha^*,i^*+1)}$, so is $H_{2,\alpha^*,i^*,k}$ and $H_{2,\alpha^*,i^*+1,0}$.

□

Lemma 30. *For any adversary $\mathcal{A}$, for $1 \le \alpha^* < \tau + 1$, $1 \le i^* \le k$, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_{2,\alpha^*,k,k}, H_{2,\alpha^*+1,0,0}) = 0 \tag{68}$$

Proof. In Game $H_{2,\alpha^*,k,k}$, we have $\boldsymbol{y}_{(i)}^{(\alpha^*-1)} = \boldsymbol{u}_{(i)} = R_i(x_{1\ldots i})$ for $i \in [k]$. Therefore, $\boldsymbol{w}_{(i)}^{(\alpha^*)} = \mathsf{S\text{-}Box}\left(\boldsymbol{y}_{(i)}^{(\alpha^*-1)}\right) = \mathsf{S\text{-}Box}\left(R_i(x_{1\ldots i})\right)$, which distributed exactly as $E_i(x_{1\ldots i})$. Therefore, in $H_{2,\alpha^*,k,k}$ we have

$$\boldsymbol{y}_{(i)}^{(\alpha^*)} = \boldsymbol{s}_{(i)}^{(\alpha^*)} \cdot \mathbf{A}_{(i,1)} + \sum_{j=1}^{i}(\boldsymbol{e}_{(j\to i)}^{(\alpha^*)}) \cdot \mathbf{A}_{(i,j+1)}, \tag{69}$$

which distributed exactly as $\boldsymbol{y}_{(i)}^{(\alpha^*)}$ in $H_{2,\alpha^*+1,0,0}$.

□

Finally, by the triangle inequality, we have $\mathsf{Adv}_{\mathcal{A}}(H_2, H_4) \leq Q \cdot \tau \cdot k^2 \cdot \mathsf{Adv}_{\mathsf{FRsparseLPN}}(n, \mu, t, n^2)$ for any efficient adversary $\mathcal{A}$ making at most Q queries, which completes the proof.

□

Lemma 31. *For any adversary $\mathcal{A}$, we have*

$$\mathsf{Adv}_{\mathcal{A}}(H_0, H_3) \leq n^k \cdot (\mu k M)^{\tau-1}. \tag{70}$$

Proof. Without loss of generality, we assume $\mathcal{A}$ is deterministic. If for all queries made by $\mathcal{A}$, $F_{\boldsymbol{S}}(\cdot)$ and $\tilde{F}^{(3)}_{\boldsymbol{S}}(\cdot)$ have the same output except with some small probability (over the randomness of secret key $\boldsymbol{S}$, and the randomness in $\tilde{F}^{(3)}_{\boldsymbol{S}}(\cdot)$), then the distinguishing advantage of $\mathcal{A}$ is information-theoretically upper bounded by that probability.

Therefore, it suffices to prove the following claim. The lemma then follows by applying a union bound over all $x \in \{0,1\}^{k \log n}$. It is important to note that we cannot directly apply a union bound over the Q queries made by $\mathcal{A}$ and conclude the result as $Q \cdot (\mu k m)^{\tau-1}$, as the queries are adaptive, which could depend on previous answers.

Claim 20. *For any $x \in \{0,1\}^{k \log n}$, it holds that*

$$\Pr[F_{\boldsymbol{S}}(x) \neq \tilde{F}^{(3)}_{\boldsymbol{S}}(x)] \leq (\mu k M)^{\tau-1},$$

where the probability is over the randomness of the secret key $\boldsymbol{S}$, and $\tilde{\boldsymbol{y}}_{(0)}$ in $\tilde{F}^{(3)}_{\boldsymbol{S}}$.

Proof. Fix any $x \in \{0,1\}^{k \log n}$. We consider the transcript of the intermediate results during the evaluation of $F_{\boldsymbol{S}}(x)$ and $\tilde{F}^{(3)}_{\boldsymbol{S}}(x)$. We use X to denote the random variable during the evaluation of $F_{\boldsymbol{S}}(x)$ (e.g., $\boldsymbol{y}$, $\boldsymbol{z}$). We use $\tilde{X}$ to denote the random variable during the evaluation of $\tilde{F}^{(3)}_{\boldsymbol{S}}(x)$ (e.g., $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{z}}$).

Claim 21. *It holds that*

$$(\mathsf{srk}^{(1)}, \ldots, \mathsf{srk}^{(\tau)})$$

is distributed uniformly at random for random key $\boldsymbol{S}$.

Proof. By definition, for $\alpha \in [\tau], i \in [k]$ we have

$$\begin{aligned} \mathsf{srk}^{(\alpha)}_{(i)} &= \boldsymbol{s}^{(\alpha)}_{(i)} \cdot \overline{\mathbf{A}}^{(\alpha)}_{(i,1)} \\ &= \boldsymbol{s}^{(\alpha)}_{(i)} \cdot \mathsf{CTrim}_{M_\alpha}(\mathbf{A}_{(i,1)}(x)) \\ &= \mathsf{CTrim}_{M_\alpha}(\boldsymbol{s}^{(\alpha)}_{(i)} \cdot \mathbf{A}_{(i,1)}(x)) \end{aligned} \tag{71}$$

Since $\boldsymbol{A}_0, \ldots, \boldsymbol{A}_n \in \mathbb{F}_2^{n \times n}$ are all full rank, $\mathbf{A}_{(i,1)}(x) = \boldsymbol{A}_{x_1} \ldots \boldsymbol{A}_{x_k}$ is also full rank. Since x is independent of $\boldsymbol{s}^{(\alpha)}_{(i)}$, so is $\mathbf{A}_{(i,1)}(x)$. Therefore, for any $\boldsymbol{u}_{(\alpha,i)} \in \mathbb{F}_2^n$, it holds that

$$\begin{aligned} &\Pr_{\boldsymbol{S}}[(\boldsymbol{s}^{(\alpha)}_{(i)} \cdot \mathbf{A}_{(i,1)}(x))_{\alpha\in[\tau], i\in[k]} = (\boldsymbol{u}_{(\alpha,i)})_{\alpha\in[\tau], i\in[k]}] \\ =&\Pr_{\boldsymbol{S}}[(\boldsymbol{s}^{(\alpha)}_{(i)})_{\alpha\in[\tau], i\in[k]} = (\boldsymbol{u}_{(\alpha,i)} \cdot (\mathbf{A}_{(i,1)}(x))^{-1})_{\alpha\in[\tau], i\in[k]}] \\ =&1/|\boldsymbol{S}| \end{aligned} \tag{72}$$

Therefore, $(\boldsymbol{s}_{(i)}^{(\alpha)} \cdot \mathbf{A}_{(i,1)}(x))_{\alpha\in[\tau],i\in[k]}$ is distributed uniformly at random, so is $(\mathsf{srk}_{(i)}^{(\alpha)})_{\alpha\in[\tau],i\in[k]}$. $\square$

Claim 22. *For $\alpha \in [\tau]$, $\mathsf{srk}^{(\alpha)}$ is independent of $(\boldsymbol{z}^{(\alpha')}, \tilde{\boldsymbol{z}}^{(\alpha')})$ for $\alpha' \le \alpha$.*

Proof. For $\alpha' \le \alpha$, $(\boldsymbol{z}^{(\alpha')}, \tilde{\boldsymbol{z}}^{(\alpha')})$ only depends on $\boldsymbol{y}^{(0)}$, $\mathsf{srk}^{(\alpha'')}$ for $\alpha'' < \alpha'$. Since $\mathsf{srk}^{(\alpha)}$ is independent of $\boldsymbol{y}^{(0)}$, and $\mathsf{srk}^{(\alpha'')}$ by Claim 21. Therefore, $\mathsf{srk}^{(\alpha)}$ is independent of $(\boldsymbol{z}^{(\alpha')}, \tilde{\boldsymbol{z}}^{(\alpha')})$.

$\square$

Observe that if $F_{\boldsymbol{S}}(x) \ne \tilde{F}_{\boldsymbol{S}}^{(3)}(x)$, then it must hold that for all $\alpha \in [\tau]$, $\boldsymbol{y}^{(\alpha)} \ne \tilde{\boldsymbol{y}}^{(\alpha)}$ since round iteration is deterministic. For $\alpha = 1, \ldots, \tau$, we consider the probability that $\boldsymbol{y}^{(\alpha)} = \tilde{\boldsymbol{y}}^{(\alpha)}$ conditioned on $\boldsymbol{y}^{(\alpha-1)} \ne \tilde{\boldsymbol{y}}^{(\alpha-1)}$. We have

$$\begin{aligned}
&\Pr_{\boldsymbol{S},\tilde{\boldsymbol{y}}^{(0)}}[\boldsymbol{y}^{(\alpha)} = \tilde{\boldsymbol{y}}^{(\alpha)} \mid \boldsymbol{y}^{(\alpha-1)} \ne \tilde{\boldsymbol{y}}^{(\alpha-1)}] \\
=&\Pr[\mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)}) = \mathsf{S\text{-}Box}(\tilde{\boldsymbol{y}}^{(\alpha-1)}) \mid \boldsymbol{y}^{(\alpha-1)} \ne \tilde{\boldsymbol{y}}^{(\alpha-1)}] \\
\ge&\Pr[\mathsf{S\text{-}Box}(\boldsymbol{y}^{(\alpha-1)}) = \mathsf{S\text{-}Box}(\tilde{\boldsymbol{y}}^{(\alpha-1)})] \\
=&\Pr[\mathsf{S\text{-}Box}(\mathsf{srk}^{(\alpha-1)} + \boldsymbol{z}^{(\alpha-1)}) = \mathsf{S\text{-}Box}(\mathsf{srk}^{(\alpha-1)} + \tilde{\boldsymbol{z}}^{(\alpha-1)})] \\
\ge&(1-\mu)^{2kM_{\alpha-1}/\ell} \ge (1-\mu)^{kM} \ge 1 - \mu kM,
\end{aligned} \tag{73}$$

where the last line is due to Lemma 3, Claim 21, and Claim 22. Therefore, it holds that

$$\begin{aligned}
&\Pr_{x,\boldsymbol{S},\boldsymbol{y}^{(0)}}[F_{\boldsymbol{S}}(x) \ne \tilde{F}_{\boldsymbol{S}}^{(1)}(x)] \\
=&\Pr\left[\bigcap_{\alpha=1}^{\tau} \boldsymbol{y}^{(\alpha)} \ne \tilde{\boldsymbol{y}}^{(\alpha)}\right] \\
=&\Pr[\boldsymbol{y}^{(0)} \ne \tilde{\boldsymbol{y}}^{(0)}] \cdot \prod_{\alpha=2}^{\tau} \Pr[\boldsymbol{y}^{(\alpha)} \ne \tilde{\boldsymbol{y}}^{(\alpha)} \mid \boldsymbol{y}^{(\alpha-1)} \ne \tilde{\boldsymbol{y}}^{(\alpha-1)}] \\
\le&(\mu kM)^{\tau}
\end{aligned} \tag{74}$$

$\square$

$\square$

$\square$

$\square$

H Proof of Theorem 10

Theorem 19 (**Restating Theorem 10**)**.** *Under the parameter setting in Theorem 9, Construction 2 has γ-AKH structure (Definition 13) where $\gamma = n^{-c+\delta}$ for any $\delta > 0$.*

Proof. We will use the same notation as in Appendix G, corresponding to the SPN version of our construction (Section 7.3).

Denote $\mathcal{X} = \{0,1\}^{k \log n}$, $\mathcal{K} = \mathbb{F}_2^{\tau k \times n}$. The required (f, h, z) in AKH structure (Definition 13) is specified as follows.

- Let $f \colon \mathcal{X} \to \mathbb{F}_2^n$ be defined as $f(x) = \overline{\mathbf{A}}^{(\tau)}_{(k,1)}(x)$.
- Let $h \colon \mathcal{K} \to \mathbb{F}_2^n$ be defined as follows. On input $\boldsymbol{S} \in \mathbb{F}_2^{\tau k \times n}$:
  1. Parse $\boldsymbol{S} = \left(\boldsymbol{s}^{(\alpha)}_{(i)}\right)_{\alpha \in [\tau], i \in [k]} \in \mathbb{F}_2^{\tau k \times n}$, where $\boldsymbol{s}^{(\alpha)}_{(i)} \in \mathbb{F}_2^n$.
  2. Output $\boldsymbol{s}^{(\tau)}_{(k)} \in \mathbb{F}_2^n$.

  It is easy to see that h is a group homomorphism, as required.
- Let $z \colon \mathcal{K} \times \mathcal{X} \to \mathbb{F}_2$ is the function outputting $z^{(\tau)}_{(k)} \in \mathbb{F}_2$, an intermediate value during the execution of F (see Construction 2). By definition, z is as efficient as F.

It follows by definition that for any $\boldsymbol{S} \in \mathcal{K}, x \in \mathcal{X}$, F can be represented as

$$F(\boldsymbol{S}, x) = \langle f(x), h(\boldsymbol{S}) \rangle + z(\boldsymbol{S}, x). \tag{75}$$

It remains to prove the following lemma.

Lemma 32 (**Vanishing of z**)**.** *It holds that*

$$\Pr_{\boldsymbol{S} \leftarrow \mathcal{K}}[z(\boldsymbol{S}, x) = 0] \leq \mu + \mathsf{negl}(\lambda).$$

Proof. It holds that

$$\begin{aligned} & \mathsf{srk}^{(\tau-1)} \text{ is uniformly at random (Claim 21) and independent of } \boldsymbol{z}^{(\tau-1)} \text{ (Claim 22)} \\ \Longrightarrow\ & \boldsymbol{y}^{(\tau-1)} = \mathsf{srk}^{(\tau-1)} + \boldsymbol{z}^{(\tau-1)} \text{ is uniformly at random} \\ \Longrightarrow\ & \boldsymbol{w}^{(\tau)} = \mathsf{S\text{-}Ber}(\boldsymbol{y}^{(\tau-1)}) \sim \mathsf{Ber}_\mu^{k \times k \cdot t^k} \text{ (Claim 3)} \end{aligned} \tag{76}$$

Since

$$z^{(\tau)}_{(k)} = \sum_{j=1}^{k} \boldsymbol{w}^{(\tau)}_{(j \to k)} \cdot \overline{\overline{\mathbf{A}}}^{(\tau)}_{(k,j+1)} = \sum_{j=1}^{k} \boldsymbol{w}^{(\tau)}_{(j \to k)} \cdot \boldsymbol{a}_j,$$

where we denote $\boldsymbol{a}_j = \overline{\overline{\mathbf{A}}}^{(\tau)}_{(k,j+1)} \in \mathbb{F}_2^{t^k}$. Therefore, $z^{(\tau)}_{(k)}$ is the sum of $k \cdot t^k$ random and independent Bernoulli variables with parameter μ. By Piling-up lemma (Lemma 17), it holds that $z^{(\tau)}_{(k)} \sim \mathsf{Ber}_{\mu'}$ where $\mu' < k \cdot t^k \cdot \mu$. Therefore,

$$\Pr_{\boldsymbol{S} \leftarrow \mathcal{K}}[z(\boldsymbol{S}, x) = 1] < k \cdot t^k \cdot \mu \leq n^{-c+\delta} \tag{77}$$

for any $\delta > 0$. □

□

I Extension to LPN over Rings

In this section, we introduce Bernoulli Sample Function over $\mathbb{Z}_p$ as a generalization of that over $\mathbb{Z}_2$ (Definition 14). Afterwards, it is trivial to extend all our constructions to $\mathbb{Z}_p$.

Definition 25 (**Bernoulli Sample Function over** $\mathbb{Z}_p$)**.** Let p_1, p_2, t be integers, $\mu = 1/t$, and $t \cdot p_2/p_1 = \mathsf{negl}(n)$. For input $x \in \mathbb{Z}_{p_1}$, $\mathsf{S\text{-}Ber}_\mu(x)$ is defined as follows.

1. $r' \leftarrow x \mod (t \cdot p_2) \in \mathbb{Z}_{t \cdot p_2}$.
2. $e_1 \leftarrow \lfloor r'/p_2 \rfloor \in \mathbb{Z}_t$.
3. $e_2 \leftarrow r' \mod p_2 \in \mathbb{Z}_{p_2}$.
4. If $e_1 \neq 0$, output 0. Otherwise output $e_2 \in \mathbb{Z}_{p_2}$.

This function is extended to input $x \in \mathbb{F}_{p_1}^m$ by applying it component-wise, yielding output in $\mathbb{F}_{p_2}^m$.

Remark 17. The nice feature of $\mathsf{S\text{-}Ber}$ that it is length-preserving, i.e., on an ring element output another ring element (of a smaller ring). However, it can no longer computed in depth $o(\log n)$ and require depth $O(\log n)$.

Claim 23. *It holds that*

$$\mathsf{SD}(\mathsf{S\text{-}Ber}(u), \mathsf{Ber}_\mu(\mathbb{Z}_{p_2})) = t \cdot p_2/p_1, \tag{78}$$

where $u \leftarrow Z_{p_1}$. In particular, if we set $p_1 = 2^n \cdot t \cdot p_2$, then

$$\mathsf{SD}(\mathsf{S\text{-}Ber}(u), \mathsf{Ber}_\mu(\mathbb{Z}_{p_2})) = 2^{-n} \tag{79}$$

where $u \leftarrow Z_{p_1}$.

Proof. Define $T_1 \colon \mathbb{Z}_{t \cdot p_2} \to \mathbb{Z}_t \times \mathbb{Z}_{p_2}$, $T_1(x) = (\lfloor x/p_2 \rfloor, x \mod p_2)$. We then have

$$\mathsf{SD}(T_1(U_{t \cdot p_2}), (U_t, U_{p_2})) = 0 \tag{80}$$

where $U_{t \cdot p_2}$ is uniform distribution over $U_{t \cdot p_2}$, (U_t, U_{p_2}) are uniform distribution over $\mathbb{Z}_t \times \mathbb{Z}_{p_2}$.

Define $T_2 \colon \mathbb{Z}_t \times \mathbb{Z}_{p_2} \to \mathbb{Z}_{p_2}$, $T_2(x_1, x_2) = \mathbb{I}[x_1 \neq 0] \cdot x_2$. We then have

$$\mathsf{SD}(T_2(U_t, U_{p_2}), \mathsf{Ber}_\mu(\mathbb{Z}_{p_2})) = 0 \tag{81}$$

Therefore, it holds that

$$\mathsf{SD}(T_2 \circ T_1(U_{t \cdot p_2}), \mathsf{Ber}_\mu(\mathbb{Z}_{p_2})) = 0 \tag{82}$$

Since $t \cdot p_2/p_1 = \mathsf{negl}(n)$, we have

$$\mathsf{SD}(U_{p_1} \mod (t \cdot p_2), U_{t \cdot p_2}) \leq t \cdot p_2/p_1. \tag{83}$$

It then follows that

$$\begin{aligned} &\mathsf{SD}(\mathsf{S\text{-}Ber}(u), \mathsf{Ber}_\mu(\mathbb{Z}_{p_2})) \\ =&\mathsf{SD}(T_2 \circ T_1(U_{p_1} \mod (t \cdot p_2)), T_2 \circ T_1(U_{t \cdot p_2})) \\ \leq&\mathsf{SD}(U_{p_1} \mod (t \cdot p_2), U_{t \cdot p_2}) \\ \leq& t \cdot p_2/p_1. \end{aligned} \tag{84}$$

□

Lemma 33 (**Derandomization Effect**)**.** *Let $x \leftarrow \mathbb{F}_{p_1}$. For any random variable (e, e') over $\mathbb{F}_{p_1} \times \mathbb{F}_{p_1}$ that is independent of x, it holds that*

$$\Pr_{x,e,e'}[\mathsf{S\text{-}Ber}(x + e) = \mathsf{S\text{-}Ber}(x + e')] \geq 1 - 2\mu - 2/p_1. \tag{85}$$

Proof. By law of total probability,

$$\begin{aligned} &\Pr_{x,e,e'}[\mathsf{S\text{-}Ber}(x + e) = \mathsf{S\text{-}Ber}(x + e')] \\ =&\sum_{v \in \mathbb{F}_2^m} \Pr_{x,e,e'}[\mathsf{S\text{-}Ber}(x + e) = v \wedge \mathsf{S\text{-}Ber}(x + e') = v] \\ \geq&\Pr_{x,e,e'}[\mathsf{S\text{-}Ber}(x + e) = 0 \wedge \mathsf{S\text{-}Ber}(x + e') = 0]. \end{aligned}$$

Since (e, e') is independent of x,

$$= \sum_{v, v' \in \mathbb{Z}_{p_1}} \Pr_{e,e'}[(e, e') = (v, v')] \cdot \Pr_x[x \notin \mathsf{BAD}_v \wedge x \notin \mathsf{BAD}_{v'}],$$

where for $v \in \mathbb{Z}_{p_1}$, BAD_v is defined to be the set

$$\mathsf{BAD}_v = \{k_1 \cdot t \cdot p_2 + k_2 - v \in \mathbb{Z}_{p_1} \mid k \in [p_1/(t \cdot p_2)], k_2 \in [p_2 - 1]\}.$$

Therefore, for $v \in \mathbb{Z}_{p_1}$, $|\mathcal{S}(v)| \leq p_1/t + 1$. We then have $|\mathsf{BAD}_v \cup \mathsf{BAD}_{v'}| \leq 2(p_1/t + 1)$. Since the x is uniform random variables,

$$\begin{aligned} \geq&\sum_{v, v' \in \mathbb{Z}_{p_1}} \Pr_{e,e'}[(e, e') = (v, v')] \cdot \left(1 - \frac{2(p_1/t + 1)}{p_1}\right) \\ =&\left(1 - \frac{2(p_1/t + 1)}{p_1}\right) \\ >&1 - 2\mu - 2/p_1. \end{aligned}$$

□

Weak PRF A sequence of moduli $(p_1, \ldots, p_\tau)$ such that $t \cdot p_{i+1}/p_i = \mathsf{negl}(n)$.

Let secret key be $\boldsymbol{S} = (\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(\tau)})$, where $\boldsymbol{s}^{(\alpha)} \in \mathbb{Z}_{p_\alpha}^n$ for $\alpha \in [\tau]$.

On input $x = Z_{p_1}^n$,

1. $\boldsymbol{a}^{(\alpha)} \leftarrow x \mod p_\alpha \in \mathbb{Z}_{p_\alpha}^n$ for $\alpha \in [\tau]$ in parallel.
2. $r^{(\alpha)} \leftarrow \langle \boldsymbol{s}^{(\alpha)}, \boldsymbol{a}^{(\alpha)} \rangle \in Z_{p_\alpha}$ for $\alpha \in [\tau]$ in parallel.
3. $y^{(0)} \leftarrow 0$.
4. For $\alpha = 1, \ldots, \tau$ *in sequential*:
   (a) $e^{(\alpha)} \leftarrow \textsf{S-Ber}(y^{(\alpha-1)}) \in \mathbb{Z}_{p_1}$.
   (b) $y^{(\alpha)} \leftarrow r^{(\alpha)} + e^{(\alpha)} \in \mathbb{Z}_{p_\alpha}$.
5. Output $y \in \mathbb{Z}_{p_\tau}$.

Security: $n^{-\tau} + \tau \cdot t \cdot p_2/p_1$

Theoretical security: set $\tau = \omega(1)$, $t \cdot p_{i+1}/p_i = \mathsf{negl}(n)$. Evaluation Depth: $\omega(\log n)$.

Exponential security: set $\tau = n/\log n$, $t \cdot p_{i+1}/p_i = 2^{-n}$. Evaluation Depth $O(n)$.

Strong PRF A sequence of moduli $(p_0, p_1, \ldots, p_\tau)$ such that $t \cdot p_{i+1}/p_i = \mathsf{negl}(n)$.

Let $H\colon [m] \to \mathbb{Z}_{p_0}$ be an n-wise independent hash function.

Let secret key be $\boldsymbol{S} = (\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(\tau)})$, where $\boldsymbol{s}^{(\alpha)} \in \mathbb{Z}_{p_\alpha}^n$ for $\alpha \in [\tau]$.

On input $x \in [m]$.

1. $x' \leftarrow \lfloor H(x) \rceil_{p_0/p_1} \in \mathbb{Z}_{p_1}$.
2. Output weakPRF(x', $\boldsymbol{S}$).

References

[ABG+14] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in AC^0 o MOD$_2$. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 251–260, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery.

[ABW10] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing*, pages 171–180, Cambridge, MA, USA, June 5–8, 2010. ACM Press.

[ADI+17] 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, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 223–254, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[AIK08] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in nc0. *Computational Complexity*, 17:38–69, 2008.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[Al 01] A. Kh. Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 1–8, Cirencester, UK, December 17–19, 2001. Springer Berlin Heidelberg, Germany.

[Ale03a] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press.

[Ale03b] Michael Alekhnovich. More on average case vs approximation complexity. In *FOCS*, pages 298–307, 2003.

[AOW15] Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 689–708, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[AR16] Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 27–56, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany.

[BBTV24] Kiril Bangachev, Guy Bresler, Stefan Tiegel, and Vinod Vaikuntanathan. Near-optimal time-sparsity trade-offs for solving noisy linear equations. *arXiv preprint arXiv:2411.12512*, 2024.

[BCCD23] Maxime Bombar, Geoffroy Couteau, Alain Couvreur, and Clément Ducros. Correlated pseudorandomness from the hardness of quasi-abelian decoding. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 567–601, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[BCE$^+$24] Chris Brzuska, Geoffroy Couteau, Christoph Egger, Pihla Karanko, and Pierre Meyer. Instantiating the hash-then-evaluate paradigm: Strengthening prfs, pcfs, and oprfs. In *International Conference on Security and Cryptography for Networks*, pages 97–116. Springer, 2024.

[BCG$^+$19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 291–308, London, UK, November 11–15, 2019. ACM Press.

[BCG$^+$19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[BCG$^+$20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st Annual Symposium on Foundations of Computer Science*, pages 1069–1080, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press.

[BCG$^+$21] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Low-complexity weak pseudorandom functions in $\mathtt {AC}0[\mathtt {MOD}2]$. In *CRYPTO (4)*, volume 12828, pages 487–516. Springer, 2021.

[BCG$^+$22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 603–633, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 896–912, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[BDGJ20] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. Fast and secure updatable encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 464–493, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[BEKS20] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. Improving speed and security in updatable encryption schemes. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 559–589, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291, Santa Barbara, CA, USA, August 22–26, 1994. Springer Berlin Heidelberg, Germany.

[BHKN13] Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness preserving reductions via Cuckoo hashing. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 40–59, Tokyo, Japan, March 3–6, 2013. Springer Berlin Heidelberg, Germany.

[BIP$^+$18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos

Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland.

[BJKL21] Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 724–753, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd Annual ACM Symposium on Theory of Computing*, pages 435–440, Portland, OR, USA, May 21–23, 2000. ACM Press.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[BLVW19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for LPN and cryptographic hashing via code smoothing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 619–635, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[BMM24] Amos Beimel, Tal Malkin, and Noam Mazor. Structural lower bounds on black-box constructions of pseudorandom functions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part V*, volume 14924 of *Lecture Notes in Computer Science*, pages 459–488, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 353–370, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

[CD23] Geoffroy Couteau and Clément Ducros. Pseudorandom correlation functions from variable-density LPN, revisited. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13941 of *Lecture Notes in Computer Science*, pages 221–250, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland.

[CDK$^+$18] Benoît Cogliati, Yevgeniy Dodis, Jonathan Katz, Jooyoung Lee, John P. Steinberger, Aishwarya Thiruvengadam, and Zhe Zhang. Provable security of (tweakable) block ciphers based on substitution-permutation networks. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 722–753, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[CGHKV24] Henry Corrigan-Gibbs, Alexandra Henzinger, Yael Kalai, and Vinod Vaikuntanathan. Somewhat homomorphic encryption from linear homomorphism and sparse LPN. Cryptology ePrint Archive, Paper 2024/1760, 2024.

[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, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 502–534, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 501–520, Santa Barbara, CA, USA, August 20–24, 2006. Springer Berlin Heidelberg, Germany.

[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, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 315–348, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[DJ24] Quang Dao and Aayush Jain. Lossy cryptography from code-based assumptions. Cryptology ePrint Archive, Paper 2024/175, 2024.

[DS15] Nico Döttling and Dominique Schröder. Efficient pseudorandom functions via on-the-fly adaptation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 329–350, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

[DSSL16] Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of confusion-diffusion networks. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 679–704, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany.

[EKM17] Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 486–514, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[EPRS17] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. Key rotation for authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 98–129, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *34th Annual ACM Symposium on Theory of Computing*, pages 534–543, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

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

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000. https://eprint.iacr.org/2000/063.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HKL$^+$12] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-LPN. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 346–365, Washington, DC, USA, March 19–21, 2012. Springer Berlin Heidelberg, Germany.

[HRV10] Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing*, pages 437–446, Cambridge, MA, USA, June 5–8, 2010. ACM Press.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 433–442, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[IL90] Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science*, pages 812–821, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd Annual ACM Symposium on Theory of Computing*, pages 60–73, Virtual Event, Italy, June 21–25, 2021. ACM Press.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[JLS24] Aayush Jain, Huijia Lin, and Sagnik Saha. A systematic study of sparse LWE. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part III*, volume 14922 of *Lecture Notes in Computer Science*, pages 210–245, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[Kha93] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *25th Annual ACM Symposium on Theory of Computing*, pages 372–381, San Diego, CA, USA, May 16–18, 1993. ACM Press.

[Kim20] Sam Kim. Key-homomorphic pseudorandom functions from LWE with small modulus. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 576–607, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

[KLR19] Michael Klooß, Anja Lehmann, and Andy Rupp. (R)CCA secure updatable encryption with integrity protection. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 68–99, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.

[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 Annual ACM Symposium on Theory of Computing*, pages 132–145, Montreal, QC, Canada, June 19–23, 2017. ACM Press.

[KV89] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *21st Annual ACM Symposium on Theory of Computing*, pages 433–444, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[Lev85] Leonid A. Levin. One-way functions and pseudorandom generators. In *17th Annual ACM Symposium on Theory of Computing*, pages 363–365, Providence, RI, USA, May 6–8, 1985. ACM Press.

[LLHG21] Yiming Li, Shengli Liu, Shuai Han, and Dawu Gu. Pseudorandom functions in NC class from the standard LWE assumption. *Designs, Codes and Cryptography*, 89(12):2807–2839, 2021.

[LLPT23] Hanjun Li, Huijia Lin, Antigoni Polychroniadou, and Stefano Tessaro. LERNA: Secure single-server aggregation via key-homomorphic masking. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part I*, volume 14438 of *Lecture Notes in Computer Science*, pages 302–334, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany.

[LPTV23] Tianren Liu, Angelos Pelecanos, Stefano Tessaro, and Vinod Vaikuntanathan. Layout graphs, random walks and the t-wise independence of SPN block ciphers. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023,* *Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 694–726, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[LT18] Anja Lehmann and Björn Tackmann. Updatable encryption with post-compromise security. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 685–716, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[LTV21] Tianren Liu, Stefano Tessaro, and Vinod Vaikuntanathan. The t-wise independence of substitution-permutation networks. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 454–483, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland.

[LW09] Allison B. Lewko and Brent Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 2009: 16th Conference on Computer and Communications Security*, pages 112–120, Chicago, Illinois, USA, November 9–13, 2009. ACM Press.

[MBD$^+$18] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Trans. Inf. Theory*, 64(5):3927–3943, 2018.

[MTSB13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In *Proceedings of the International Symposium on Information Theory*, pages 2069–2073, Istanbul, Turkey, July 7-12 2013. IEEE.

[MV12] Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 68–85, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346, Prague, Czech Republic, May 2–6, 1999. Springer Berlin Heidelberg, Germany.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, October 19–22, 1997. IEEE Computer Society Press.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *J. Comput. Syst. Sci.*, 58(2):336–375, 1999.

[NRR02] Moni Naor, Omer Reingold, and Alon Rosen. Pseudorandom functions and factoring. *SIAM J. Comput.*, 31(5):1383–1404, 2002.

[Pra62] E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part IV*, volume 15367 of *Lecture Notes in Computer Science*, pages 3–36, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[Sch18] Peter Scholl. Extending oblivious transfer with low communication via key-homomorphic PRFs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 554–583, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Cham, Switzerland.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949.

[Val84] Leslie G. Valiant. A theory of the learnable. In *16th Annual ACM Symposium on Theory of Computing*, pages 436–445, Washington, DC, USA, 1984. ACM Press.

[Vio13] Emanuele Viola. The communication complexity of addition. In Sanjeev Khanna, editor, *24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 632–651, New Orleans, LA, USA, January 6–8, 2013. ACM-SIAM.

[VZ12] Salil P. Vadhan and Colin Jia Zheng. Characterizing pseudoentropy and simplifying pseudorandom generator constructions. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 817–836, New York, NY, USA, May 19–22, 2012. ACM Press.

[YS16] Yu Yu and John P. Steinberger. Pseudorandom functions in almost constant depth from low-noise LPN. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 154–183, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany.

[Zha24] Mark Zhandry. Optimal traitor tracing from pairings. Cryptology ePrint Archive, Paper 2024/867, 2024.