Exploring Crypto Dark Matter:
New Simple PRF Candidates and Their Applications*

Dan Boneh† Yuval Ishai‡ Alain Passelègue§ Amit Sahai§ David J. Wu†

Abstract

Pseudorandom functions (PRFs) are one of the fundamental building blocks in cryptography. We explore a new space of plausible PRF candidates that are obtained by mixing linear functions over different small moduli. Our candidates are motivated by the goals of maximizing simplicity and minimizing complexity measures that are relevant to cryptographic applications such as secure multiparty computation.

We present several concrete new PRF candidates that follow the above approach. Our main candidate is a *weak* PRF candidate (whose conjectured pseudorandomness only holds for uniformly random inputs) that first applies a secret mod-2 linear mapping to the input, and then a public mod-3 linear mapping to the result. This candidate can be implemented by depth-2 ACC^0 circuits. We also put forward a similar depth-3 *strong* PRF candidate. Finally, we present a different weak PRF candidate that can be viewed as a deterministic variant of "Learning Parity with Noise" (LPN) where the noise is obtained via a mod-3 inner product of the input and the key.

The advantage of our approach is twofold. On the theoretical side, the simplicity of our candidates enables us to draw natural connections between their hardness and questions in complexity theory or learning theory (e.g., learnability of depth-2 ACC^0 circuits and width-3 branching programs, interpolation and property testing for sparse polynomials, and natural proof barriers for showing super-linear circuit lower bounds). On the applied side, the "piecewise-linear" structure of our candidates lends itself nicely to applications in secure multiparty computation (MPC). Using our PRF candidates, we construct protocols for distributed PRF evaluation that achieve better round complexity and/or communication complexity (often both) compared to protocols obtained by combining standard MPC protocols with PRFs like AES, LowMC, or Rasta (the latter two are specialized MPC-friendly PRFs). Our advantage over competing approaches is maximized in the setting of MPC with an honest majority, or alternatively, MPC with preprocessing.

Finally, we introduce a new primitive we call an *encoded-input PRF*, which can be viewed as an interpolation between weak PRFs and standard (strong) PRFs. As we demonstrate, an encoded-input PRF can often be used as a drop-in replacement for a strong PRF, combining the efficiency benefits of weak PRFs and the security benefits of strong PRFs. We conclude by showing that our main weak PRF candidate can plausibly be boosted to an encoded-input PRF by leveraging error-correcting codes.

*This is a preliminary full version of [BIP+18]

†Stanford University. Email: {dabo, dwu4}@cs.stanford.edu

‡Technion. Email: yuvali@cs.technion.ac.il. Work done in part while visiting UCLA.

§UCLA. Email: alain.passelegue@inria.fr, sahai@cs.ucla.edu

1 Introduction

In this paper, we continue a line of work on constructing low-complexity pseudorandom functions. We explore a new space of simple candidate constructions that enjoy several advantages over existing candidates. We start with some relevant background.

As discussed in [ABG+14], there are two primary paradigms for designing cryptographic primitives. The "theory-oriented" or "provable security" approach is to develop constructions whose security can be *provably* reduced to the hardness of well-studied computational problems (e.g., factoring, discrete log, or learning with errors). The second and "practice-oriented" approach aims at obtaining efficient constructions for specific functionalities (e.g., block ciphers or hash functions). Here, designers typically try to maximize concrete efficiency at the expense of relying on heuristic arguments and prior experience to argue security. But ultimately, confidence in the underlying security assumptions or cryptographic designs only grows if they withstand the test of time.

There are several limitations to these approaches. On the one hand, both the efficiency and the structure of provably-secure constructions are inherently limited by the underlying computational problems. This leads to constructions that are far less efficient than those obtained from the practice-oriented approach. On the other hand, despite the efficiency of practical constructions, their designs are often complex, thereby complicating their analysis. Consequently, it is difficult to argue whether the lack of cryptanalysis against practical constructions is due to their actual security or due to the complexity of their design. The structure of both types of constructions often makes them poorly suited as building blocks for cryptographic applications that are different from the ones envisioned by their designers (e.g., secure multiparty computation).

In this work, we depart from these traditional approaches and consider a surprisingly unexplored space of cryptographic constructions. Our approach is driven by *simplicity*, and aims at circumventing some of the limitations of the existing approaches. Our hope is to obtain constructions that are (1) relatively easy to describe and analyze, (2) concretely efficient, and (3) well-suited for different applications. In particular, we aim at relying on assumptions that are *simple* to state, and yet at the same time, breaking them would likely require new techniques that may themselves have other applications. In a sense, the assumptions we introduce have a win-win flavor and can be of independent interest beyond the cryptographic community (e.g., to complexity theorists, learning theorists, or mathematicians). A notable example for prior work in this direction is Goldreich's proposal of a simple one-way function candidate [Gol00], which had an unexpected impact in different areas of cryptography and beyond (see [App13] for a survey). More closely relevant to this work, the works of Miles and Viola [MV12] and (especially) Akavia et al. [ABG+14] proposed heuristic constructions of simple pseudorandom functions and proved their security against natural classes of attacks, without reducing their security to any previously studied assumption.

What do we mean by simplicity? The concrete direction we take is exploring whether the idea of mixing *linear functions* over *different moduli* can be a source of hardness in the context of secret-key cryptographic primitives. Our starting observation is that computing the sum of m binary-valued variables modulo 3 is actually a *high-degree* polynomial over $\mathbb{Z}_2$. More precisely, the mapping function $\mathsf{map}\colon \{0,1\}^m \to \mathbb{Z}_3$ where $\mathsf{map}(x) \coloneqq \sum_{i\in[m]} x_i \pmod 3$ is a polynomial of high-degree over the *binary* field $\mathbb{Z}_2$ (but a simple linear function over $\mathbb{Z}_3$). Surprisingly, this simple

idea of mixing different small moduli enables new constructions of "piecewise-linear[1]" symmetric primitives that are conceptually simple to describe, can plausibly achieve strong security guarantees, and minimize complexity measures that are relevant to natural cryptographic applications.

Our focus: pseudorandom functions. In this work, we focus specifically on pseudorandom functions (PRFs) [GGM86]—one of the most fundamental building blocks of modern cryptography. Our primary focus is on *weak* pseudorandom functions: namely, functions whose behavior looks indistinguishable from that of a random function to any adversary who only observes the input-output behavior of the function on *random* domain elements. Since weak PRFs cannot replace standard (or strong) PRFs in all cryptographic applications, we then show how our construction can be adapted to yield a new primitive we call an *encoded-input* PRF. An encoded-input PRF is defined similarly to a standard (strong) PRF, except that its input domain is restricted to an efficiently recognizable set. Encoded-input PRFs can be viewed as an intermediate primitive between strong PRFs and weak PRFs that combines the security advantages of the former and efficiency advantages of the latter. Indeed, we show that in many cases they can be used as a replacement for a strong PRF. At the same time, we exhibit simple candidates of encoded-input PRFs in complexity classes where strong PRFs are not known to exist. Finally, a unique feature of our new PRF candidates is that they are very "MPC-friendly." As we show in Section 6, in some natural settings of secure computation, our PRFs can be computed more efficiently in a distributed fashion compared to standard block ciphers like AES and even custom-built MPC-friendly block ciphers like LowMC [ARS+15] or Rasta [DEG+18].

Previous work on simple PRFs. Before describing our contributions, it is useful to survey some closely relevant previous works on low-depth PRFs (see Sections 1.2 and 3.2 for a broader survey). We denote by AC^0 the class of polynomial-size, constant-depth circuits with unbounded fan-in AND, OR, and NOT gates and by $\mathsf{ACC}^0[m]$ the class of such circuits that can additionally have unbounded fan-in MOD_m gates, which return 0 or 1 depending on whether the sum of their inputs is divisible by m. We denote by ACC^0 the union over all m of $\mathsf{ACC}^0[m]$.

With the goal of minimizing the depth complexity of weak PRFs, Akavia et al. [ABG+14] proposed the first candidate that can be computed by $\mathsf{ACC}^0[2]$ circuits. More precisely, their candidate construction can be computed by depth-3 circuits in which the first layer consists of MOD_2 gates computing a matrix-vector product $\mathbf{A}x$, where $\mathbf{A} \in \mathbb{Z}_2^{n \times n}$ is the secret key and $x \in \mathbb{Z}_2^n$ is the input. The second and third layer define a public DNF formula. While the Akavia et al. candidate could plausibly provide exponential security,[2] Bogdanov and Rosen [BR17] recently showed that this candidate (on n-bit inputs) can be computed by a rational function of degree $O(\log n)$, which in turn gives rise to a quasi-polynomial time attack. Their work also raises the question of coming up with an explicit function g for which $g(\mathbf{A}x+b)$ is a weak PRF with better than quasi-polynomial security. Applebaum and Raykov [AR16] show that low-complexity PRFs can be based on one-wayness assumptions. In particular, under a variant of Goldreich's one-wayness assumption [Gol00], they present a weak PRF with quasi-polynomial security that can be implemented (on any fixed key) by depth-3 AC^0 circuits.

[1] In our context, we use the term "piecewise-linear" to refer to the fact that our pseudorandom function candidates can be expressed as a *composition* of linear functions over different moduli.

[2] Roughly speaking, we say that a weak PRF is *exponentially* secure if the distinguishing advantage of any adversary (modeled as a Boolean circuit) of size 2^λ is bounded by $2^{-\Omega(\lambda)}$.

These recent results leave several open questions regarding the complexity of low-depth (weak) PRFs. First, even if one settles for quasi-polynomial time security, there is no proposed PRF candidate of any kind that can be realized by *depth-2* circuits over any standard basis. When restricting attention to (weak) PRFs that offer a better level of security, the situation is even worse. While it is known that weak PRFs with better than quasi-polynomial security do not exist in AC^0,[3] and that strong PRFs with similar security do not exist in $\mathsf{ACC}^0[p]$ for any prime p,[4] it is plausible that weak PRFs with *exponential* security could still exist in $\mathsf{ACC}^0[2]$. But to the best of our knowledge, no such candidates have been suggested. If we do settle for quasi-polynomial security, then the result of Kharitonov [Kha93, Theorem 9] (resp., Viola [Vio13, Theorem 11]) gives a weak PRF in AC^0 (resp., strong PRF in $\mathsf{ACC}^0[p]$ for any p) based on the hardness of factoring. This raises the question of whether it is possible to construct (weak or strong) PRFs with exponential (or even just better than quasi-polynomial) security in ACC^0. In this work, we propose a new candidate weak PRF that can be computed by depth-2 ACC^0 circuits. Our candidate is conceptually simple and can plausibly satisfy exponential security, thus addressing both of the above challenges simultaneously. We also propose other variants of this candidate, including a candidate for an exponentially secure strong PRF that can be computed by depth-3 ACC^0 circuits. We provide a comparison of the known positive and negative results for weak and strong PRFs in different complexity classes in Table 1.

	Complexity Class		
Circuit Depth	AC^0	$\mathsf{ACC}^0[p]$	$\mathsf{ACC}^0[m]$
Depth 2			**Weak PRF (§3) (exponential)**
Depth 3	Weak PRF [AR16] (quasi-polynomial)	Weak PRF [ABG+14] (quasi-polynomial)	**Strong PRF (§7.3) (exponential)**
Depth > 3	Weak PRF [Kha93] (quasi-polynomial)	Strong PRF [Vio13] (quasi-polynomial)	
Lower Bound	No weak PRF with better than quasi-polynomial security [LMN89]	No strong PRF with better than quasi-polynomial security [CIKK16]	

Table 1: Comparison of positive and negative results for low-complexity PRF candidates. We write $\mathsf{ACC}^0[p]$ to denote the class AC^0 with MOD_p gates for a *prime* p and $\mathsf{ACC}^0[m]$ to denote the class AC^0 with MOD_m gates for any integer m. For each candidate, we denote in parenthesis their security (i.e., quasi-polynomial security or exponential security). The entries shown in **bold** (the right-hand column) are from this work.

[3] Specifically, the classic learning result of Linial et al. [LMN89] showed that AC^0 circuits can be learned from *random* examples in quasi-polynomial time.

[4] The recent learning result by Carmosino et al. [CIKK16] showed that for any prime p, $\mathsf{ACC}^0[p]$ circuits can be learned using membership queries in quasi-polynomial time. Extending this result to the setting of learning from uniformly random examples (without membership queries) or to composite moduli seems challenging.

1.1 Our Contributions

In this section, we give a more detailed overview of the main results of this paper.

New weak PRF candidates. We put forward several new (weak) PRF candidates that mix linear functions over different moduli. We start by describing our most useful candidate, and will discuss other variants later. Our primary weak PRF candidate follows a very similar design philosophy as that taken by Akavia et al. [ABG+14]. Recall first that in the Akavia et al. construction, the secret key is a matrix $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ and the input is a vector $x \in \mathbb{Z}_2^n$. The output of the PRF is defined as $\mathsf{F}_{\mathbf{A}}(x) \coloneqq g(\mathbf{A}x)$, where the function g is a non-linear mapping (in the case of the Akavia et al. construction, the function g is a "tribes" function and can be expressed as a DNF formula). In our setting, we adopt the same high-level structure, but substitute a different and conceptually simpler non-linear function g. In our candidate, we define the non-linear function to be the function that interprets the binary outputs of $\mathbf{A}x$ as 0/1 values over $\mathbb{Z}_3$, and the output of the function is simply the sum of the input values over $\mathbb{Z}_3$. This gives a simple candidate answer to the aforementioned open question of Bogdanov and Rosen [BR17].

Specifically, we define the mapping function $\mathsf{map}\colon \{0,1\}^m \to \mathbb{Z}_3$ that maps $y \in \{0,1\}^m \mapsto \sum_{i \in [m]} y_i \pmod 3$. Our weak PRF candidate (with key $\mathbf{A}$) is then defined as

$$\mathsf{F}_{\mathbf{A}}(x) \coloneqq \mathsf{map}(\mathbf{A}x) \ . \tag{1.1}$$

We formally introduce our candidate (and discuss several generalizations[5]) in Section 3. We state our formal conjectures regarding the hardness of our candidate in Section 3.1. There are several properties of our weak PRF candidate that we want to highlight:

- **Conceptual simplicity.** Our candidate is conceptually very simple to describe. It reduces to computing a (secret) matrix-vector product over $\mathbb{Z}_2$, reinterpreting the output vector as a 0/1 vector mod-3 and then computing the sum of its components. The simplicity of our construction is fairly apparent compared to block cipher candidates like AES or number-theoretic constructions of PRFs. In spite of its simplicity, to the best of our knowledge, such a candidate has not previously been proposed, let alone studied.

- **Low complexity.** Our candidate can be computed by *depth-2* $\mathsf{ACC}^0[2,3]$ circuits. More precisely, the first layer consists entirely of MOD_2 gates to compute the matrix-vector product $\mathbf{A}x$, and the second layer consists of two MOD_3 gates that computes the binary representation of the output. We refer to Remark 3.10 for a more precise definition.

- **MPC friendliness.** The simplicity of our candidate also lends itself nicely for use in MPC protocols. In Section 6, we give an efficient protocol that enables *distributed* evaluation of our PRF in a setting where both the key and the input are secret-shared. We discuss this further in the sequel. As we show in Table 2 and Table 3, the round complexity and communication complexity of our distributed evaluation protocol outperform existing MPC protocols for distributed evaluation of not only AES, but even those for MPC-friendly block ciphers like LowMC [ARS+15] and Rasta [DEG+18]. This applies both to the 3-party case with one corrupted party and (especially) to the case of secure 2-party computation with preprocessing.

[5] An immediate generalization is replacing 2 and 3 by different numbers. However, the particular choice of 2 and 3 turns out to be the most useful for our purposes. A more useful generalization replaces the above choice of map by a suitable compressive mod-3 linear mapping, which yields weak PRF candidates with a longer output.

Cryptanalysis. In Section 4, we consider several classic cryptanalytic techniques on our weak PRF candidate. While our analysis is by no means exhaustive, we are able to rule out several classes of attacks, thereby providing some confidence into the security of our new candidate. Following the work of Akavia et al. [ABG+14], we focus on two primary classes of attacks:

- **Lack of correlation with fixed function families.** First, we rule out the learning-type attacks of Linial et al. [LMN89] by showing that there are no *fixed* function families of *exponential* size that are noticeably correlated with our PRF candidate (previously, Linial et al. showed that for all AC^0 functions, there exists a quasi-polynomial-size function family such that any AC^0 function is noticeably correlated with a function in that class; this implies a quasi-polynomial time learning algorithm for AC^0).
- **Inapproximability by low-degree polynomials.** Next, we show that there does not exist a low-degree polynomial approximation to our PRF candidate. Our argument here follows from the well-known Razborov-Smolensky lower bounds [Raz87, Smo87] for ACC^0 circuits, which say that for distinct primes p, q, the MOD_p function cannot be computed (or even approximated) by a polynomial-size circuit in $\mathsf{ACC}^0[q]$. We conjecture that the Razborov-Smolensky lower bounds also generalize to rule out low-degree *rational* approximations. Namely, for distinct primes p, q, there does not exist a low-degree rational function that approximates MOD_p gates sufficiently well over $\mathrm{GF}(q^\ell)$ for any ℓ (Conjecture 4.3). We believe that this question is of independent interest from a complexity-theoretic perspective, and leave it as an interesting challenge.

Given the above, we conjecture that our main weak PRF candidate is exponentially secure. We hope that our exploratory analysis will encourage further study and refinement of our conjectures.

Additional PRF candidates. In addition to our main weak PRF candidate (Eq. (1.1)), we also propose a similar strong PRF candidate and an alternative weak PRF candidate.

- **Strong PRF candidate in depth-3 ACC^0.** Our weak PRF candidate from Eq. (1.1) is not a strong PRF, and as we discuss later in this section and in Section 5.3, there is a *non-adaptive* attack against our candidate. Moreover, in Appendix B.1, we describe a more general adaptive attack that rules out many natural strong PRF constructions in depth-2 ACC^0. However, the existing attacks do not seem to extend to depth-3 ACC^0, and in Section 7.3, we propose a strong PRF candidate in depth-3 ACC^0 that relies on the same general technique of mixing linear operations over different moduli (Construction 7.9, Remark 7.14).

  Our depth-3 strong PRF candidate is obtained by first applying a *public* random mod-3 linear mapping to the input, taking the binary decomposition of the resulting vector (to obtain a mod-2 vector), and then evaluating our weak PRF candidate F (Equation 1.1) on the decomposed mod-2 vector. Essentially, we can view the initial mod-3 mapping and binary decomposition as computing a public "encoding" of the input. The strong PRF candidate is then essentially our weak PRF applied to the encoding of the input. The overall PRF computation thus consists of a mod-3/mod-2/mod-3 computation, where the mod-3 mappings are public and the mod-2 mapping is secret.

  Specifically, let $\mathbf{G} \in \mathbb{Z}_3^{n' \times n}$ be a fixed public matrix, $\mathsf{bin}\colon \mathbb{Z}_3^{n'} \to \{0,1\}^{2n'}$ be the component-wise binary decomposition function (that maps each $\mathbb{Z}_3$ component into two bits corresponding to the binary representation of the component). The PRF key is a matrix $\mathbf{A} \in \mathbb{Z}_2^{m \times 2n'}$

and on input $x \in \{0,1\}^n$, the output is

$$\mathsf{F}'_{\mathbf{A}}(x) := \mathsf{F}_{\mathbf{A}}(\mathsf{bin}(\mathbf{G} \cdot x)) \ , \tag{1.2}$$

where $\mathsf{F}_{\mathbf{A}}$ is our weak PRF candidate from Eq. (1.1). To the best of our knowledge, this is the first strong PRF candidate computable by a depth-3 circuit that plausibly provides exponential (or even better than quasi-polynomial) security. We discuss this candidate and its applications in Section 7.3.

- **An alternative weak PRF candidate.** As we discuss below (and in Section 6), the structure of our main candidate enables efficient protocols for distributed evaluation in several standard MPC settings (specifically, the honest majority setting and the MPC with preprocessing setting). In other settings such as the two-party setting, it is natural to rely on a "garbling scheme" such as that of Yao [Yao86] or its optimized variants. However, when applied to our candidate (Eq. (1.1)), the cost of this approach will be high because of the super-linear number of multiplications needed for computing the matrix-vector product. In Section 6.5, we introduce an alternative weak PRF candidate (Construction 6.3) that is more suitable for two-party distributed evaluation.

  The secret key in our alternative candidate is a vector $\mathbf{k} \in \{0,1\}^n$, and on input $x \in \{0,1\}^n$, the output is defined to be

$$\mathsf{F}_{\mathbf{k}}(x) := \sum_{i \in [n]} \mathbf{k}_i x_i \bmod 2 + \sum_{i \in [n]} \mathbf{k}_i x_i \bmod 3 \pmod 2 \ . \tag{1.3}$$

  This construction can be viewed as a deterministic LPN instance with noise rate $1/3$, where the noise is generated via a deterministic, key-dependent, and input-dependent computation. Namely, the noise is 1 if and only if $\langle \mathbf{k}, x \rangle = 1 \bmod 3$. Equivalently, $\mathsf{F}_{\mathbf{k}}(x) = 1$ if and only if $\langle \mathbf{k}, x \rangle \bmod 6 \in \{3,4,5\}$, which corresponds to a special instance of the learning with rounding (LWR) assumption with *constant-size composite* modulus. We note that using a composite modulus in this setting is critical for security in the constant-modulus regime, since otherwise there is a direct polynomial-time linearization attack (e.g., [AG11]) on the scheme. An advantage of this candidate over our main candidate is that it outputs unbiased *bits* (rather than elements of $\mathbb{Z}_3$). On the downside, this candidate falls short of providing full exponential security because (similarly to LPN), it is susceptible to BKW-style attacks. We discuss this alternative weak PRF candidate (as well as a two-party distributed evaluation protocol for computing it) in Section 6.5.

Theoretical implications. We next turn to studying the implications and applications of our new PRF candidates. Unless stated otherwise, we refer here to our main depth-2 weak PRF candidate. We first describe several theoretical implications related to complexity theory and learning theory that are implied by our conjectures:

- **Hardness of learning for depth-2 ACC^0 and width-3 branching programs.** As mentioned earlier, one of the key structural properties of our weak PRF candidate is that it can be computed by a depth-2 ACC^0 circuit. Another low-complexity feature, which crucially depends on the choice of the moduli 2 and 3, is that it can be computed by (polynomial-length) *width-3 permutation branching programs* [Bar85]. The existence of a weak PRF in

any complexity class rules out learning algorithms for that class even with uniformly random examples (but *without* membership queries). This means that, assuming the exponential security of our weak PRF candidate in Eq. (1.1), the classes of depth-2 ACC^0 circuits and width-3 permutation branching programs are not learnable (in the standard sense of PAC-learnability [Val84] *without* membership queries), even under the uniform distribution and even when allowing sub-exponential time learning algorithms. We explore these connections in greater detail in Sections 5.1 and 5.2. We note that efficient learning algorithms for the above classes would imply an efficient learning algorithm for DNF formulas [EKR95]. While there are quasi-polynomial time learning algorithms for DNF formulas (in fact, even for AC^0 circuits) under the uniform distribution [LMN89, Ver90], no such learning algorithm (even a sub-exponential one) is known for depth-2 ACC^0 or width-3 branching programs.

- **Hardness of interpolating and property-testing sparse polynomials.** In Section 5.3, we give an alternative characterization of Eq. (1.1) as essentially implementing a *sparse* multilinear polynomial over $\mathbb{Z}_3$, where the monomials are determined by the key $\mathbf{A}$. We then show that the conjectured hardness of our weak PRF candidate implies that sparse polynomials over $\mathbb{Z}_3$ (with sufficient degree and sparsity) are hard to interpolate given *random* evaluations drawn from a subset of the domain, namely from $\{-1, 1\}^n$. Similar to the previous connections to hardness of learning, if it is easy to interpolate the polynomial corresponding to the operation of the PRF (on random inputs), then the interpolation algorithm gives a trivial distinguisher for the scheme. While the problem of sparse polynomial interpolation has been the subject of extensive study [Zip79, BOT88, KY88, Zip90, Wer94, GS09, AGR14], much less is known when the interpolation algorithm only sees random evaluations from a subset of the domain. Our conjectures imply hardness results for this variant of the sparse interpolation problem. In fact, as we show in Remark 5.11, our conjectures even rule out property-testing algorithms [PRS02, AKK+03, JPRZ04, DLM+07] for sparse polynomials.

- **Natural proofs barrier for super-linear circuit lower bounds.** Our work also has relevance to minimizing the *sequential time complexity* or *circuit size* of *strong* PRFs. We consider the problem of constructing "asymptotically optimal" strong PRFs, namely ones that have exponential security in the input length and can be computed by linear-size circuits. This problem is motivated by the goal of ruling out *natural proofs* of super-linear circuit lower bounds, in the sense of Razborov and Rudich [RR94]. While previous works constructed PRFs that can be evaluated by linear-size circuits [IKOS08] or in linear time on a RAM machine [AR16], these PRFs fail to achieve full exponential security. The work of Miles and Viola [MV12] presented a simplified abstraction of existing block cipher designs and proved their security under a class of natural attacks. One of their constructions can be implemented by *quasi-linear* size circuits and is shown to have *exponential security* against a wide class of attacks, thus falling a bit short of the asymptotic optimality goal. Our depth-3 strong PRF candidate from Eq. (1.2) (with a suitable instantiation of the public matrix $\mathbf{G}$ described in Remark 7.15) yields a concrete candidate that can plausibly meet this goal. Thus, we give the first candidate construction for an asymptotically optimal strong PRF, which in turn rules out natural proofs of super-linear circuit lower bounds.

Applications to MPC and distributed PRF evaluation. A particularly appealing property of our weak PRF candidate is that it is very MPC-friendly. Protocols for PRF evaluation in a

distributed setting (where the secret key and input are distributed or secret-shared between two or more parties) have received a significant amount of attention recently, and new block ciphers have been proposed specifically to be MPC-friendly [ARS+15, DEG+18]. The structure of our weak PRF lends itself nicely to an efficient MPC protocol (with semi-honest security) for evaluating the PRF with a secret-shared key and a secret-shared input. Consider a scenario where the PRF key and input are secret-shared across multiple servers. Our protocol proceeds roughly as follows:

- If we use a *linear* secret-sharing scheme to share the keys and the inputs over $\mathbb{Z}_2$ (alternatively, a field of characteristic 2), then the matrix-vector product $\mathbf{A}x$ can be computed *non-interactively*: each party simply operates locally on their shares (of the key and input).[6]

- Next, the servers engage in a simple interactive protocol to convert their secret-shared values (over $\mathbb{Z}_2$) to a linear secret-sharing of the same value over $\mathbb{Z}_3$ (effectively implementing the non-linear step in our PRF). Working in the 3-server setting (in a semi-honest model tolerating at most one corruption), we can implement this protocol very efficiently using the protocol of Araki et al. [AFL+16]. Here, the "share conversion" procedure essentially requires 13 bits of communication for each bit of $\mathbf{A}x$.

- Once the parties have a linear secret-sharing of $\mathbf{A}x$ over $\mathbb{Z}_3$, computing the output can again be done non-interactively. Note that to extend our weak PRF candidate to output multiple bits, we replace the summation over $\mathbb{Z}_3$ with a matrix-vector product. Namely if $y \leftarrow \mathbf{A}x \in \{0,1\}^m$, then we define the PRF output to be $\mathbf{G}y \pmod 3$, where $\mathbf{G}$ here is a fixed *public* matrix in $\mathbb{Z}_3^{t \times m}$ (Remark 3.3). Even with this extension, computing the output (given a $\mathbb{Z}_3$ secret-sharing of the values $\mathbf{A}x$) still corresponds to computing a *linear* function over $\mathbb{Z}_3$. Again, this is possible non-interactively.

The takeaway is that even though our weak PRF candidate is *highly nonlinear* (due to the mixing of mod-2 and mod-3 operations), the piecewise-linear structure means that it can be securely computed by a constant-round *information-theoretic* MPC protocol with $O(|x|)$ bits of communication. In Table 2, we provide some *concrete* comparisons of our protocol for distributed evaluation of our PRF candidate to some of the existing candidates. As the baseline for our comparisons, we use the protocol of Araki et al. [AFL+16] as the representative for 3-party secret-sharing-based MPC protocols, and optimized garbled circuit constructions [KS08, ZRE15] for 2-party protocols. We compare against both the AES block cipher as well as several settings of LowMC [ARS+15] and Rasta [DEG+18], two custom-designed block ciphers tailored for MPC applications. We describe our precise methodology for deriving these estimates in Section 6.2.

From Table 2, we see that using an optimistic setting of parameters for our candidate, the communication and round complexity of our 3-server protocol for distributed (weak) PRF evaluation is better than the generic MPC protocols applied to existing (strong) PRF candidates in terms of both round complexity and communication complexity in almost all cases. The only case where another protocol has smaller communication complexity is the case of evaluating the AND-gate-optimized variant of LowMC (using the Araki et al. protocol); however, evaluating this variant of LowMC requires over 250 rounds of communication compared to the 2 rounds needed for our protocol.

[6] More precisely, one needs here a linear secret-sharing scheme that supports multiplication. In our 3-server implementation we use replicated additive shares (also known as "CNF secret-sharing") to achieve this. We refer to Section 6.1 for the full details.

Construction	Number of Servers	Round Complexity	Communication Complexity
Araki et al. (AES)	3	40	$\approx 1.6 \cdot 10^4$
Araki et al. (LowMC, min-depth)	3	14	$\approx 7.9 \cdot 10^3$
Araki et al. (LowMC, min-gates)	3	252	$\approx 2.3 \cdot 10^3$
Araki et al. (Rasta, min-depth)	3	2	$\approx 2.6 \cdot 10^{10}$
Araki et al. (Rasta, min-gates)	3	6	$\approx 6.3 \cdot 10^3$
Garbled Circuit (AES)	2	2	$\approx 1.4 \cdot 10^6$
Garbled Circuit (LowMC, min-gates)	2	2	$\approx 1.9 \cdot 10^5$
Garbled Circuit (Rasta, min-gates)	2	2	$\approx 5.4 \cdot 10^5$
Our Protocol (Optimistic)	3	2	$\approx 3.8 \cdot 10^3$
Our Protocol (Conservative)	3	2	$\approx 5.5 \cdot 10^3$
Our Protocol (General)	3	2	$13n + 4t$

Table 2: Comparison of semi-honest oblivious PRF evaluation protocols. In all cases, we assume that the keys and inputs have been secret-shared between the (2 or 3) servers. We estimate the round complexity and the *total* communication complexity (in bits) needed to evaluate the PRF on the shared key and input. All of our comparisons assume semi-honest servers with up to one corruption and assuming a concrete security parameter of $\lambda = 128$. When comparing to the LowMC block cipher [ARS+15] and the Rasta block cipher [DEG+18], we compare against two variants: a depth-optimized variant (min-depth) that minimizes the multiplicative depth of the circuit implementing the block cipher, and a gates-optimized variant (min-gates) that minimizes the number of AND gates. We refer to Section 6.2 for the parameter settings we use for our estimates. For our protocol, we set the dimensions m, n according to our concrete parameter estimates from Table 4 (in particular we let $m = n$), and set the output dimension to be $t = 128$ (for output space $\mathbb{Z}_3^{128}$).

Compared to the communication-intensive protocols based on garbled circuits, the communication complexity of our protocol is roughly two orders of magnitude smaller than garbled circuit evaluation of LowMC and Rasta, and three orders of magnitude smaller than garbled circuit evaluation of AES. The secret-sharing-based protocols are much more competitive in terms of communication, but these protocols generally have much larger round complexities, which can be problematic in high-latency networks. To summarize, our new PRFs have the advantage that they are very friendly to compute in a distributed MPC setting when both the key and the input are secret-shared. We note that even *weak* PRFs are still useful in a variety of application scenarios. In Section 6.6 we describe a concrete application of MPC-friendly weak PRFs for implementing distributed flavors of secure keyword search and searchable symmetric encryption. Moreover, for applications that require strong PRFs, one can apply the *encoded-input* variant of our weak PRF with a modest loss of efficiency (see Section 7).

Distributed PRF evaluation in the preprocessing model. The piecewise-linear structure of our weak PRF enables even more savings if we consider the MPC with preprocessing model [Bea92,

Construction	Round Complexity	Output Bits	Online Communication	Preprocessing Size
Yao + AES	2	128	$6.6 \cdot 10^4$	$1.5 \cdot 10^6$
Yao + LowMC	2	128	$6.6 \cdot 10^4$	$2.9 \cdot 10^5$
Yao + Rasta	2	351	$1.8 \cdot 10^5$	$8.1 \cdot 10^5$
Our Protocol	4	128	$2.6 \cdot 10^3$	$3.5 \cdot 10^3$

Table 3: Comparison of protocols for two-party fully-distributed PRF evaluation in the *preprocessing* model. We measure the online round complexity, the online communication (in bits), and the size of the correlated randomness (in bits) for the different protocols. We use Yao's two-party protocol as the representative protocol for evaluating existing block ciphers such as AES, LowMC, and Rasta. We refer to Section 6.3 for a complete description of how the estimates were computed. For our protocol, we set the dimensions $m = n = 256$ according to our concrete parameter estimates from Table 4, and assume that the key $\mathbf{A}$ is a block-circulant matrix (and in particular, can be represented by a single vector (Remark 3.4).

BDOZ11, DPSZ12, IKM+13]. In this model, prior to the computation, a trusted dealer distributes some input-independent randomness to the computing parties. This correlated randomness can significantly reduce the online complexity (both computation and communication) of the protocol execution. The piecewise-linear structure of our weak PRF candidate makes it very amenable for fully distributed evaluation in the preprocessing model, and we give efficient protocols for both fully distributed evaluation as well as the closely related problem of oblivious PRF evaluation [FIPR05] in Sections 6.3 and 6.4. In Table 3, we compare the cost of two-party fully distributed PRF evaluation of our weak PRF candidate to the costs of generic Yao-based protocols for evaluating alternative PRF candidates in the preprocessing model.

Compared to the generic approaches for fully-distributed evaluation, the online communication complexity of our protocol is over 25x smaller than the generic approach (applied to existing PRF candidates like AES as well as MPC-friendly PRFs like LowMC and Rasta). In terms of the amount of correlated randomness needed, the gap is even greater (due to the large sizes of the garbled circuits that need to be stored). Compared to the generic protocol for distributed evaluation of LowMC, the amount of preprocessing needed for distributed evaluation of our candidate is over 80x smaller (with even larger improvements when comparing to AES or Rasta). One disadvantage of our protocol is that it requires 4 rounds while the generic approaches give a 2 round protocol.

Garbling our alternative weak PRF candidate. As mentioned before, the cost of distributed evaluation of our main candidate (Eq. (1.1)) using Yao-based protocols in the standard two-party setting (or the OT-hybrid model) is high due to the large number of multiplications needed for computing the matrix-vector product. Our alternative weak PRF candidate (Eq. (1.3)) is more suitable in this setting, and we describe a simple two-party evaluation protocol (in the OT-hybrid model) in Section 6.5. The core ingredient in our protocol is a lightweight *information-theoretic* garbling scheme using arithmetic randomized encoding techniques (cf. [AIK11]). The full two-party distributed evaluation protocol additionally relies on a single (parallel) invocation of a 1-out-of-6 OT; the overall two-party distributed evaluation protocol for this alternative candidate is 3 rounds

(rather than the usual 2 rounds with Yao's protocol). The output size of this garbling scheme (as well as the total communication complexity of the distributed evaluation protocol) is linear in the input size *times* the output size of the PRF. Thus, this candidate is particularly attractive when the PRF output is short.

Towards strong pseudorandomness. Turning now to strong pseudorandomness, we first describe in Section 5.3 a simple non-adaptive distinguishing attack against our basic PRF candidate by reducing it to a sparse polynomial interpolation problem over $\mathbb{Z}_3$ (which can be solved using existing algorithms for sparse polynomial interpolation [Zip90]). Next, in Appendix B, we show that we can also express our PRF as an automaton with multiplicity. We can then apply known learning results for these function families [BV96] to obtain an adaptive attack against our weak PRF candidate. In fact, we show that the learning algorithms for automaton with multiplicity rule out a broad class of depth-2 strong PRF candidates (which includes a direct generalizations of our basic candidate to the setting of mod-p/mod-q moduli). Neither the non-adaptive sparse interpolation attack against our basic candidate nor the more general adaptive attack based on learning automaton with multiplicity seem to extend to the setting of weak pseudorandomness. Both attacks require seeing the outputs of the PRF on heavily-correlated inputs that are unlikely to arise given uniform samples. Moreover, we can show that if the learning attack in [BV96] can be generalized to the weak pseudorandomness setting (where the learning algorithm is only provided function evaluations on a random subset of the domain), then the same algorithm implies a polynomial-time attack on the learning with rounding (LWR) [BPR12] assumption with any polynomial moduli p and q (Lemma B.7).

Encoded-input PRFs and strong PRFs. Motivated by the fact that many applications of PRFs (e.g., message authentication codes (MACs)) do not naturally follow from weak pseudorandomness, we introduce an intermediate notion between weak PRFs and strong PRFs we refer to as *encoded-input PRFs*. Our new notion suffices for instantiating most applications of strong PRFs, and at the same time, still admits simple constructions (and circumvents known lower bounds on the existence of strong PRFs in various complexity classes). At a high-level, an encoded-input PRF is a function that behaves like a PRF on some (possibly sparse) subset of its domain. Moreover, this subset is specific to the PRF family, and in particular, *independent* of the key. For instance, a suitable subset might be the set of valid codewords in a linear error-correcting code. In Section 7, we formally define this notion, and then show that many standard applications of PRFs (e.g., MACs, authenticated encryption) can be instantiated from encoded-input PRFs by incorporating an additional validity check for the encoded input. The validity check can be made more efficient by using an additional proof provided by the evaluator. We then propose an efficient candidate construction of encoded-input PRFs by combining our weak PRFs with error-correcting codes (Construction 7.9). The resulting construction resists the adaptive attacks we describe in Appendix B and can remain MPC-friendly. Using our candidate encoded-input PRFs, we are able to construct MACs with low-complexity verification and CCA-secure encryption with low-complexity decryption (that is, both operations can be computed by a depth-3 ACC^0 circuit). In fact, as mentioned earlier, for a suitable instantiation of our encoding function (e.g., taking the generator matrix of a *linear* error-correcting code), we obtain a candidate *strong PRF* (Eq. (1.2)) that can be computed by a *depth-3* ACC^0 circuit (Remark 7.14).

1.2 Related Work

There is a large body of work on minimizing different complexity measures of (weak or strong) PRFs. Most relevant to the present work are works proposing PRF constructions that can be evaluated by different classes of low-depth circuits such as AC^0, ACC^0, TC^0 [Kha93, BFKL94, NR99, NR04, NRR00, BPR12, BLMR13, Vio13, ABG+14, BP14, YS16, AR16]. Of these candidates, those in AC^0 [Kha93, AR16] and in ACC^0 [ABG+14, Vio13] are either vulnerable to quasi-polynomial time attacks [Kha93, AR16, ABG+14] or can only be shown to have quasi-polynomial time security [Vio13]. In more detail, the result of Viola [Vio13, Theorem 11] says that assuming hardness of factoring against 2^{n^ε}-time adversaries (for some constant ε), there is a strong PRF in ACC^0 with security against quasi-polynomial time adversaries. We discuss these candidates and their cryptanalysis in greater detail in Section 3.2. On the concrete efficiency side, numerous works have focused on designing simple PRFs that are well-suited for use in specific scenarios such as multiparty computation, homomorphic encryption, or evaluation on embedded systems (see, for example, [Can06, Sha08, ARS+15, MJSC16, CCF+16, AGR+16, DEG+18] and the references therein).

2 Preliminaries

We begin by defining some basic notation that we will use throughout this work. For a positive integer n, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. We use bold uppercase letters (e.g., $\mathbf{A}$, $\mathbf{B}$) to denote matrices.

For a finite set S, we write $x \xleftarrow{R} S$ to denote that x is drawn uniformly at random from S. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote a draw from a distribution $\mathcal{D}$. Unless otherwise noted, we write λ to denote the security parameter. We say that a function $f(\lambda)$ is negligible in λ if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We write $f(\lambda) = \mathrm{poly}(\lambda)$ to denote that f is bounded by some (fixed) polynomial in λ. We say that an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input.

For two sets $\mathcal{X}$ and $\mathcal{Y}$, we write $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ to denote the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. For two functions f and g on a common domain $\mathcal{X}$, we say that f is ε-close to g if $\Pr_x[f(x) \neq g(x)] \leq \varepsilon$ and that it is ε-far from g if $\Pr_x[f(x) \neq g(x)] > \varepsilon$. Next, we review the definition of a pseudorandom function (PRF) [GGM84].

Definition 2.1 (Pseudorandom Function)**.** Let $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles of finite sets indexed by a security parameter λ. Let $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an efficiently-computable collection of functions $\mathsf{F}_\lambda \colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. Then, we say that the function family $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is a (t, ε)-*strong pseudorandom function* if for all adversaries $\mathcal{A}$ running in time $t(\lambda)$, and taking $k \xleftarrow{R} \mathcal{K}_\lambda$ and $f_\lambda \xleftarrow{R} \mathsf{Funs}[\mathcal{X}_\lambda, \mathcal{Y}_\lambda]$, we have that

$$\left|\Pr[\mathcal{A}^{\mathsf{F}_\lambda(k,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{f_\lambda(\cdot)}(1^\lambda) = 1]\right| \leq \varepsilon(\lambda).$$

We say that the function family $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is an (ℓ, t, ε)-*weak pseudorandom function* if for all adversaries $\mathcal{A}$ running in time $t(\lambda)$ and taking $k \xleftarrow{R} \mathcal{K}_\lambda$, $f_\lambda \xleftarrow{R} \mathsf{Funs}[\mathcal{X}_\lambda, \mathcal{Y}_\lambda]$, $x_1, \ldots, x_\ell \xleftarrow{R} \mathcal{X}_\lambda$, we have that

$$\left|\Pr\left[\mathcal{A}\left(1^\lambda, \{(x_i, \mathsf{F}_\lambda(k, x_i))\}_{i\in[\ell]}\right)\right] - \Pr\left[\mathcal{A}\left(1^\lambda, \{(x_i, f_\lambda(x_i))\}_{i\in[\ell]}\right)\right]\right| \leq \varepsilon(\lambda).$$

To simplify the notation, we will often drop the index λ on F. We will also write F_k to denote $\mathsf{F}(k, \cdot)$.

Domains and their representations. The key-space, domain, and range of all of the PRF candidates we consider in this work consist of vector spaces over finite fields (i.e., $\mathbb{Z}_p^k$ for some p and k). For notational convenience, we write everything using vector space notation. However, when measuring the complexity of evaluating the PRF, we measure everything in terms of Boolean operations (as opposed to arithmetic or finite field operations). Specifically, we view the keys, inputs, and outputs of our PRF candidates as vectors of bit-strings, where each bit-string encodes the binary representation of its respective field element. For example, a vector $v \in \mathbb{Z}_p^k$ would be represented by a binary string of length $k \cdot \lceil \log p \rceil$, where each block of $\lceil \log p \rceil$ bits represents a single component of v. This way, we can discuss the *Boolean* circuit complexity of evaluating a PRF over a key-space $\mathbb{Z}_p^{m \times n}$, domain $\mathbb{Z}_p^n$, and range $\mathbb{Z}_q^t$.

Circuit classes. We also recall the definition of several basic complexity classes. First, the circuit class AC^0 consists of all circuits with constant depth, polynomial size, and unbounded fan-in (containing only $\mathsf{AND}, \mathsf{OR}$, and NOT gates). The circuit class TC^0 (resp., TC^1) consists of all circuits with constant (resp., logarithmic) depth, polynomial size, unbounded fan-in and threshold gates.

Definition 2.2 (Modular Gates)**.** For any integer m, the MOD_m gate outputs 1 if m divides the sum of its inputs, and 0 otherwise.

Definition 2.3 (Circuit Class ACC^0)**.** For integers $m_1, \ldots, m_k > 1$, we say that a language $\mathcal{L}$ is in $\mathsf{ACC}^0[m_1, \ldots, m_k]$ if there exists a circuit family $\{C_n\}_{n \in \mathbb{N}}$ with constant depth, polynomial size, and consisting of unbounded fan-in AND, OR, NOT, and $\mathsf{MOD}_{m_1}, \ldots, \mathsf{MOD}_{m_k}$ gates that decides $\mathcal{L}$. We write ACC^0 to denote the class of all languages that is in $\mathsf{ACC}^0[m_1, \ldots, m_k]$ for some $k \geq 0$ and integers $m_1, \ldots, m_k > 0$.

3 Candidate Weak Pseudorandom Functions

In this section, we introduce our candidate weak pseudorandom function families. We begin with a basic candidate below (Construction 3.1), and then describe several generalizations and extensions. When describing our applications in the subsequent sections, we will focus primarily on our basic construction.

Construction 3.1 (Mod-2/Mod-3 Weak PRF Candidate)**.** Let λ be a security parameter, and define parameters $m = m(\lambda)$ and $n = n(\lambda)$. The weak PRF candidate is a function $\mathsf{F}_\lambda : \mathbb{Z}_2^{m \times n} \times \mathbb{Z}_2^n \to \mathbb{Z}_3$ with key-space $\mathcal{K}_\lambda = \mathbb{Z}_2^{m \times n}$, domain $\mathcal{X}_\lambda = \mathbb{Z}_2^n$ and output space $\mathcal{Y}_\lambda = \mathbb{Z}_3$. For a key $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$, we write $\mathsf{F}_{\mathbf{A}}(x)$ to denote the function $\mathsf{F}_\lambda(\mathbf{A}, x)$. We define $\mathsf{F}_{\mathbf{A}}$ as follows:

- On input $x \in \mathbb{Z}_2^n$, compute $y' = \mathbf{A}x \in \mathbb{Z}_2^m$.
- The output is defined by applying a non-linear mapping to y'. In this case, we take our non-linear mapping to be the function $\mathsf{map} : \{0,1\}^m \to \mathbb{Z}_3$ that outputs the sum of the inputs values modulo 3. Specifically, for $y' \in \{0,1\}^m$, we write $\mathsf{map}(y') \coloneqq \sum_{i \in [m]} y'_i \pmod 3$.

We define $\mathsf{F}_{\mathbf{A}}(x) \coloneqq \mathsf{map}(\mathbf{A}x)$. Note that we compute the matrix-vector product $\mathbf{A}x$ over $\mathbb{Z}_2$, and then re-interpret the values as their integer values 0 and 1.

Remark 3.2 (Weak PRF Candidate for Arbitrary p and q)**.** The weak PRF candidate in Construction 3.1 can be generalized to work over two arbitrary fields $\mathbb{Z}_p$ and $\mathbb{Z}_q$ where $p \neq q$. In particular, we define the key-space to be $\mathcal{K}_\lambda = \mathbb{Z}_p^{m \times n}$, the domain to be $\mathcal{X}_\lambda = \mathbb{Z}_p^n$, and the range to be $\mathcal{Y}_\lambda = \mathbb{Z}_q$. We define the non-linear mapping $\mathsf{map}_{p,q} \colon \{0, 1, \ldots, p-1\}^m \to \mathbb{Z}_q$ that computes the sum of input values modulo q:

$$\mathsf{map}_{p,q}(y') \coloneqq \sum_{i \in [m]} y'_i \pmod{q}.$$

Putting all the pieces together, the PRF is defined to be $\mathsf{F}_{\mathbf{A}}(x) \coloneqq \mathsf{map}_{p,q}(\mathbf{A}x)$. In this case, Construction 3.1 corresponds to the special case where $p = 2$ and $q = 3$. Note that for certain choices of p, q, the output of this mapping might not be balanced (this is not the case for $p = 2$ and $q = 3$), and pseudorandomness is then defined with respect to the corresponding distribution. We now describe several variations on our general candidate:

- We can consider a binary input space $\mathcal{X}_\lambda = \mathbb{Z}_2^{n'}$ rather than a mod-p input. In this case, we require that the key $\mathbf{A}$ to be compressing so that the product $\mathbf{A}x$ for a random $x \in \mathbb{Z}_2^{n'}$ is statistically close to the uniform distribution over $\mathbb{Z}_p^m$. For instance, this holds by the leftover hash lemma [HILL99] if we take $n'(\lambda) = \Omega(m \log p)$.

- We can consider more complex input spaces and non-linear mappings. As a concrete example, we can define a PRF where the input domain is an elliptic curve group $E(\mathbb{Z}_q)$ of prime order p. That is, we take the domain to be $\mathcal{X}_\lambda = E(\mathbb{Z}_q)^n$; the key-space and range are unchanged: $\mathcal{K}_\lambda = \mathbb{Z}_p^{m \times n}$ and $\mathcal{Y}_\lambda = \mathbb{Z}_q$. In this case, the linear mapping $\mathbf{A}x$ corresponds to computing a linear combination of elliptic curve points. We can define the non-linear mapping $\mathsf{map}_{p,q}$ from $E(\mathbb{Z}_q)$ into $\mathbb{Z}_q$ to be the mapping that returns the x-coordinate of the curve point (recall that each element in $E(\mathbb{Z}_q)$ can be represented by a pair of (x, y)-coordinates in $\mathbb{Z}_q$).

Remark 3.3 (Multiple Output Bits)**.** The output of our weak PRF candidate from Construction 3.1 consists of a single element in $\mathbb{Z}_3$. In many scenarios (such as the ones we describe in Section 6), we require a PRF with longer output. One way to extend Construction 3.1 to provide longer outputs is to take the vector $\mathbf{A}x \in \mathbb{Z}_2^m$, reinterpret it as a 0/1 vector $y' \in \mathbb{Z}_3^m$, and output $\mathbf{G}y' \in \mathbb{Z}_3^t$, where $\mathbf{G} \in \mathbb{Z}_3^{t \times m}$ is a *fixed* public matrix. Formally, we define the mapping $\mathsf{map}_{\mathbf{G}} \colon \{0,1\}^m \to \mathbb{Z}_3^t$ that maps $y' \mapsto \mathbf{G}y'$, and define the PRF candidate $\mathsf{F} \colon \mathbb{Z}_2^{m \times n} \times \mathbb{Z}_2^n \to \mathbb{Z}_3^t$ to be $\mathsf{F}_{\mathbf{A}}(x) \coloneqq \mathsf{map}_{\mathbf{G}}(\mathbf{A}x)$. Construction 3.1 then corresponds to the special case where $\mathbf{G} = \mathbf{1}^{1 \times m}$, where $\mathbf{1}^{1 \times m}$ denotes the all-ones matrix of dimension 1-by-m. In our constructions, we propose taking $\mathbf{G}$ to be the generator matrix of a linear error-correcting code over $\mathbb{Z}_3$. This choice is motivated by the fact that the generator matrix of a linear code with sufficient distance implements a good extractor for a bit-fixing source [CGH+85]. As a concrete candidate for our constructions, we propose taking $\mathbf{G}$ to be the generator matrix of a BCH code over $\mathbb{Z}_3$. Note that we require $t < m$. Otherwise, if $t \geq m$, then we can use linear algebra (over $\mathbb{Z}_3$) to recover $y' = \mathbf{A}x$ from the output $\mathbf{G}y'$ (since $\mathbf{G}$ is public). Given multiple pairs (x, y'), we can recover the secret key $\mathbf{A}$ (over $\mathbb{Z}_2$). In particular, in our concrete parameter settings, we require $m - t \geq \lambda$.

Remark 3.4 (Using Structured Matrices as the PRF Key)**.** We can improve the asymptotic (and concrete) efficiency of our weak PRF candidate (Construction 3.1) by taking the key to be a *structured* matrix rather than a random matrix. For example, we can take $\mathbf{A}$ to be a uniformly random *Toeplitz* matrix rather than a uniformly random matrix. This has the advantage that the size of the PRF key is reduced from mn to $m+n$. In our concrete parameter proposals (Section 4.5),

both $m, n = O(\lambda)$, so using a Toeplitz matrix reduces the size of the key from being quadratic in the security parameter to being linear in the security parameter. A similar optimization for using a random Toeplitz matrix in place of a random matrix was previously proposed to reduce the key size in authentication schemes based on the learning parity with noise (LPN) problem [GRS08, Pie12].

Similarly, we can also take the key $\mathbf{A}$ to be a generator matrix for a (random) quasi-cylic code (c.f., [Pan15, ABB+17, MBD+18]). Using generator matrices of quasi-cyclic codes enables both short keys (the generator matrix of a quasi-cyclic code is a block-circulant matrix, which can be represented by a single vector of dimension n) as well as more efficient PRF evaluation. Namely, we can use FFT algorithms to efficiently implement the matrix-vector multiplication $\mathbf{A}x$.

3.1 Conjectures on the Security of Weak PRF Candidates

We now state three conjectures on our new family of weak PRF candidates, sorted in order from the weakest to the strongest:

Conjecture 3.5 (General Mod-p/Mod-q Weak PRF Candidate)**.** Let λ be a security parameter. Then, there exist fixed primes p and q and $m, n = \text{poly}(\lambda)$ such that for all $\ell, t = \text{poly}(\lambda)$, there exists a function $\varepsilon = \text{negl}(\lambda)$ such that the family $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ from Remark 3.2 is an (ℓ, t, ε)-weak PRF.

Conjecture 3.6 (Mod-2/Mod-3 Weak PRF Candidate)**.** Let λ be a security parameter. Then, there exist $m, n = \text{poly}(\lambda)$ such that for all $\ell, t = \text{poly}(\lambda)$, there exists $\varepsilon = \text{negl}(\lambda)$ such that the function family $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ from Construction 3.1 is an (ℓ, t, ε)-weak PRF.

Conjecture 3.7 (Exponential Hardness of Mod-2/Mod-3 Weak PRF Candidate)**.** Let λ be a security parameter. Then, there exist constants $c_1, c_2, c_3, c_4 > 0$ such that for $n = c_1\lambda$, $m = c_2\lambda$, $\ell = 2^{c_3\lambda}$, and $t = 2^\lambda$, the function family $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ from Construction 3.1 is an (ℓ, t, ε)-weak PRF for $\varepsilon = 2^{-c_4\lambda}$.

Remark 3.8 (Further Generalizations)**.** As stated, Conjectures 3.6 and 3.7 are specific to the security of our mod-2/mod-3 weak PRF candidate from Construction 3.1. But more generally, we can consider an analogous pair of conjectures for any fixed mod-p/mod-q candidate (where p and q are distinct primes). Going further, we can even conjecture that the analogous claims hold for *all* choices of p and q. In this work however, we focus on the security of the mod-2/mod-3 candidate, since that candidate is most well-suited for our MPC applications.

Remark 3.9 (Polynomial Number of Samples)**.** Conjecture 3.7 says that the distinguishing advantage of any 2^λ-time weak PRF adversary is exponentially small given an exponential number of samples $\ell = 2^{\Omega(\lambda)}$. In many applications of weak PRFs, it suffices to require hardness against an adversary that sees a polynomial number of samples. For these settings, we can formulate the following *weaker* conjecture: there exists constants $c_1, c_2 > 0$ such that for $n = c_1\lambda$, $m = c_2\lambda$, $t = 2^\lambda$, and any $\ell = \text{poly}(\lambda)$, there exists a constant $c_3 > 0$ such that the function family $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ from Construction 3.1 is a (ℓ, t, ε)-weak PRF for $\varepsilon = 2^{-c_3\lambda}$.

Remark 3.10 (Weak PRF in ACC^0)**.** An appealing property of the mod-2/mod-3 PRF candidate from Construction 3.1 is that the PRF can be computed by a depth-2 ACC^0 circuit (in fact, a depth-2 $\mathsf{ACC}^0[2, 3]$ circuit suffices). Specifically, if $\mathbf{A} \in \mathbb{Z}_2^{m\times n}$ is the secret key to the PRF, then the function $\mathsf{F}_\mathbf{A}$ can be computed by a depth-2 circuit where the first layer consists of m MOD_2 gates,

one associated with each row of $\mathbf{A}$ (concretely, each MOD_2 gate takes as input the subset of input bits on which the corresponding row of $\mathbf{A}$ depends). All of the MOD_2 gates feed into two MOD_3 gates, each computing one bit of the binary encoding of the output value (more precisely, the MOD_3 gate computing the most significant bit of the output outputs 1 if the sum of the inputs is 2 mod 3 and the MOD_3 gate computing the least significant bit of the outputs outputs 1 if the sum of its input bits is 1 mod 3). Note that we can also implement the PRF in depth-2 $\mathsf{ACC}^0[6]$, that is, ACC^0 with MOD_6 gates only (using essentially the same construction). In either case, we conclude that under Conjecture 3.6, there exists a weak-PRF candidate in depth-2 ACC^0. Intuitively, this means that under Conjecture 3.6, the complexity class ACC^0 should be hard to learn. We formalize this intuition in Section 5.1.

3.2 Comparison with Other Weak PRF Candidates

In this section, we compare our weak PRF candidate (Construction 3.1) to previous candidates of low-complexity PRFs [BFKL94, NR99, NR04, NRR00, BPR12, BLMR13, Vio13, ABG+14, BP14, AR16]. We conclude by discussing several advantages of our construction.

The Akavia et al. candidate. Akavia et al. [ABG+14] previously introduced a weak PRF candidate in ACC^0 (more precisely, in the class $\mathsf{AC}^0 \circ \mathsf{MOD}_2$) that shares many structural properties with our candidate (Construction 3.1). Specifically, the key is a random matrix $\mathbf{A} \in \mathbb{Z}_2^{n \times n}$ and the PRF is defined to be $\mathsf{F}_{\mathbf{A}}(x) := g(\mathbf{A}x)$, where the function g is a specially-designed non-linear "tribes"[7] function. Their construction is then computable by a depth-3 $\mathsf{ACC}^0[2]$ circuit. Our work follows a very similar design philosophy, except we replace the tribes function with the conceptually simpler operation of computing the sum of the outputs $\mathbf{A}x$ modulo 3. Recently, Bogdanov and Rosen [BR17] showed that the Akavia et al. construction (on n-bit inputs) can be computed by a rational polynomial of degree $O(\log n)$. This gives a *quasi-polynomial* time attack (running in time $n^{O(\log n)}$) on the Akavia et al. candidate.

Candidates based on hard learning problems. Blum et al. [BFKL94] proposed a weak PRF construction based on hard learning problems. Specifically, they propose a distribution over (polynomial-size) DNF formulas (or alternatively, decision trees) and conjecture that such functions are hard to learn given uniform samples. They then give a direct construction of a weak PRF assuming hardness of learning for this distribution. More concretely, the input space of the resulting PRF candidate is $\{0,1\}^n$ and the key consists of two random disjoint sets $A, B \subseteq [n]$. On input $x \in \{0,1\}^n$, the PRF first computes y_A to be the parity of the bits of x indexed by A and y_B to be the majority function over the bits indexed by B. The output of the PRF is $y = y_A \oplus y_B$. We note though that this candidate is not known to be computable in ACC^0, since the majority function on $\Omega(n)$ bits is not known to be in ACC^0.

Candidates based on expander graphs. Applebaum and Raykov [AR16] gave a weak PRF candidate based on a variant of Goldreich's low-locality one-way function (which is in turn based on expander graphs) [Gol00]. Their weak-PRF candidate can be computed by a depth-3 AC^0 circuit. Although AC^0 is a weaker complexity class than ACC^0, the classic learning result of

[7] A tribe function $T_{w,s} \colon \{0,1\}^{ws} \to \{0,1\}$ is a width-w, size-s DNF, defined via $T_{w,s}(x) = \vee_{j=0}^{s-1}(\wedge_{i=1}^{w} x_{wj+i})$.

Linial et al. [LMN89] gives a quasi-polynomial distinguisher against *all* weak PRF candidates in AC^0.

Number-theoretic candidates. Kharitonov [Kha93] gave a weak PRF in AC^0 satisfying quasi-polynomial security assuming the hardness of factoring. The classic PRF constructions by Naor and Reingold [NR99, NR04, NRR00] give strong PRFs in TC^0 from standard number-theoretic assumptions such as the decisional Diffie-Hellman (DDH) problem or factoring. Viola [Vio13] subsequently built upon the Naor-Reingold family of constructions to obtain a strong PRF in $\mathsf{ACC}^0[m]$ (for any *possibly prime* $m \geq 2$) with quasi-polynomial security (assuming sub-exponential hardness of factoring).

Lattice-based candidates. The classic learning parity with noise (LPN) and learning with errors (LWE) [Reg05] assumptions are also natural starting points for building highly-parallelizable PRFs. However, as discussed in [BPR12], the main obstacle to leveraging the traditional *noisy-learning* problems to constructing *deterministic*[8] pseudorandom functions is finding a way to introduce (sufficiently independent) errors terms into the exponentially-many function outputs of the PRF, while keeping the function deterministic and the key-size polynomial. Lattice-based constructions of PRFs have thus relied on a "derandomized" variant of LWE called the learning with rounding (LWR) assumption (and variants thereof) [BPR12, BLMR13, BP14]. The LWR-based constructions can be computed by simple circuits (TC^0 for the ring-LWR-based construction [BPR12] and TC^1 for the standard LWR-based constructions [BPR12, BLMR13, BP14]). Note that all of the lattice-based constructions are in fact *strong* PRFs. Note that the LWR assumption (Definition B.6) also gives a direct construction of a weak PRF (where the secret key is the LWR secret, and the PRF evaluation consists of taking the rounded inner product between the secret key and the input).

Advantages of our construction. We now describe two appealing properties of our new weak PRF candidate compared to the existing ones:

- **Low complexity:** Our weak PRF candidate is the first that can be computed by an ACC^0 circuit and plausibly satisfy exponential security (Conjecture 3.7). Previous PRF candidates in ACC^0 (or AC^0) only provided quasi-polynomial security [Vio13, ABG+14, AR16]. In fact, our candidates are computable by a *depth-2* ACC^0 circuit, which is the *minimal* depth possible for any PRF candidate. To our knowledge, there are no other candidates that can be computed by a depth-2 AC^0 or ACC^0 circuit (even if we just require polynomial hardness).
- **MPC-friendliness:** Another advantage of our construction is that our PRF is very MPC-friendly. Specifically, we consider scenarios where multiple parties hold shares of the PRF key as well as the PRF input, and the goal is for the parties to compute the PRF output on their joint inputs. The structure of our PRF is very amenable for use in MPC protocols. Notably, much of the computation is *linear* (over $\mathbb{Z}_2$ and $\mathbb{Z}_3$). Using (standard) MPC protocols based on linear secret-sharing, computing linear functions on secret-shared values can be done *non-interactively*. Communication is only needed to handle the non-linear transformation from values over $\mathbb{Z}_2$ to values over $\mathbb{Z}_3$. In Section 6, we show that this step can be done very efficiently using either the protocol of Araki et al. [AFL+16] or using oblivious transfers. In

[8] Constructing *randomized* weak PRFs, however, is possible directly from LWE, as shown by Applebaum et al. [ACPS09].

contrast, evaluating the tribes function (in the case of Akavia et al. [ABG+14]) or the majority function (in the case of Blum et al. [BFKL94]) over secret-shared values will incur additional overhead in either round complexity or communication complexity (or both).

4 Rationales for Security

In this section, we provide several rationales to support the conjectured security of our candidate. First, we follow the security analysis of the weak-PRF candidate proposed by Akavia et al. [ABG+14] and show that (1) standard learning algorithms cannot break the security of our construction, and (2) our candidate cannot be expressed as (or even approximated by) a low-degree polynomials over finite fields. In addition, we conjecture that it is difficult to approximate our construction with low-degree rational functions. Finally, we suggest concrete parameters for our candidate weak PRF.

4.1 Lack of Correlation with Fixed Function Families

The most natural way to rule out the existence of pseudorandom functions in a complexity class is to provide a learning algorithm for the class. For instance, Linial, Mansour, and Nisan [LMN89] showed that AC^0-functions can be learned in quasi-polynomial time given access to uniformly random samples. This means that there are no weak PRFs in AC^0. Specifically, Linial et al. showed that *every* AC^0-function is *noticeably* correlated with at least one linear function which depends on at most polylogarithmically many variables. This in turn yields a quasi-polynomial time learning algorithm for AC^0.

For simplicity, we focus on our main mod 2-mod 3 candidate whose output is in $\mathbb{Z}_3$ (identified with $\{0, \pm 1\}$ below) and moreover, we assume $n = m$ (this also corresponds to the parameters we suggest later). We show in this section that with overwhelming probability, a randomly chosen function in our PRF family does not have a noticeable correlation with any sufficiently small (but still exponential-size) collection of functions $\mathcal{H} = \{h \colon \{0,1\}^n \to \{0, \pm 1\}\}$. Our analysis relies on techniques similar to those used by Akavia et al. [ABG+14, Proposition 16].

Lemma 4.1 (No Correlation with Fixed Function Families)**.** *Let $\mathcal{H} = \{h \colon \{0,1\}^n \to \{0, \pm 1\}\}$ be a collection of functions of size s. Then,*

$$\Pr_{\mathbf{A}}\left[\exists h \in \mathcal{H} \mid \Pr_x\left[\mathsf{map}(\mathbf{A}x) = h(x)\right] > \frac{1}{3} + \frac{1}{2^{n-1}} + \varepsilon\right] \leq \frac{5s}{2^n \cdot \varepsilon^2},$$

where $\mathbf{A} \xleftarrow{\mathrm{R}} \{0,1\}^{n \times n}$. In particular, if we take $s = 2^{n/2}$, then with overwhelming probability over the choice of $\mathbf{A}$, there is no function $h \in \mathcal{H}$ that has non-negligible correlation with the function $\mathsf{F}_{\mathbf{A}}(x) = \mathsf{map}(\mathbf{A}x)$. In particular, for any polynomial $p(n)$ and any $s \leq 2^{n/2}$,

$$\Pr_{\mathbf{A}}\left[\exists h \in \mathcal{H} \mid \Pr_x\left[\mathsf{map}(\mathbf{A}x) = h(x)\right] > \frac{1}{3} + \frac{1}{2^{n-1}} + \frac{1}{p(n)}\right] \leq \frac{5p(n)^2}{2^{n/2}} = \mathrm{negl}(n).$$

We give the proof of Lemma 4.1 in Appendix A.1.

4.2 Inapproximability by Low-Degree Polynomials

Another necessary condition for a PRF family is that the family should be hard to approximate by low-degree polynomials. Specifically, assume there exists a degree-d multivariate polynomial

f over GF(2) such that $\mathsf{F}_k(x) = f(x)$ for all $x \in \{0,1\}^n$. Then, given (sufficiently many) PRF evaluations $(x_i, \mathsf{F}_k(x_i))$ on uniformly random values x_i, an adversary can set up a linear system where the unknowns corresponds to the coefficients of f. Since f has degree d, the resulting system has $N = \sum_{k=0}^{d} \binom{n}{k}$ variables. Thus, given $O(2^d \cdot N)$ random samples, the adversary can solve the linear system and recover the coefficients of f (and therefore, a complete description of F_k). We note that this attack still applies even if F_k is $1/O(2^d \cdot N)$-close to a degree d polynomial. In this case, the solution to the system will be $1/O(2^d \cdot N)$-close to F_k with constant probability (which still suffices to break pseudorandomness). Thus, for a candidate PRF family to be secure, the family should not admit a low-degree polynomial approximation.

In our setting, we are able to rule out low-degree polynomial approximations by appealing to the classic Razborov-Smolensky lower bounds for ACC^0 [Raz87, Smo87], which essentially says that for distinct primes p and q, MOD_p gates cannot be computed in $\mathsf{ACC}^0[q^\ell]$ for any $\ell \geq 1$. Translated to our setting, this essentially says that our "modulus-switching" mapping $\mathsf{map}_p \colon \{0,1\}^n \to \mathbb{Z}_p$, which implements the mapping $x \mapsto \sum_{i \in [n]} x_i \pmod p$, is hard to approximate over $\mathrm{GF}(q^\ell)$ as long as $p \neq q$. We formalize this in the following lemma.

Lemma 4.2 (Inapproximability by Low-Degree Polynomials)**.** *For $n > 0$ and $d < n/2$, let $B(n,d) = \frac{1}{2^n} \cdot \sum_{i=0}^{n/2-d-1} \binom{n}{i}$. Then, for all primes $p \neq q$, the function $\mathsf{map}_p \colon \{0,1\}^n \to \mathbb{Z}_q$ on n-bit inputs that maps $x \mapsto \sum_{i \in [n]} x_i \pmod p$ is $B(n,d)$-far from all degree-d polynomials over $\mathrm{GF}(q^\ell)$ for all $\ell \geq 1$.*

We give the proof of Lemma 4.2 in Appendix A.2.

4.3 Inapproximability by Low-Degree Rational Functions

The low-degree polynomial approximation attack described in Section 4.2 generalizes to the setting where the PRF F_k can be approximated (sufficiently well) by a low-degree *rational* function. For instance, suppose there exist multivariate polynomials f, g over GF(2) of degree at most d such that $f(x) = \mathsf{F}_k(x) \cdot g(x)$ for all $x \in \{0,1\}^n$. Then, a similar attack can be mounted, as any random input-output pair corresponds to an equation in the $2N$ variables (with $N = \sum_{k=0}^{d} \binom{n}{k}$) defining polynomials f and g. Thus, if our PRF candidate is $1/O(2^d \cdot N)$-close to a degree-d rational function, then there is an $O(2^d \cdot N)$-time attack given $O(2^d \cdot N)$ evaluations of the PRF.

While the Akavia et al. weak PRF candidate [ABG+14] cannot be approximated by a low-degree polynomial, Bogdanov and Rosen [BR17] showed that the function has rational degree $O(\log n)$ (i.e., the PRF can be written as a rational polynomial of degree $O(\log n)$), where n is the length of the key. This gives a quasi-polynomial distinguisher against the Akavia et al. candidate.

In our case, we conjecture that the map_p function (respectively, the $\mathsf{map}_{p,q}$ function for our more general candidates from Remark 3.2) cannot by approximated (sufficiently well) by a low-degree rational function over $\mathrm{GF}(q^\ell)$, for any $q \neq p$ and $\ell \geq 1$. While the Razborov-Smolensky argument used to argue hardness of approximation of map_p by low-degree polynomials over $\mathrm{GF}(q^\ell)$ does not generalize to rational functions, we still believe that this is a very plausible conjecture.

Conjecture 4.3 (Inapproximability by Rational Functions)**.** For any distinct primes $p \neq q$, any integer $\ell \geq 1$, and any $d = o(n)$, there exists a constant $\alpha < 1$ such that the function $\mathsf{map}_p \colon \{0,1\}^n \to \mathbb{Z}_p$ that maps $x \mapsto \sum_{i \in [n]} x_i \pmod p$ is $1/(2^d \cdot N)^\alpha$-far from all degree-d rational functions over $\mathrm{GF}(q^\ell)$.

We believe that studying this conjecture is a natural and well-motivated complexity problem. Proving or disproving this conjecture would lead to a better understanding of ACC^0.

Finally, we note that while Conjecture 4.3 is essential for the *asymptotic* security of our candidate, the concrete cost of this attack is large enough that the concrete security of our instantiations is unlikely to be affected even if the conjecture turns out to be false. For instance, assuming $n = 512$ and that the degree of the rational approximation over GF(2) is very modest (i.e., 10), then the system of equations would already have over $\sum_{k=0}^{10} \binom{512}{k} \approx 2^{68}$ variables. Solving a linear system over this many variables (naïvely) would already require more than 2^{128} operations.

4.4 Resilience to Standard Cryptanalysis Techniques

In this section, we survey several other relevant cryptanalytic techniques and their impact on the conjectured security of our weak PRF candidate.

Pairwise independence and unbiased outputs. First, we note that our candidate is pairwise independent. This is immediate as for any pair of distinct non-zero inputs $x_1, x_2 \in \mathbb{Z}_2^n$, the values of $\mathbf{A}x_1$ and $\mathbf{A}x_2$ are independent and uniformly random over $\mathbb{Z}_2^m$ (over the randomness of $\mathbf{A}$). Then, appealing to Claim A.1, the joint distribution of $\mathsf{map}(\mathbf{A}x_1)$ and $\mathsf{map}(\mathbf{A}x_2)$ for any $x_1 \neq x_2$ is $(1/2^m)$-close to the uniform distribution over $\mathbb{Z}_3$. Correspondingly, this means that the bias of our weak PRF candidate is negligible. Moreover, by refining the proof of Claim A.1, it is easy to show that if $m \equiv 0 \bmod 6$, restricting the input domain to $\mathbb{Z}_2^n \setminus \{0^n\}$ gives a perfectly uniform distribution (in which case, the weak PRF outputs are totally unbiased). Pairwise-independence is sufficient to argue that basic versions of differential and linear cryptanalysis (in the sense of the definitions proposed in [MV12]) do not apply to our candidate. We note that these linear and differential cryptanalysis are particularly relevant when evaluating the security of our encoded-input PRF (Section 7.3), since there, the adversary can make adaptive queries (over a restricted subset of the domain).

Blum-Kalai-Wasserman attacks. Due to the structural similarities between our candidate and the learning parity with noise (LPN) assumption, the Blum-Kalai-Wasserman (BKW) attack [BKW00] seems particularly relevant. Recall that the BKW algorithm on LPN relies on the following insight: given two LPN samples $(\vec{a}, \vec{a} \cdot \vec{s} + e)$, $(\vec{b}, \vec{b} \cdot \vec{s} + e)$, where $\vec{a}, \vec{b}, \vec{s} \in \mathbb{Z}_2^n$ and $e \in \mathsf{Ber}_\tau$ (here Ber_τ denotes the Bernoulli distribution with some parameter $\tau < 1/2$), the adversary can create a "new" sample by adding the two samples (over $\mathbb{Z}_2$). Doing this with carefully-chosen vectors drawn from a large set of samples, it is possible to obtain LPN samples for the basis vectors (e.g., for vectors of the form $\vec{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$). We can then guess the corresponding bit of the LPN secret s_i by taking a majority vote over the "new" LPN samples with respect to $\vec{e}_i$.

We do not see a way to adapt such attacks to our candidate as it does not seem possible to create "new" samples given a collection of samples. In particular, the mixing of the mod-2 and the mod-3 operations in our basic candidate destroys the linear structure exploited by BKW.

Other classical techniques. Several other classical techniques used in cryptanalysis, such as algebraic or correlation attacks, are closely related to the degree of approximation by polynomials or by rational functions. Thus, we can appeal to our previous analysis and conjectures (Sections 4.1 to 4.3) to argue that our weak PRF candidate plausibly resists those attacks.

Assumption	$\lambda = 80$	$\lambda = 128$
LPN	300	384
Construction 3.1 (Optimistic)	160	256
Construction 3.1 (Conservative)	300	384

Table 4: Proposed parameters (for Construction 3.1, we set $m = n$) and comparison with parameters for LPN.

Further cryptanalysis. To conclude, we emphasize that the analysis we have done is not intended to be exhaustive, and we invite the community to further evaluate the security of our new candidate. We believe though that the initial exploratory study we have conducted provides evidence to support the security of our candidate.

4.5 Concrete Parameters

We now propose some concrete parameters for our candidate. Our proposals (summarized in Table 4) are based on our exploration of possible attacks as well as concrete parameters for LPN with constant noise rate. Specifically, we use the parameters suggested by [EKM17, Table 4] based on the estimated runtime on a machine with 2^{60} bits of memory and assuming a constant noise rate $\tau = 1/4$.[9] We propose optimistic and conservative parameters. Our optimistic choice of parameters ($n = m = 2\lambda$, where λ is the security parameter) suggests better parameters than those for LPN, which is in part justified by the fact that the most efficient attacks against LPN (e.g., BKW) do not seem to apply to our candidate. Our conservative parameters are the same as those suggested for LPN. We further conjecture that choosing a structured key (e.g., a Toeplitz matrix or a block-circulant matrix; see Remark 3.4) does not significantly affect the parameters. Based on our exploratory analysis, we see no need to use larger parameters to instantiate our candidate. We encourage further cryptanalysis to support or disprove the validity of our proposals.

5 Connections to Learning Theory

In this section, we highlight several connections of the hardness of our weak PRF candidates with concrete problems in learning theory.

5.1 Hardness of PAC-Learning for ACC^0

In this section, we show that our conjectures from Section 3.1 imply hardness of PAC-learning for the complexity class ACC^0. We begin by reviewing the definition of PAC learning.

Definition 5.1 (PAC Learnability [Val84])**.** Let $\mathcal{C}$ be a class of Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$. We say that $\mathcal{C}$ is *PAC-learnable* if there exists an algorithm $\mathcal{A}$ such that for every $f \in \mathcal{C}$,

[9] Better algorithms for LPN are possible if we allow for machines with even larger memory, but as noted in [EKM17], a machine with 2^{60} bits of memory is already significantly larger than the largest existing supercomputers today.

every distribution $\mathcal{D}$ over $\mathcal{X}$, every $0 < \varepsilon < 1/2$, $0 < \delta < 1$, if we set $h \leftarrow \mathcal{A}(\varepsilon, \delta, \{(x_i, f(x_i))\}_{i\in[N]})$ for some (sufficiently large) N, then with probability at least $1 - \delta$, the hypothesis h satisfies

$$\Pr_{x\in\mathcal{D}}[h(x) \neq f(x)] \leq \varepsilon.$$

We say that $\mathcal{C}$ is *efficiently PAC-learnable* if $N = \mathrm{poly}(n, 1/\varepsilon, 1/\delta)$ and the running time of $\mathcal{A}$ is $\mathrm{poly}(n, 1/\varepsilon, 1/\delta)$.

Theorem 5.2 (Hardness of Learning for Depth-2 ACC^0)**.** *Under Conjecture 3.6, the complexity class of depth-2 ACC^0 circuits is not efficiently PAC-learnable (even under the uniform distribution).*

Proof. Let λ be a security parameter, and let $m, n = \mathrm{poly}(\lambda)$ be parameters under which Conjecture 3.5 holds. We show that if ACC^0 is efficiently PAC-learnable, then there exists an efficient distinguisher $\mathcal{B}$ for the weak PRF candidate from Construction 3.1 with parameters m, n. At a high-level, this follows from the fact that the weak PRF candidate $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ from Construction 3.1 can be computed by a family of ACC^0 circuits (Remark 3.10), so an efficient PAC-learning algorithm for ACC^0 immediately gives a distinguisher for $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$.

For a matrix $\mathbf{A} \in \mathbb{Z}_2^{m\times n}$, define the function $f_{\mathbf{A}}(x) \coloneqq \mathbb{1}(\mathsf{map}(\mathbf{A}x), 0)$ to be the function that outputs 1 if and only if $\mathsf{map}(\mathbf{A}x) = 0$ and 0 otherwise. Then, define the class $\mathcal{C} = \{f_{\mathbf{A}} : \mathbf{A} \in \mathbb{Z}_2^{m\times n}\}$ of Boolean functions on n-bit inputs. As discussed in Remark 3.10, the function $f_{\mathbf{A}} \colon \{0,1\}^n \to \{0,1\}$ can be computed by a depth-2 ACC circuit. Let $0 < \varepsilon < 1/2$ and $0 < \delta < 1$ be constants such that $(1-\delta)(1-\varepsilon) \geq 3/4$. By assumption, if depth-2 ACC^0 is efficiently PAC-learnable, there exists an algorithm $\mathcal{A}$ such that given $N = \mathrm{poly}(n)$ samples of the form $\{(x_i, f_{\mathbf{A}}(x_i))\}_{i\in[N]}$ where $x_i \xleftarrow{\mathrm{R}} \{0,1\}^n$, $\mathcal{A}$ outputs a hypothesis h such that with probability at least $1 - \delta$, $\Pr_x[h(x) \neq f(x)] \leq \varepsilon$, where the probability is taken over a random choice of $x \in \{0,1\}^n$. Moreover, algorithm $\mathcal{A}$ runs in time $\mathrm{poly}(n)$. We use $\mathcal{A}$ to build a distinguisher $\mathcal{B}$ for $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ with $\ell = N + 1 = \mathrm{poly}(n)$ samples:

- Algorithm $\mathcal{B}$ receives a challenge $(x_1, y_1), \ldots, (x_N, y_N), (x_{N+1}, y_{N+1})$ from the weak PRF challenger. It runs $\mathcal{A}(\varepsilon, \delta, \{(x_i, \mathbb{1}(y_i, 0))\}_{i\in[N]})$ to obtain a hypothesis h.
- Finally, algorithm $\mathcal{B}$ output 1 if $h(x_{N+1}) = 1 - \mathbb{1}(y_{N+1}, 0)$, and 0 otherwise.

To complete the proof, we bound the distinguishing probability of $\mathcal{A}$:

- If $y_i = \mathsf{F}_{\mathbf{A}}(x_i) = \mathsf{map}(\mathbf{A}x)$ (for some matrix $\mathbf{A} \in \mathbb{Z}_2^{m\times n}$), then algorithm $\mathcal{B}$ is providing $\mathcal{A}$ with samples of the form $(x_i, f_{\mathbf{A}}(x_i))$ where $x_i \xleftarrow{\mathrm{R}} \{0,1\}^n$. Since $\mathcal{A}$ is a PAC-learning algorithm for ACC^0 (and correspondingly, for the circuit class $\mathcal{C}$), with probability at least $(1-\delta)(1-\varepsilon) \geq 3/4$, $h(x_{N+1}) = \mathbb{1}(y_{N+1}, 0)$. Thus, in the case, algorithm $\mathcal{B}$ outputs 1 with probability at least $3/4$.
- If all of the y_i's are random over $\mathbb{Z}_3$, then y_{N+1} is independent of h and x_{N+1}. In this case, the probability that $\mathbb{1}(y_{N+1}, 0) = h(x_{N+1})$ is at most $2/3$. In this case, algorithm $\mathcal{B}$ outputs 1 with probability at most $2/3$.

The distinguishing advantage of $\mathcal{B}$ is $(1-\delta)(1-\varepsilon) - 2/3 \geq 3/4 - 2/3$, which is non-negligible. This contradicts Conjecture 3.6, and the claim follows. In the above analysis, we only required a learning algorithm $\mathcal{A}$ that operates given samples from the uniform distribution (as opposed to an arbitrary distribution). Thus, under Conjecture 3.6, depth-2 ACC^0 circuits are not efficiently PAC-learnable even if we only require learnability given uniform samples. □

Theorem 5.3 (Hardness of Learning for ACC^0)**.** *Under Conjecture 3.5, the complexity class* ACC^0 *is not efficiently PAC-learnable (even under the uniform distribution).*

Proof (Sketch). The proof proceeds exactly as the proof of Theorem 5.2, with the exception that the circuit needed to implement PRF evaluation is no longer depth-2 (but still constant depth). For a security parameter λ, let p, q and $m, n = \mathrm{poly}(\lambda)$ be the parameters from Conjecture 3.5 under which $\{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is a weak PRF. For a matrix $\mathbf{A} \in \mathbb{Z}_p^{m\times n}$, define the function $f_{\mathbf{A}} \colon \{0,1\}^{n\cdot\lceil \log p\rceil} \to \{0,1\}$ where $f_{\mathbf{A}}(x)$ first interprets the input $x \in \{0,1\}^{n\cdot\lceil \log p\rceil}$ as the binary representation of the elements of an vector $x' \in \mathbb{Z}_p^n$. Then, it computes and outputs $\mathbb{1}(\mathsf{map}_{p,q}(\mathbf{A}x'), 0)$. That is, $f_{\mathbf{A}}(x) = 0$ if $\mathsf{map}_{p,q}(\mathbf{A}x') = 0$ and 1 otherwise. Since p, q are constants, the function $f_{\mathbf{A}}$ can be computed by an $\mathsf{ACC}^0[p, q]$ circuit (where the MOD_p and MOD_q gates are used to implement the modular arithmetic and the modulus switching). The claim then follows by a similar argument as that used in the proof of Theorem 5.2. □

Remark 5.4 (Hardness of Learning for ACC^0 vs. AC^0)**.** By the same argument as in the proof of Theorem 5.2, if we make the stronger conjecture that our mod-2/mod-3 weak PRF candidate satisfies *exponential* hardness (Conjecture 3.7), then we can rule out all sub-exponential time learning algorithms for depth-2 $\mathsf{ACC}^0[2, 3]$ circuits. In contrast, there are quasi-polynomial time algorithms for learning AC^0 circuits [LMN89]. Thus, under Conjecture 3.7, learning depth-2 $\mathsf{ACC}^0[2, 3]$ is *fundamentally harder* than learning AC^0.

5.2 Hardness of Learning for Width-3 Branching Programs

Next we show that our hardness conjectures on our mod-2/mod-3 weak PRF candidate from Construction 3.1 imply a similar hardness of learning results for width-3 permutation branching programs. We first recall the definition of a branching program.

Definition 5.5 (Branching Program)**.** A branching program B on n-bit inputs of width w and length ℓ consists of a sequence of instructions $\{(j_i, f_i, g_i)\}_{i\in[\ell]}$, a start state $z_0 \in [w]$, and a set of accepting states $A \subseteq [w]$. Each instruction (j_i, f_i, g_i) is specified by an index $j_i \in [n]$ and two functions $f_i, g_i \colon [w] \to [w]$. To evaluate B on an n-bit input $x = x_1 \cdots x_n \in \{0,1\}^n$, compute

$$z_i \leftarrow \begin{cases} f_i(z_{i-1}) & x_{j_i} = 0 \\ g_i(z_{i-1}) & x_{j_i} = 1, \end{cases}$$

for all $i \in [\ell]$. The output of the branching program on x, denoted $B(x)$ is 1 if $z_\ell \in A$, and 0 otherwise. We say that B is a permutation branching program if the functions f_i, g_i are permutations on $[w]$ for all $i \in [\ell]$. We say that a branching program is "read-once" if the branching program reads each bit of the input exactly once (i.e., for each $j \in [n]$, there is exactly one $i \in [\ell]$ where $j_i = j$). We define "read-twice" branching programs analogously.

Theorem 5.6 (Hardness of Learning Width-3 Branching Programs)**.** *Under Conjecture 3.6, the class of width-3 branching programs is not PAC-learnable (even under the uniform distribution).*

Proof. The theorem immediately follows from a result by Barrington [Bar85, Lemma 1] who showed that any "3-2-parity circuit" of size s can be simulated by a permutation branching program with width 3 and length $O(ns)$. Here, a 3-2-parity circuit is a circuit with a single mod-3 output gate whose inputs consist of the outputs of mod-2 gates. This precisely captures the structure of the

circuit that evaluates our weak PRF candidate in Construction 3.1. The claim then follows by the same argument as that used in the proof of Theorem 5.2. □

Remark 5.7 (Difficulties in Learning Width-3 Branching Programs)**.** Ergün et al. [EKR95] previously showed that a learning algorithm for width-3 branching programs would imply an algorithm for learning DNFs (given uniform samples). In the same work, they showed that a learning algorithm for width-3 branching programs would also imply a learning algorithm for a restricted version of the LPN problem. Thus, there is evidence that learning width-3 branching programs is seemingly difficult, thus ruling out another candidate class of attacks against our construction.

Remark 5.8 (Learning Width 3 Branching Programs vs. Learning DNFs)**.** As noted before, the work of Ergün et al. [EKR95] showed that a learning algorithm for width-3 branching programs implies an algorithm for learning DNFs (under the uniform distribution). While classic results in learning theory have shown quasi-polynomial time algorithms for learning DNFs or even general AC^0 circuits [LMN89, Ver90], to our knowledge, there are no such algorithms for learning width-3 branching programs. In fact, if we assume that our mod-2/mod-3 weak PRF candidate (Construction 3.1) satisfies *exponential* hardness (Conjecture 3.7), then by the same argument as that used in the proof of Theorem 5.6, there are no sub-exponential time algorithms for learning width-3 branching programs (without membership queries). This means that under Conjecture 3.7, learning width-3 branching programs is *fundamentally harder* than learning DNFs or even AC^0.

5.3 Hardness of Interpolating Sparse Multilinear Polynomials

To conclude, we highlight a connection between the conjectured security of our weak PRF candidate (Construction 3.6) and the hardness of interpolating sparse multivariate polynomials over $\mathbb{Z}_3$. We begin by showing how the behavior of our PRF candidate corresponds to evaluating a sparse polynomial over $\mathbb{Z}_3$.

Take a matrix $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ and an input $x \in \mathbb{Z}_2^n$. Consider the change of variables $y_i = 1 + x_i \pmod 3$. In particular, we map $0 \mapsto 1$ and $1 \mapsto -1$. For $i \in [n]$, let $\mathbf{A}_i$ denote the i^{th} row of $\mathbf{A}$. Under this transformation, the inner product $\mathbf{A}_i x = \sum_{j \in [n]} \mathbf{A}_{i,j} x_j \in \mathbb{Z}_2$ corresponds to the product $\prod_{j \in [n]} y_j^{\mathbf{A}_{i,j}} \in \mathbb{Z}_3$. Essentially, we are embedding the operations over $\mathbb{Z}_2$ by working within the multiplicative subgroup of order 2 in $\mathbb{Z}_3$. This means that we can write

$$\mathsf{F}_{\mathbf{A}}(x) = \sum_{i \in [m]} \left(\prod_{j \in [n]} y_j^{\mathbf{A}_{i,j}} - 1 \right) \in \mathbb{Z}_3, \tag{5.1}$$

where $y_i = 1 + x_i \pmod 3$ are the transformed variables and all of the operations occur over $\mathbb{Z}_3$. Thus, for every choice of the key $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$, the PRF $\mathsf{F}_{\mathbf{A}}$ implements a sparse *multilinear* polynomial over $\mathbb{Z}_3$ with n variables of degree at most n and containing at most m non-zero monomials.[10] Thus, security of our weak PRF candidate implies that it should be hard to approximate sparse multilinear polynomials over $\mathbb{Z}_3$ given *random* evaluations on inputs drawn uniformly at random from the set $\{\pm 1\}^n$. To formalize this, we define the notion of approximately interpolating a polynomial over a finite field. Our definition is adapted from the definition of PAC-learning.

[10] A multilinear polynomial over n variables of degree up to n can have up to 2^n monomials. Our PRF candidate can be expressed as a multilinear polynomial containing $m = \text{poly}(n)$ monomials. Thus, our PRF candidate is very sparse.

Definition 5.9 (Approximate Polynomial Interpolation)**.** Let $\mathbb{F}$ be a finite field, and fix parameters m, n, and d. Let $\mathcal{F} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be a family of polynomials over n variables of degree at most d and containing at most m non-zero coefficients. We say that $\mathcal{F}$ can be *efficiently approximated given evaluations from a distribution* $\mathcal{D}$ if there exists an algorithm $\mathcal{A}$ such that for every $f \in \mathcal{F}$, and every $0 < \varepsilon < 1/2$, $0 < \delta < 1$, if we set $g \leftarrow \mathcal{A}(\varepsilon, \delta, \{x_i, f(x_i)\}_{i \in [N]})$ for some $N = \mathrm{poly}(n, m, d, 1/\varepsilon, 1/\delta)$, with probability at least $1 - \delta$, the function g is ε-close to f, and moreover, the running time of $\mathcal{A}$ is bounded by $\mathrm{poly}(n, m, d, 1/\varepsilon, 1/\delta)$.

Theorem 5.10 (Hardness of Interpolating Sparse Polynomials over $\mathbb{Z}_3$)**.** *Let λ be a parameter. Under Conjecture 3.5, there exists $m, n = \mathrm{poly}(\lambda)$ such that the class of sparse* multilinear *polynomials over $\mathbb{Z}_3$ on n variables with degree at most n and containing at most m non-zero coefficients cannot be efficiently approximated given evaluations drawn uniformly from $\{\pm 1\}^n$.*

Interpolating sparse polynomials. Numerous works have studied the problem of interpolating sparse polynomials over both finite fields [Wer94, GS09, AGR14] and over fields of characteristic zero [Zip79, BOT88, KY88, Zip90]. However, existing algorithms rely on making structured, and oftentimes, adaptively-chosen queries to the underlying polynomial. The existing algorithms do not generalize to the setting where they only have access to *random* evaluations of the polynomial over a restricted subset of the domain. In fact, our conjectures imply an even stronger requirement: it should be difficult to *test* whether a particular function can be represented by a sparse polynomial. We discuss this in Remark 5.11.

Remark 5.11 (Connection to Property Testing)**.** Theorem 5.10 shows that the conjectured hardness of our weak PRF candidate implies that it is difficult to (approximately) interpolate sparse multilinear polynomials over $\mathbb{Z}_3$ (with appropriate degree and sparsity). We can strengthen this to say that it should be difficult to *test* whether a function $\mathsf{F}\colon \mathbb{Z}_3^n \to \mathbb{Z}_3$ has a sparse multilinear polynomial representation of degree n and at most $m = \mathrm{poly}(n)$ non-zero coefficients given only evaluations of the function on *random points* from the set $\{\pm 1\}^n$. This question falls into the general model of property testing [PRS02, AKK+03, JPRZ04, DLM+07], and the question of testing whether a function has a sparse polynomial representation was explicitly considered in the work of Diakonikolas et al. [DLM+07]. While Diakonikolas et al. gave an efficient algorithm for testing whether a function has a sparse polynomial representation, their algorithm relies on making structured queries to the oracle. We do not know of any efficient algorithm for testing whether a function can be represented as a sparse polynomial given only *random* evaluations (over a restricted subset of the domain). The conjectured hardness of our new PRF candidates suggests that this should be a hard problem.

Non-adaptive attacks. While we are primarily interested in the security of our candidate as a weak PRF, we note that that there is a *non-adaptive* distinguishing attack (in fact, a non-adaptive key-recovery attack) on the mod-2/mod-3 variant of our weak PRF candidate (Construction 3.1). Recall that in a non-adaptive attack, the adversary chooses a set of points on which to evaluate the PRF, but it must commit to all of the points ahead of time (before seeing any PRF evaluations). The attack is essentially Zippel's algorithm [Zip90] for sparse polynomial interpolation. We give a high-level sketch of the idea here. For simplicity of exposition, we will just sketch the distinguishing attack. At a high-level, the attack operates by fixing all but a logarithmic number of variables in the polynomial, and then interpolating the polynomial over the restricted subset (which can be

done via linearization). This is also conceptually very similar to the approach from [KL01] for ruling out strong PRFs in depth-2 TC^0.

Let $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ be the PRF key. To simplify the description, suppose $m = n$ (the attack naturally extends to the setting where $m \neq n$). Then, by Eq. (5.1), $\mathsf{F}_{\mathbf{A}}(\cdot)$ is a degree-n multilinear polynomial over n^2 variables and contains n non-zero monomials. The non-adaptive distinguisher works as follows. First, it fixes an arbitrary ± 1 assignment to all but $\ell = c \log n$ variables in $\mathsf{F}_{\mathbf{A}}$ for some constant $c > 1$. Let $\mathsf{F}_{\mathbf{A}}^{(\ell)}$ denote the restriction of $\mathsf{F}_{\mathbf{A}}$ to the inputs lying in the ℓ-dimensional subspace $S \subset \mathbb{Z}_3^{n^2}$ that is consistent with the chosen assignment. By design, $\mathsf{F}_{\mathbf{A}}^{(\ell)}$ is a degree-ℓ multilinear polynomial over $\mathbb{Z}_3$ over ℓ variables and containing at most n non-zero terms. The non-adaptive distinguisher queries the polynomial $\mathsf{F}_{\mathbf{A}}$ on all $2^\ell = O(n)$ elements in the intersection $S \cap \{-1, 1\}^{n^2}$. It uses the evaluations of $\mathsf{F}_{\mathbf{A}}$ (which coincide with the evaluations of $\mathsf{F}_{\mathbf{A}}^{(\ell)}$) to interpolate the restricted polynomial $\mathsf{F}_{\mathbf{A}}^{(\ell)}$ (via linearization). Now, if the distinguisher obtained outputs from the PRF, then the interpolated polynomial will contain at most $n < \ell = 2^c n$ non-zero monomials. If instead the distinguisher obtained outputs of a random function, then the resulting polynomial will contain more than n non-zero monomials with noticeable probability (for instance, when $c \gg 1$). This yields a distinguisher that succeeds with noticeable probability. Mounting this attack requires having sufficiently-many evaluations of the weak PRF on a low-dimensional subspace. Thus, this attack does not seem to give an attack against the weak PRF, and if there was such an attack, it would have implications for interpolating (and property testing for) sparse multilinear polynomials over $\mathbb{Z}_3$ (Theorem 5.10, Remark 5.11).

6 Applications to Secure Multiparty Computation

An attractive feature of our candidate is that it supports efficient evaluation in a fully distributed setting, where both the PRF key and the PRF input are secret-shared between multiple parties. We highlight one such application of this primitive to distributed searchable symmetric encryption (SSE) in Section 6.6.

6.1 Fully-Distributed Weak PRF Evaluation

In this section, we describe a 3-party protocol with security against one passive corruption for secure evaluation of our weak PRF candidate (Construction 3.1).[11] At the beginning of the protocol, we assume that the servers hold a secret-sharing of both the input x and the PRF key k. At the end of the protocol execution, each server should hold a fresh secret-sharing of the output.

We assume the parties use an additive secret sharing scheme, so additions on secret-shared values are free. For multiplications, we use the protocol from Araki et al. [AFL+16] that allows 3 servers to take secret shares of bits $a, b \in \{0, 1\}$ and compute a share of the product $ab \in \{0, 1\}$ where each server only needs to broadcast a single bit. In other words, using the Araki et al. protocol for evaluating Boolean circuits, computing XOR is free while computing an AND requires 1-bit of communication. The protocol relies on pseudorandom secret sharing (PRSS) [CDI05] and requires a one-time setup of replicated PRF keys. We note that we can achieve information-theoretic security without the need for the (trusted) setup at twice the cost of the basic protocol.

[11] The protocol uses two rounds of interaction between the servers.

We now describe our protocol π_{fde} for distributed evaluation of our mod-2/mod-3 candidate (Construction 3.1). We assume a structured key (e.g., a block-circulant matrix), so the key can be compactly represented by a single vector $k \in \mathbb{Z}_2^n$. This assumption is only needed to simplify the protocol description. Our protocol naturally generalizes to the setting with an unstructured (i.e., fully random) key with no overhead (in either communication or round complexity). To recall, to evaluate our PRF, we first evaluate the matrix-vector product between the key and the input: $k, x \mapsto h \in \mathbb{Z}_2^m$. We then reinterpret h as an m-dimensional vector over $\mathbb{Z}_3$. The output $\mathsf{map}_{\mathbf{G}}(h) \in \mathbb{Z}_3^t$ can then be computed as a linear function $\mathsf{map}_{\mathbf{G}}$ on h. We begin by defining the fully-distribution evaluation functionality that we seek to instantiate.

Definition 6.1 (Fully-Distributed Evaluation Functionality)**.** The ideal fully-distributed PRF evaluation functionality is defined as follows:

- **Inputs:** The servers hold replicated additive shares of the input and the key over $\mathbb{Z}_2$. Concretely, let k_1, k_2, k_3 be vectors in $\mathbb{Z}_2^n$ such that $k_1 \oplus k_2 \oplus k_3 = k$ and similarly x_1, x_2, x_3 vectors in $\mathbb{Z}_2^n$ such that $x_1 \oplus x_2 \oplus x_3 = x$. Server i holds k_j, x_j with $j \neq i$.
- **Outputs:** The first two servers hold random $y_1, y_2 \in \mathbb{Z}_3^t$ such that $y_1 + y_2 = \mathsf{F}_k(x)$.

We write $[h]_p$ to denote an additive sharing of h over $\mathbb{Z}_p$—that is, a tuple of values whose sum is h mod p. Depending on the context, this will sometimes be a triple of shares held by the 3 servers and sometimes a pair of shares held by the first 2 servers. Our protocol uses a sub-protocol $\pi_{2,3}$ that transforms an additive sharing $[h]_2$ (i.e., a mod-2 secret-sharing of h) held by the 3 servers into an additive sharing $[h]_3$ (i.e., a mod-3 secret-sharing of h) held by the first two servers. We define this functionality f_{23} below.

Definition 6.2 (Share Conversion Functionality f_{23})**.** The share-conversion functionality converts a 3-party mod-2 secret sharing of a value $h \in \{0, 1\}$ into a 2-party mod-3 secret sharing of the same value h. Specifically, the functionality's input/output behavior is as follows:

- **Inputs:** Every server $i \in [3]$ has an input $b_i \in \{0, 1\}$. Server 1 has an additional input $c \in \mathbb{Z}_3$.
- **Outputs:** Servers 1 and 3 receive no output. Server 2 receives an output $d \in \mathbb{Z}_3$ such that $c + d = b_1 \oplus b_2 \oplus b_3 \pmod 3$.

It is straightforward to design a Boolean circuit that implements the ideal share-conversion functionality from Definition 6.2. We give the circuit in Figure 6.2 below. The circuit consists of 3 AND gates and 10 XOR gates. To obtain our final share-conversion protocol, we use the PRSS-based protocol by Araki et al. [AFL+16] to evaluate the circuit in Figure 1.

The protocol π_{fde}. We now describe our protocol π_{fde} for fully-distributed evaluation of our mod-2/mod-3 weak PRF candidate. Recall that at the beginning of the protocol, we assume that the three servers have a replicated additive secret-sharing of the input and the key. The protocol π_{fde} then consists of three phases:

- During the first phase, each server S_i computes an additive share $h_i \in \mathbb{Z}_2^m$ of the linear mapping $(k, x) \mapsto h$ defined by the key. This can be done *locally* using the replicated additive shares of the input and the key. This follows from the fact that for any two secret-shared values a, b split into 3 shares (i.e., $a = a_1 + a_2 + a_3$ and $b = b_1 + b_2 + b_3$), we have that

Simple Circuit that Implements f_{23}

- **Input:** $((c_0c_1, b_1), b_2, b_3) \in \{0,1\}^5$, where c_0c_1 is the 2-bit representation of $c \in \mathbb{Z}_3$.
- **Output:** $d_0d_1 \in \{0,1\}^2$, representing $d \in \mathbb{Z}_3$.
- **Computation:**
  $d_0 = c_1 \cdot (1 \oplus b_1 \oplus b_2 \oplus b_3) \oplus c_0 \cdot (b_1 \oplus b_2 \oplus b_3)$
  $d_1 = c_0 \oplus (1 \oplus c_1) \cdot (b_1 \oplus b_2 \oplus b_3)$.

Figure 1: A simple circuit that implements the share-conversion functionality f_{23} (Definition 6.2).

$ab = (a_1 + a_2 + a_3)(b_1 + b_2 + b_3) = \sum_{1 \le i,j \le 3} a_i b_j$. In a replicated secret-sharing scheme, server S_i knows a_j, b_j for $j \neq i$. This means that every term $a_i b_j$ in the sum can be computed by at least 1 of the servers.

- In the second step of the protocol, the three servers evaluate the share-conversion protocol $\pi_{2,3}$ to their secret-shared values. For each component of their additive share, the servers runs the interactive protocol $\pi_{2,3}$ to transform additive shares (held by the 3 servers) modulo 2 into additive shares (held by the first 2 servers) modulo 3. At the end of this phase, servers S_1 and S_2 hold a share $[h]_3$ of the linear mapping.
- In the final step of the protocol, the two parties evaluate $\mathsf{map}_{\mathbf{G}}$ on their share. Since the matrix $\mathbf{G}$ is public, this is a linear operation, and can be done non-interactively. The output is the output of the protocol.

Observe that by construction, only the second step of the protocol is interactive. Moreover, the protocol requires just two rounds of interaction. We give the full protocol in Figure 2 below.

Protocol π_{fde}

1. Each server locally computes its shares $h_i \in \mathbb{Z}_2^m$ of the linear mapping $(k, x) \mapsto h$.
2. Server S_1 chooses $c \xleftarrow{R} \mathbb{Z}_3^m$.
3. Servers S_1, S_2, S_3 runs m parallel instances of the share-conversion protocol $\pi_{2,3}$. On the j^{th} instance, each server provides as input its share of $h_{i,j} \in \{0,1\}$, and server S_1 additionally provides as input its value $c_j \in \mathbb{Z}_3$. At the end of this step, server S_2 obtains a share $c' \in \mathbb{Z}_3^m$ where $c + c' = h$.
4. Servers S_1 and S_2 locally apply $\mathsf{map}_{\mathbf{G}}$ to their shares c and c', respectively. They then output their share in $\mathbb{Z}_3^t$.

Figure 2: Description of our fully distributed PRF evaluation protocol with 3 servers.

6.2 Concrete Efficiency of Distributed PRF Evaluation

In this section, we compare the *concrete* efficiency of secure evaluation of our PRF to alternative constructions. Here, we assume that both the input x and the key k to the PRF are secret-shared across multiple servers. We measure the concrete cost in terms of the round complexity and the communication complexity needed for joint evaluation of the PRF. For all of our estimates, we use a concrete security parameter of $\lambda = 128$.

Our methodology. We compare the concrete cost of evaluating our PRF to that of evaluating AES (a common baseline for MPC applications) as well as custom-designed block ciphers optimized for MPC applications like LowMC [ARS+15] and Rasta [DEG+18]. When computing our performance metrics, we use the following values taken from [ARS+15, Table 2] for the size and depth of the Boolean circuit computing AES and LowMC. We use the values taken from [DEG+18, Table 1] for the size and depth of the Boolean circuit computing Rasta.

- **AES-128:** We assume that the AES-128 block cipher can be computed by a Boolean circuit with depth 40 (ignoring the cost of computing the key schedule), and 5440 AND gates. The output of each PRF invocation is 128 bits.
- **LowMC:** There are many different ways to instantiate the LowMC family of block ciphers. For comparisons, we consider variants along two extremes: one that minimizes the multiplicative depth and one that minimizes the number of AND gates of the Boolean circuit implementing the cipher. At 128 bits of security, this corresponds to the following:
  - **Depth-optimized LowMC:** Using the variant that minimizes the multiplicative depth, the LowMC block cipher can be computed by a Boolean circuit with depth 14 and 2646 AND gates. The output is a block of 256 bits.
  - **Gates-optimized LowMC:** Using the variant that minimizes the number of AND gates, the LowMC block cipher can be computed by a Boolean circuit with depth 252 and 756 AND gates. The output is a block of 128 bits.
- **Rasta:** Similar to LowMC, there are several ways to instantiate the Rasta family of block ciphers. For our comparisons, we again consider two variants: one that minimizes the multiplicative depth and one that minimizes the number of AND gates of the Boolean circuit implementing the cipher. At 128-bits of security, this corresponds to the following:
  - **Depth-optimized Rasta:** Using the variant that minimizes the number of AND gates, the Rasta block cipher can be computed by a Boolean circuit with depth 2 and $\approx 2^{33}$ AND gates. The output is a block of $\approx 2^{33}$ bits.
  - **Gates-optimized Rasta:** Using the variant that minimizes the multiplicative depth, the Rasta block cipher can be computed by a Boolean circuit with depth 6 and 2106 AND gates. The output is a block of 351 bits.

There are mainly two different approaches for secure computation protocols: the secret-sharing approach [GMW87, BOGW88, CCD88] and the garbled circuit approach [Yao86]. We consider both approaches for oblivious PRF evaluation:

- **Secret-sharing approaches:** In the secret-sharing approach [GMW87, BOGW88, CCD88] for oblivious PRF evaluation, we assume that each of the parties has a secret share of the input to the computation. For our estimates, we use the work of Araki et al. [AFL+16] who give a highly-efficient 3-party MPC protocol with security against semi-honest adversaries in the honest-majority setting (i.e., security against a single passive corruption in the 3-party setting). In their protocol, each of the three parties has to communicate a single bit per AND gate in the circuit. Thus, the *total* communication complexity of the Araki et al. [AFL+16] protocol is $3 \cdot n$ bits where n is the number of AND gates in the circuit. The round complexity corresponds to the depth of the circuit. The Araki et al. [AFL+16] is well-suited in low latency networks (due to the potentially large round complexity).

- **Garbled circuit approaches:** The classic garbled circuit protocol [Yao86] gives a general 2-round protocol for secure computation in the presence of semi-honest adversaries. Using state-of-the-art garbled circuit optimizations like free-XOR [KS08], and half-gates [ZRE15], the total communication complexity scales linearly with the number of AND gates in the circuit. Concretely, two ciphertexts are communicated for each AND gate, which at the 128-bit security level, translates to 256 bits of communication per AND gate. Note that a small amount of communication is also needed for the oblivious transfers (OTs), but we will neglect those in our basic comparison (the communication is dominated by the size of the circuit). Compared to the secret-sharing-based protocols, the total communication is substantially higher (over 256 times greater than the Araki et al. protocol). However, the round complexity is optimal, which makes these protocols better suited in high latency networks.

In Table 2, we provide a concrete comparison of the communication complexity and round complexity for oblivious evaluation of our PRF candidate. We compare them to the corresponding costs of using the Araki et al. protocol or an optimized garbled-circuit protocol to evaluate standard block ciphers like AES and MPC-optimized block ciphers like LowMC and Rasta.

6.3 Concrete Efficiency of Distributed Evaluation in the Preprocessing Model

In many MPC settings, we can obtain more efficient protocols by working in the *preprocessing* model (or *correlated randomness* model) [Bea92, BDOZ11, DPSZ12, KOS16], where we assume that prior to the computation, a trusted dealer distributes some input-independent randomness to the computing parties. Concretely, this randomness-generation process can itself be implemented by a separate input-independent MPC protocol. The correlated randomness can significantly reduce the online complexity (both computation and communication) of the protocol execution. The piecewise-linear structure of our weak PRF candidates makes them very amenable for fully distributed evaluation in the preprocessing model. Here, we will focus on the two-party setting.

OT correlations. We first recall the classic preprocessing protocol [Bea95] for implementing 1-out-of-2 string OT (on n-bit messages) from a random OT correlation:

- **Preprocessing:** In the preprocessing step, a trusted dealer chooses two random messages $r_0, r_1 \xleftarrow{R} \{0,1\}^n$ and a random bit $z \xleftarrow{R} \{0,1\}$. It gives (r_0, r_1) to the sender and (z, r_z) to the receiver. The pairs (r_0, r_1) and (z, r_z) is referred to as an OT correlation.
- **OT:** In the online phase of the protocol, the sender has messages $x_0, x_1 \in \{0,1\}^n$ and the receiver has a bit $b \in \{0,1\}$. The receiver sends $b' \leftarrow b \oplus z$ to the sender, and the sender replies with two messages $m_0 \leftarrow x_0 \oplus r_{b'}$ and $m_1 \leftarrow x_1 \oplus r_{b'\oplus 1}$. The receiver computes and outputs $m_b \oplus r_z$.

Thus, given a random OT-correlation, the sender and the receiver can implement a 1-out-of-2 OT in 2 rounds with $2n+1$ bits of communication. The size of the precomputed values is $3n+1$ bits. More generally, we can view the messages as coming from any finite group $\mathbb{G}$, in which case the online communication consists of $2n$ elements in $\mathbb{G}$ and a single bit, and the OT correlation consists of $3n$ elements in $\mathbb{G}$ and a single bit.

Oblivious linear-function evaluation. The oblivious linear-function evaluation (OLE) functionality allows a receiver (who holds an element $x \in \mathbb{F}$ in some finite field $\mathbb{F}$) to learn any affine function $ax + b$ held by a sender (who holds inputs $a, b \in \mathbb{F}$). OLE is a useful building block for secure arithmetic computation [NP99, IPS09, DGN+17, BCGI18] and plays an analogous role as oblivious transfer [GMW87, Kil88, IPS08] for the setting of secure Boolean circuit evaluation.

Just as a random OT correlation can be used to implement oblivious transfer, a random OLE correlation can be used to implement OLE. A random OLE correlation over a finite field $\mathbb{F}$ consists of a random affine function for the sender (specified by two field elements $r_a, r_b \in \mathbb{F}$) and the evaluation of the affine function $r_a r_x + r_b$ at a random point $r_x \in \mathbb{F}$ for the receiver. We recall the protocol for implementing OLE from a random OLE correlation below:

- **Preprocessing:** In the preprocessing step, a trusted dealer chooses $r_a, r_b, r_x \xleftarrow{R} \mathbb{F}$ and computes $z = r_a r_x + r_b \in \mathbb{F}$. It gives (r_a, r_b) to the sender and (r_x, z) to the receiver. The pairs (r_a, r_b) and (r_x, z) is referred to as an OLE correlation.
- **OLE:** In the online phase of the computation, the sender has an input $(a, b) \in \mathbb{F}$ and the receiver has an input $x \in \mathbb{F}$. The receiver sends $m_x \leftarrow x - r_x$ to the sender, and the sender replies with $m_a \leftarrow a - r_a$ and $m_b \leftarrow r_a m_x + b - r_b$. Finally, the receiver outputs $m_a x + m_b + z$.

By construction, at the end of the above protocol, the receiver computes and outputs

$$m_a x + m_b + z = (a - r_a)x + r_a(x - r_x) + b - r_b + r_a r_x + r_b = ax + b,$$

exactly as required. Moreover, the above protocol directly generalizes to computing matrix-vector products over $\mathbb{F}$. Concretely, suppose the receiver holds a vector $\mathbf{x} \in \mathbb{F}^n$ while the sender holds an affine function $f(\mathbf{x}) \coloneqq \mathbf{A}\mathbf{x} + \mathbf{b}$ where $\mathbf{A} \in \mathbb{F}^{m \times n}$ and $\mathbf{b} \in \mathbb{F}^m$. Given a random OLE correlation $(\mathbf{R_A}, \mathbf{r_b})$ and $(\mathbf{r_x}, \mathbf{R_A r_x} + \mathbf{r_b})$, the sender and receiver can obliviously compute the matrix-vector product $\mathbf{A}\mathbf{x} + \mathbf{b}$ while just communicating $(mn + m + n)$ elements in $\mathbb{F}$. Moreover, if the matrix $\mathbf{A}$ is block-circulant (and thus, can be represented by cyclic shifts of a single vector $\mathbf{a} \in \mathbb{F}^n$), the communication of the above protocol is further reduced to just $2n + m$ field elements. The size of the precomputed OLE correlations in this case is $2n + 2m$.

When the inputs $(\mathbf{A}, \mathbf{x}, \mathbf{b})$ are additively shared between the sender and the receiver, the two parties can compute a sharing of the affine function $\mathbf{A}\mathbf{x} + \mathbf{b}$ using two invocations of OLE. Specifically, suppose $\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_1$, $\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1$, and $\mathbf{b} = \mathbf{b}_0 + \mathbf{b}_1$, where the sender holds $\mathbf{A}_0, \mathbf{x}_0, \mathbf{b}_0$ while the receiver holds $\mathbf{A}_1, \mathbf{x}_1, \mathbf{b}_1$. Then, $\mathbf{A}\mathbf{x} + \mathbf{b} = (\mathbf{A}_0 + \mathbf{A}_1)(\mathbf{x}_0 + \mathbf{x}_1) + (\mathbf{b}_0 + \mathbf{b}_1)$. The parties use two invocations of OLE to compute (an additive sharing of) the products $\mathbf{A}_0\mathbf{x}_1$ and $\mathbf{A}_1\mathbf{x}_0$. All other components can be computed locally by one of the two parties.

Distributed evaluation in the preprocessing model. We now describe a two-party protocol for fully distributed evaluation of our mod-2/mod-3 weak PRF candidate in the preprocessing model. The two parties hold an additive sharing of the key $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ and an input $x \in \mathbb{Z}_2^n$, and their goal is to compute an additive sharing of $\mathsf{F}_{\mathbf{A}}(x) = \mathsf{map}(\mathbf{A}x)$. We assume that the sender and the receiver have a sufficient number of OLE and OT correlations.

- The parties use two (parallel) OLE invocations to compute an additive secret-sharing of $\mathbf{A}x \in \mathbb{Z}_2^m$. At the end of this protocol, the two parties have an additive secret sharing of $y' = \mathbf{A}x \in \mathbb{Z}_2^m$. Denote the two shares by $y'_0, y'_1 \in \mathbb{Z}_2^m$. This step requires two rounds of interaction.

- The two parties use m (parallel) invocations of 1-out-of-2 OT (with messages in $\mathbb{Z}_3$) to convert their shares of $y' \in \mathbb{Z}_2^m$ into shares over $\mathbb{Z}_3^m$. Specifically, on the i^{th} invocation, the sender chooses a random $r_{0,i} \xleftarrow{R} \mathbb{Z}_3$ and uses $(r_{0,i} + y'_{0,i}, r_{0,i} + y'_{0,i} + 1) \in \mathbb{Z}_3^2$ as its input to the OT while the receiver uses its share $y'_{1,i}$ as its input. Let $r_{1,i}$ be the receiver's output from the OT. By construction, $r_{0,i}$ and $r_{1,i}$ is an additive secret sharing of y'_i over $\mathbb{Z}_3$. This step requires two rounds of interaction.

- The two parties locally apply the map function to their respective $\mathbb{Z}_3$-shares of y'. Since map is a public linear function over $\mathbb{Z}_3$, this can be done non-interactively. Note that replacing map with a general matrix-vector product to obtain longer outputs (see Remark 3.3) does not require additional communication.

The above protocol requires 4 rounds of communication. We now consider the communication complexity and the preprocessing size:

- **Communication complexity:** When the matrix $\mathbf{A}$ is block-circulant, the OLE computation requires a total of $2(2n + m)$ bits of communication. The m invocations of OT requires communicating a total of $2m$ elements in $\mathbb{Z}_3$ and m bits. Since the OTs are performed in parallel, all of the $\mathbb{Z}_3$ elements can be packed together into a single bit-string of length approximately $\lceil 2m \log_2(3) \rceil$. Thus, the total communication complexity is $\approx 4n + 6.2m$ bits.

- **Preprocessing size:** When the matrix $\mathbf{A}$ is block-circulant, the OLE correlations consist of $2(2n + 2m)$ bits. The OT correlations consists of $3m$ elements in $\mathbb{Z}_3$ and m bits. As before, the $3m$ elements of $\mathbb{Z}_3$ can be packed together (specifically, the $2m$ elements on the sender's side and the m elements on the receiver's side are packed separately into bitstrings). The total size of the correlated randomness is $\approx 4n + 9.8m$ bits.

Comparison with existing PRF candidates. We compare the costs of two-party distributed evaluation of our mod-2/mod-3 PRF candidate with that of using a Yao-based evaluation protocol in conjunction with an existing PRF. In the preprocessing model, we assume that the garbled circuit is included as part of the correlated randomness (along with sufficiently many OT correlations). In the fully-distributed setting, the online communication only consists of the oblivious transfers the receiver uses to obtain the labels corresponding to its share of the PRF key and PRF input. At the 128-bit security level, each OT is over a 128-bit message, so the total communication cost needed for each OT is 257 bits. The size of the correlated randomness is 385 bits for each bit of the PRF key and PRF input (for the OT correlations) together with the size of the garbled circuit computing the PRF (256 bits per AND gate). We refer to Table 3 and Section 1.1 for the full comparisons of our protocol against the Yao-based protocols.

6.4 Oblivious PRF Evaluation in the Preprocessing Model

Our fully-distributed evaluation protocol for evaluating Construction 3.1 from Section 6.3 readily extends to a similar protocol for semi-honest oblivious PRF evaluation. Recall that in an oblivious PRF [FIPR05], one party (the "server") holds the PRF key $\mathbf{A}$ while the other party (the "client") holds an input x. At the end of the oblivious evaluation protocol, the client should learn $\mathsf{F}_{\mathbf{A}}(x)$ while the server should not learn anything. Our protocol for fully-distributed evaluation directly gives a protocol for oblivious evaluation of Construction 3.1:

- In the first step, the client and the server use a single OLE invocation to compute an additive secret-sharing of the product $y' = \mathbf{A}x \in \mathbb{Z}_2^m$. Specifically, the server chooses a random vector $\mathbf{b} \xleftarrow{R} \mathbb{Z}_2^m$. Then, the client and the server use a single OLE invocation to compute $\mathbf{A}x + \mathbf{b}$. At the end of this step, the client knows $\mathbf{A}x + \mathbf{b}$ while the server knows $\mathbf{b}$, which is a secret sharing of $\mathbf{A}x$.

- The client and the server uses m (parallel) OT invocation of 1-out-of-2 OT (with messages in $\mathbb{Z}_3$) to convert their shares of $y' \in \mathbb{Z}_2^m$ into shares over $\mathbb{Z}_3^m$. This step is identical to the corresponding step in the fully distributed evaluation protocol.

- The two parties locally apply the map function to their respective $\mathbb{Z}_3$ shares of y', and the server concludes by sending its share of $\mathsf{map}(y')$ to the client. Note that the server can include this message with the final flow of the OT protocol, so this step does not increase the round complexity.

This yields a 4-round oblivious PRF protocol in the preprocessing model where the total online communication cost is roughly $2n + 5.2m$ bits (with proper packing of $\mathbb{Z}_3$ elements). With our suggested parameters ($m = n = 256$), the total communication is roughly 1800 bits, and the total size of the correlated randomness is $2n + 7.8m$, or roughly 2500 bits.

Comparison against generic approaches. A generic approach for building an oblivious PRF is to combine Yao's garbled circuits with an existing PRF. In this case, the online communication cost consists of an OT (on 128-bit messages) for each bit of the client's PRF input together with a 128-bit label for each of bit of the sender's PRF key. Concretely, this translates to 257 bits for each bit of the PRF input and 128 bits for each bit of the PRF key. The size of the correlated randomness is 385 bits for each bit of the input (for the OT correlations) and the size of the garbled circuit (256 bits per AND gate). Compared to these generic approaches, our protocol for oblivious PRF evaluation is over 25x better in terms of online communication complexity and over 95x better in terms of the size of the correlated randomness. The improvement is more substantial when comparing against the generic approaches for evaluating other PRFs like AES and Rasta. We give the full comparison in Table 5.

Comparison against algebraic approaches. There are also several highly-efficient constructions of oblivious PRFs based on discrete log assumptions [NPR99, NR04, FIPR05, JL09]. The simplest construction that provides semi-honest security in the random oracle model is the construction of Naor et al. [NPR99] that operates over a group $\mathbb{G}$ of prime order p where the decisional Diffie-Hellman (DDH) assumption is conjectured to hold. The PRF key is a random element $k \xleftarrow{R} \mathbb{G}$ and the PRF F_k on inputs $x \in \mathcal{X}$ is given by $\mathsf{F}_k(x) \coloneqq H(x)^k$ where $H \colon \mathcal{X} \to \mathbb{G}$ is a hash function (modeled as a random oracle). This PRF admits a simple oblivious evaluation procedure (without preprocessing) where the total communication consists of 2 group elements. At 128-bits of security, this corresponds to roughly 256 bits per group element for a total communication of 512 bits per oblivious PRF evaluation. Compared to the online communication cost of our protocol, oblivious evaluation of the DDH-based construction is better by a factor of roughly 3.6x.

An advantage of our PRF is that it plausibly provides security against quantum computers; in contrast, group-based oblivious PRFs are insecure against quantum adversaries [Sho94]. Another advantage of our oblivious PRF protocol is that computing the matrix-vector product between a

PRF Candidate	Round Complexity	Output Bits	Online Communication	Preprocessing Size
AES	2	128	$4.9 \cdot 10^4$	$1.4 \cdot 10^6$
LowMC, min-gates	2	128	$4.9 \cdot 10^4$	$2.4 \cdot 10^5$
Rasta, min-gates	2	351	$1.4 \cdot 10^5$	$6.7 \cdot 10^5$
DDH-based PRF [NPR99]	2	256	$5.1 \cdot 10^2$	–
Construction 3.1	4	128	$1.8 \cdot 10^3$	$2.5 \cdot 10^3$

Table 5: Comparison of protocols for (semi-honest) oblivious PRF evaluation in the preprocessing model. We measure the online round complexity, the online communication (in bits), and the size of the correlated randomness (in bits) for fully-distributed evaluation of the different PRF candidates. For Construction 3.1, we take $m = n = 256$ (using the parameters from Table 4) and assume that the key $\mathbf{A}$ is a block-circulant matrix.

structured matrix $\mathbf{A}$ and an input x is a relatively lightweight computation compared to evaluating a modular exponentiation over an elliptic-curve group (c.f., [HKL+12, ABB+17, Cho16] for estimates of the computational costs of the different operations). Note that the group-based OPRF cannot take advantage of preprocessing for fixed-based exponentiations (since the base is input-dependent), which is a limiting factor for its efficiency. Thus, in scenarios where computation is the bottleneck (e.g., on fast networks), our OPRF protocol likely outperforms the group-based protocols.

6.5 An Alternative Weak PRF Candidate

While our weak PRF candidate in Construction 3.1 admits a concretely-efficient secure computation protocol when the input and key are secret-shared across three servers, the large number of multiplications makes it less amenable for garbled circuit evaluation (in the two-party setting). Of course, as we demonstrated in Sections 6.3 and 6.4, our main candidate is well-suited for two-party distributed evaluation (and oblivious PRF evaluation) in the preprocessing model. In this section, we introduce an alternative variant of our weak PRF candidate that has a simple distributed evaluation protocol in the two-party setting, and which only relies on OTs (as opposed to the more structured OLE correlations needed for the protocols in Sections 6.3 and 6.4). Our alternative candidate can be viewed either as a deterministic version of LPN, where the "noise" is obtained by taking an inner product over a different modulus, or as a special instance of the learning with rounding (LWR) assumption with constant-size composite modulus (Remark 6.4). We present our candidate below:

Construction 6.3 (Alternative Mod-2/Mod-3 Weak PRF Candidate)**.** Let λ be a security parameter, and let $n = n(\lambda)$ be the key length (and input length). The weak PRF candidate is a function $\mathsf{F}_\lambda \colon \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_2$ with key-space $\mathcal{K}_\lambda = \{0,1\}^n$, domain $\mathcal{X}_\lambda = \{0,1\}^n$ and output space $\mathcal{Y}_\lambda = \mathbb{Z}_2$. For a key $\mathbf{k} \in \mathbb{Z}_2^n$, we write $\mathsf{F}_\mathbf{k}(x)$ to denote the function $\mathsf{F}_\lambda(\mathbf{k}, x)$. We define $\mathsf{F}_\mathbf{k}$ as follows:

- On input $x \in \{0,1\}^n$,

$$\mathsf{F}_{\mathbf{k}}(x) = \sum_{i\in[n]} \mathbf{k}_i x_i \bmod 2 + \sum_{i\in[n]} \mathbf{k}_i x_i \bmod 3 \pmod 2. \tag{6.1}$$

In other words, the PRF evaluation consists of computing the inner product between the key $\mathbf{k}$ and the input x modulo 2 and modulo 3, and then adding the results modulo 2 (where the output of the mod-3 inner product is viewed as a 0/1 element of $\mathbb{Z}_2$). This construction resembles an LPN instance with noise rate 1/3, except instead of sampling the noise independently, the noise is generated via a deterministic, key-dependent, and input-dependent computation. Namely, the noise is 1 if and only if $\langle \mathbf{k}, x\rangle = 1 \bmod 3$.

Remark 6.4 (Alternative View of Construction 6.3)**.** It is easy to see from Eq. 6.1 that $\mathsf{F}_{\mathbf{k}}(x) = 1$ if and only if $\langle \mathbf{k}, x\rangle \bmod 6 \in \{3,4,5\}$. This observation gives another way to evaluate the construction (which we leverage in our secure distributed evaluation protocol). Equivalently, we can express the operation of the PRF as $\mathsf{F}_{\mathbf{k}}(x) = \lfloor \cdot \rceil_2$, where $\lfloor \cdot \rceil_2 \colon \mathbb{Z}_6 \to \mathbb{Z}_2$ is the rounding operator, and we view the key $\mathbf{k}$ and input x as binary vectors over $\mathbb{Z}_6$. Thus, the security of our candidate is closely related to the hardness of the learning with rounding (LWR) assumption [BPR12] with constant-size composite modulus (specifically, using 2 and 6 for the inner and outer modulus, respectively). We note that using a composite modulus in this setting is critical for security in the constant-modulus regime. Otherwise, there is a direct linearization attack (e.g., [AG11]) on the scheme.

Multiple works have studied the hardness of LWR in several parameter settings [BPR12, AKPW13, BGM+16, ASA16]. The known reductions from LWR to worst-case lattice problems do not apply in the constant modulus setting, and we leave the problem of analyzing the security (or insecurity) of LWR with constant-size composite modulus as an interesting research direction.

Remark 6.5 (Insecurity of Variant with $\mathbb{Z}_3$-Outputs)**.** It might seem natural to consider the variant of Construction 6.3 where the outputs are computed mod 3 rather than mod 2 (which would yield longer outputs). However, there is a polynomial-time distinguisher against the mod 3 variant. Indeed, with outputs computed modulo 3, we can obtain 2 as an output only if $\langle \mathbf{k}, x\rangle \neq 0 \bmod 3$ (since the mod-2 inner product adds either 0 or 1 to the output). Moreover, over $\mathbb{Z}_3$, we have $1^2 = 2^2 = 1 \bmod 3$ while $0^2 = 0 \bmod 3$. Hence, if $\mathsf{F}_{\mathbf{k}}(x) = 2 \pmod 3$, then x has to satisfy $(\langle \mathbf{k}, x\rangle)^2 = 1 \pmod 3$, which is a quadratic equation over $\mathbb{Z}_3$ where the unknowns correspond to the bits of $\mathbf{k}$. We can efficiently solve the latter system and recover $\mathbf{k}$ via linearization once we have $O(n^2)$ evaluations.

Security of Construction 6.3. While Construction 6.3 may appear slightly simpler than the weak PRF candidate in Construction 3.1, it shares the structural similarity of mixing mod-2 and mod-3 operations. As such, many of the rationales we discussed in Section 4 for arguing security of Construction 3.1 (e.g., lack of correlation with fixed function families and inapproximability by low-degree functions) also apply to this alternative candidate. However, as we discuss below, there are sub-exponential attacks against this candidate (based on BKW), as well as a simpler non-adaptive attack (that does not rely on sparse polynomial interpolation):

- **BKW attacks:** The structure of Construction 6.3 is very similar to that of the LWE or LPN problem. In particular, we can view $\mathbf{k}$ to be the LWE or LPN secret, x to be the input, and $\sum_{i\in[n]} \mathbf{k}_i x_i \bmod 3 \pmod 2$ to be the error. The main difference here is that the noise

is *correlated* with both the input as well as the secret key, rather than being independently sampled. Nonetheless, it does not seem straightforward to leverage these correlations to construct a distinguisher for the weak PRF. We do note though that BKW-style attacks (discussed in Section 4.4) do apply to this new candidate, and so asymptotically, the new candidate can only provide sub-exponential security at best. Due to the large number of samples needed to run BKW-style attacks in practice, in many applications, we do not need to significantly increase the concrete parameters to achieve security against known attacks.

- **Simpler non-adaptive attack:** As discussed in Section 5.3, there is a non-adaptive distinguishing attack against the weak PRF candidate from Construction 3.1 via sparse polynomial interpolation. In the case of the alternative candidate (Construction 6.3), there is a simpler non-adaptive attack: namely, the distinguisher simply queries the PRF on the elementary basis vectors (i.e., vectors with Hamming weight 1). Since the key for the PRF is a uniformly random binary vector, with probability 1, the value of the PRF on an input x with Hamming weight 1 will be 0. Thus, all the evaluations will be 0 in this case, compared to 1/2 of the evaluations if the adversary received outputs from a random function. This immediately gives a non-adaptive distinguisher against the candidate.

We leave it as an interesting challenge to further study the security of this weak PRF candidate.

Remark 6.6 (Search-to-Decision Reduction)**.** Due to its structural similarity with the LWE and LPN problems, it is straightforward to give a "search-to-decision reduction" for Construction 6.3 (c.f., [BFKL94, Reg05, AIK09] for similar types of reductions). In other words, if there exists a distinguisher against the weak PRF candidate in Construction 6.3, then there is in fact an adversary that can recover the secret PRF key given random inputs/outputs of the PRF.

Remark 6.7 (Branching Program Complexity)**.** As discussed in Section 5.2, our mod-2/mod-3 weak PRF candidate from Construction 3.1 can be efficiently computed by a width-3 branching program. The same is true for the alternative candidate in Construction 6.3, and in fact, it is straightforward to implement Construction 6.3 using a *read-twice* width-3 permutation branching program (by embedding the $\mathbb{Z}_2$ and $\mathbb{Z}_3$ arithmetic into the symmetric group S_3) or by a *read-once* width-6 permutation branching program (immediate from Remark 6.4). Thus, assuming that Construction 6.3 is a weak PRF, we additionally obtain new hardness of learning results for read-once/read-twice small-width branching programs.

Distributed evaluation in the two-party setting. We now describe a secure two-party protocol for distributed evaluation of our alternative PRF candidate (where both the key and the input are secret-shared). As before, we work in the semi-honest model. Our two-party distributed evaluation protocol consists of 3 rounds, where the two parties implement a garbled circuit evaluation protocol (in the style of Yao) in the first 2 rounds as well as a single invocation of a 1-out-of-6 OT protocol in the last 2 rounds. We begin by introducing the information-theoretic garbling scheme that we use (for the first two rounds of the protocol execution). This kind of garbling is a special case of the general notion of randomized encoding of functions [IK00, AIK04] that is suitable for secure two-party computation. It is sometimes referred to as a *decomposable* randomized encoding [IKOS08] or a *projective* garbling scheme [BHR12].

Definition 6.8 (Information-Theoretic Garbling)**.** Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. An information-theoretic projective garbling for f, or *garbling scheme* for short, is an efficiently-computable function $\hat{f}(w; r)$ satisfying the following requirements:

- **Decomposability:** Each output bit of $\hat{f}$ depends on at most one bit of w. Equivalently, for every input length ℓ there exist 2ℓ "key functions" $\{L_i^{(0)}(r), L_i^{(1)}(r)\}_{i\in[\ell]}$ such that for all $w \in \{0,1\}^\ell$ we have $\hat{f}(w; r) = (L_1^{(w_1)}(r), L_2^{(w_2)}(r), \ldots, L_\ell^{(w_\ell)}(r))$.
- **Correctness:** There exists an efficient decoder $\mathcal{D}$ such that for all $w \in \{0,1\}^*$ we have $\Pr_r\left[\mathcal{D}(\hat{f}(w; r)) = f(w)\right] = 1$.
- **Security:** There exists an efficient simulator $\mathcal{S}$ such that for every $\ell \in \mathbb{N}$ and $w \in \{0,1\}^\ell$, the output of $\mathcal{S}(1^\ell, f(w))$ is distributed identically to $\hat{f}(w; r)$.

The key building block in our two-party evaluation protocol is an information-theoretic garbling scheme (in the sense of Definition 6.8) for computing the mapping $f_z \colon \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_6$ where

$$f_z(\mathbf{k}, x) \coloneqq \langle \mathbf{k}, x\rangle + z \in \mathbb{Z}_6. \tag{6.2}$$

This gives an efficient two-party distribution evaluation protocol based on the alternative view of the evaluation procedure described in Remark 6.4. To recall, to compute $\mathsf{F}_{\mathbf{k}}(x)$, it suffices to compute $\langle \mathbf{k}, x\rangle \bmod 6$ and return 1 if $\langle \mathbf{k}, x\rangle \in \{3, 4, 5\}$, and 0 otherwise. Based on this evaluation procedure, we construct a two-party, 3-round protocol as follows:

1. One of the parties (i.e., the "garbler") chooses a blinding factor $z \xleftarrow{R} \mathbb{Z}_6$. The two parties engage in a two-party protocol such that at the end of the protocol, the other party (i.e., the "evaluator") learns the blinded PRF output $\langle \mathbf{k}, x\rangle + z \in \mathbb{Z}_6$. More precisely, they use the information-theoretic garbling scheme for the function f_z (Eq. (6.2)). Specifically, the garbler first garbles the function $(\mathbf{k}, x) \mapsto \langle \mathbf{k}, x\rangle + z \in \mathbb{Z}_6$, and the receiver OTs for the labels corresponding to the bits of its input (i.e., the bits of $\mathbf{k}$ and x). Note that when $\mathbf{k}$ and x are secret-shared between the two parties, the garbler simply permutes the labels for the bits of $\mathbf{k}$ and x, depending on the shares it possesses (i.e., it interchanges the labels for an input bit if its share of the bit is 1). Note that security of our protocol requires that the blinding factor z be hidden from the evaluator (who has the garbled representation of f_z); thus, we require that the garbling scheme additionally satisfies *function-privacy*.
2. The garbler (who knows z) uses a two-round 1-out-of-6 OT protocol with the evaluator (who knows $\langle \mathbf{k}, x\rangle + z \in \mathbb{Z}_6$) to learn the unblinded PRF evaluation $\mathsf{F}_{\mathbf{k}}(x) \in \mathbb{Z}_2$. Specifically, the garbler sends its first round OT message with the garbled circuit; the evaluator sends back the OT response in the final round of the protocol (after it learns the blinded evaluation $\langle \mathbf{k}, x\rangle + z \in \mathbb{Z}_6$). The full protocol requires 3 rounds of communication.

To complete the protocol description, it suffices to construct an information-theoretic garbling scheme for the function f_z from Eq. (6.2). Here, we rely on an information-theoretic garbling scheme for *arithmetic circuits* (c.f. [AIK11]). We summarize the key properties we require in the following theorem:

Theorem 6.9 (Garbling Scheme for Inner Products [AIK11, Lemma 5.1])**.** *Let R be a finite ring, and for a fixed $z \in R$, let $f_z \colon R^n \times R^n \to R$ be the shifted inner product $f_z(x, y) = \langle x, y \rangle + z$. Then, there exists an information-theoretic arithmetic garbling scheme for f_z with the following properties:*

- ***Decomposability:*** *The garbling scheme consists of an efficiently-computable randomized algorithm $\hat{f}_z(x, y; r)$ that on input $(x, y) \in R^n \times R^n$ and randomness r, outputs an encoding $(\hat{x}, \hat{y}) \in R^{2n} \times R^{2n}$. Moreover, each ring element in $\hat{x}$ (resp., $\hat{y}$) depends on exactly one component of x (resp., y). In fact, there are exactly two components in $\hat{x}$ (resp., $\hat{y}$) that depend on each component of x (resp., y).*
- ***Correctness:*** *There exists an efficient decoder $\mathcal{D}$ such that for all $(x, y) \in R^n \times R^n$, $\Pr_r[\mathcal{D}(\hat{f}_z(x, y; r)) = f_z(x, y)] = 1$.*
- ***Security and Function-Privacy:*** *There exists an efficient simulator $\mathcal{S}$ such that for every $n \in \mathbb{N}$, $x, y \in R^n$, and $z \in R$, the output of $\mathcal{S}(1^n, f_z(x, y))$ is identically distributed to $\hat{f}_z(x, y; r)$. Notably, the simulator $\mathcal{S}$ is* not *given z.*

Theorem 6.9 directly gives an information-theoretic garbling scheme for the function f_z from Eq. (6.2). In particular, when the input vectors are binary-valued, the decomposability property in Theorem 6.9 precisely coincides with the decomposability property required in Definition 6.8.

Complexity of two-party protocol for distributed evaluation of Construction 6.3. Using the information-theoretic garbling scheme for shifted inner products together with a 2-round 1-out-of-6 OT protocol, we obtain a 3-round protocol for two-party evaluation of our alternative PRF candidate. By Theorem 6.9, in the fully-distributed setting, a single evaluation of our protocol requires computing $2n$ 1-out-of-2 OTs on messages containing two $\mathbb{Z}_6$ elements and a single evaluation of a 1-out-of-6 OT on a 1-bit message. We note that this procedure suffices for obtaining a single bit of output. To obtain ℓ-bit outputs, we can run ℓ copies of the protocol in parallel. Naïvely, this would amount to performing $2n\ell$ 1-out-of-2 OTs on 6-bit messages (for representing two $\mathbb{Z}_6$ elements) and ℓ invocations of a 1-out-of-6 OT on 1-bit messages. But since the input is shared across all of the invocations, we can reduce the first set of $2n\ell$ OTs to $n\ell$ 1-out-of-2 OTs on 6-bit messages (corresponding to the encodings for the key bits) and n 1-out-of-2 OTs on $\lceil 2\ell \cdot \log_2 6 \rceil$-bit messages (corresponding to a (packed) encoding for the input bits).

Comparison with other schemes. We now briefly compare the complexity of our two-party distributed evaluation protocol for our alternative PRF candidate with that of using a Yao-based approach in conjunction with existing PRF candidates like AES, LowMC, and Rasta. To simplify the comparisons, we measure the costs in the OT-hybrid model, and take the communication cost of each OT to be the sum of the total input lengths for the client and the server. We additionally measure the total number of OTs. From the above analysis, the communication complexity of our two-party distributed evaluation protocol in the OT-hybrid model can be decomposed as follows:

- $n\ell$ 1-out-of-2 OTs on 6-bit messages: this requires $n\ell \cdot (2 \cdot 6 + 1) = 13n\ell$ bits of communication in the OT-hybrid model.
- n 1-out-of-2 OTs on $\lceil 2\ell \cdot \log_2 6 \rceil$-bit messages: this requires $n \cdot (2 \lceil 2\ell \cdot \log_2 6 \rceil + 1)$ bits of communication in the OT-hybrid model.

PRF Candidate	Output Size	Communication (bits)		Number of OTs
		OT	Total	
AES	128	$6.6 \cdot 10^4$	$1.5 \cdot 10^6$	256
LowMC, min-gates	128	$6.6 \cdot 10^4$	$2.6 \cdot 10^5$	256
Rasta, min-gates	351	$1.8 \cdot 10^5$	$7.2 \cdot 10^5$	702
Construction 6.3	20	$1.8 \cdot 10^5$	$1.8 \cdot 10^5$	8084
	40	$3.6 \cdot 10^5$	$3.6 \cdot 10^5$	15784
	128	$1.1 \cdot 10^6$	$1.1 \cdot 10^6$	49664

Table 6: Comparison of two-party distributed evaluation protocols for different PRF candidates. For the comparisons, we work in the OT hybrid model and measure the total communication cost (in bits) and the number of OTs required. In the OT-hybrid model, we take the communication cost of an OT to be the size of the sender's and receiver's inputs. For AES, LowMC, and Rasta, we measure the cost using Yao's 2-round distributed evaluation protocol. We consider the variants of LowMC and Rasta that minimize the number of AND gates. For Construction 6.3, we use the 3-round distributed evaluation protocol described in Section 6.5. We set $n = 384$ to target 128 bits of security (estimated based on LPN parameters with constant noise rate $\tau = 1/3$).

- ℓ 1-out-of-6 OTs on 1-bit messages: this requires $\ell \cdot (6+3) = 9\ell$ bits of communication in the OT-hybrid model.

For the Yao-based protocols for two-party evaluation of AES, LowMC, and Rasta, we additionally measure the cost of communicating a garbled circuit implementing the underlying functionality. Here, we consider the garbling cost with the free-XOR [KS08] and half-gate [ZRE15] optimizations.

For our candidate, we choose $n = 384$. Our estimates are based on the best-known existing attacks on the closely-related LPN problems. Specifically, the recent estimates from [EKM17, Table 7] indicate that the best attack on an LPN instance with noise rate $\tau = 1/3$ and dimension $n = 384$ on a machine with 2^{60} bits of memory will take time between 2^{124} and 2^{153}. We note that smaller parameters are possible if we additionally restrict the number of samples the LPN adversary obtains (and indeed, the best attacks oftentimes requires an extremely large number of samples). However, the precise trade-off between the number of samples and the distinguishing advantage is not simple to characterize, so we take the more conservative approach in our estimates and set $n = 384$. We give the full comparison of our protocol with that obtained by combining Yao's protocol with an existing PRF candidate in Table 6.

Table 6 shows that in the setting where we only require short PRF outputs (e.g., 20-bits of output),[12] then the communication complexity of our protocol in the OT-hybrid model is about 30% smaller than that obtained from using Yao's protocol to evaluate LowMC. If we require longer outputs, then the combination of Yao's protocol with an MPC-friendly block cipher gives a more efficient two-party distributed evaluation protocol.

[12] It is not clear that the performance of block ciphers like AES, LowMC, or Rasta can be improved when we only require a small number of output bits.

6.6 Applications to Searchable Encryption

In this section, we describe how our shared-input shared-output oblivious PRF primitive can be used to build a distributed searchable symmetric encryption (SSE) system.

Distributed SSE. In the classic SSE problem [SWP00, Goh03, CGKO06], a *single* server holds a set of encrypted documents and a client wants to retrieve all of the documents that match its query. Here, we focus on keyword search and we assume that each document is tagged with a set of keywords. A straightforward PRF-based SSE protocol works as follows. First, the client chooses a key k for a PRF F. Then, for each keyword w, the client adds a mapping from $\mathsf{F}(k, w)$ onto an encrypted list of document indices (corresponding to the documents that contain the keyword w). To search, for a keyword w, the client who holds the PRF key computes the value $\mathsf{F}(k, w)$ and sends it to the server, which then looks up the value in the encrypted index and replies with the set of encrypted document indices. Security of the PRF ensures that the index does not reveal the set of keywords to the server. Note that if we work in the random oracle model, then even a weak PRF suffices to argue security. Namely, the tag associated with a keyword is $\mathsf{F}(k, H(w))$, where H here is a hash function (modeled as a random oracle).

A limitation of the basic single-server SSE is that only clients who possess the secret PRF key k is able to search the database. In the *distributed*-SSE setting, a database curator can construct the index (exactly as in the single-server SSE setting), but then secret share the PRF key to multiple different servers. When a client (who may be different from the database curator and who does not know k) wants to perform a query on a keyword w, they would engage in an MPC protocol with the servers such that at the end of the protocol, the client learns $\mathsf{F}(k, H(w))$ (and nothing more) while the servers learn nothing (about the query w). A shared-input, shared-key oblivious PRF directly provides an efficient solution for implementing distributed SSE. To query on a keyword w, an honest client would first secret-share its input $H(w)$ and send one share to each server. The servers would then engage in an oblivious PRF evaluation where each server's input is their individual key-share and the share of the client's query. At the end of the computation, the servers send the shares of the output $\mathsf{F}(k, H(w))$ to the client, who reconstructs the shares and learns the tag $\mathsf{F}(k, H(w))$. Now, the client can simply look up the entries associated with this tag in the database. Thus, our new weak PRF candidate provides an efficient way of realizing distributed SSE assuming the client is semi-honest. (If a client was malicious, it could query the weak PRF on arbitrary inputs of its choosing; to achieve security in this setting, we will need to rely on the encoded-input weak PRF we introduce in Section 7).

7 Encoded-Input Pseudorandom Functions

Motivated by applications in which a *weak* PRF does not suffice, in this section we introduce a new "encoded-input PRF" primitive that allows us to combine the efficiency advantages of weak PRFs with the security advantages of strong PRFs. Our candidate instantiations of this new primitive also imply a depth-3 *strong* PRF variant of our main depth-2 weak PRF candidate.

As we showed in Section 5.3, there is already a non-adaptive attack against our basic weak PRF candidate via sparse polynomial interpolation. In Appendix B, we show an even stronger result: namely, that strong PRFs do not exist in a large class of depth-2 circuits (which in particular also includes the mod-p/mod-q generalizations of our candidate). Our lower bound relies on the classic

learning algorithm for automata with multiplicity by Bergadano and Varricchio [BV96].

There are many scenarios where a weak PRF does not suffice for security. For instance, if we consider the distributed SSE from Section 6.6 and impose the additional requirements of security against *malicious* adversaries, then a weak PRF no longer suffices. To address this limitation, we introduce a new notion we call an *encoded-input pseudorandom function* (EI-PRF) that can be used as a drop-in replacement for strong PRFs in many applications. At a high-level, an EI-PRF is a function that behaves like a PRF on some (possibly sparse) subset of the domain. As a concrete example, a suitable subset might be the set of codewords under a linear error-correcting code (Construction 7.9).

We now formally define the EI-PRF primitive, describe several natural applications, and propose candidate instantiations whose efficiency is comparable to that of our weak PRF candidate. This EI-PRF candidate remains MPC-friendly, and can thus be useful for MPC applications that require a strong PRF.

7.1 Definitions of (P)EI-PRFs

We define two versions of our notion: *encoded-input pseudorandom function* (EI-PRF) and *protected encoded-input pseudorandom function* (PEI-PRF).

Definition 7.1 (Encoded-Input PRF)**.** Let $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{X}' = \{\mathcal{X}'_\lambda\}_{\lambda\in\mathbb{N}}$, and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles of finite sets indexed by a security parameter λ. Let $\{\mathsf{F}'_\lambda\}_{\lambda\in\mathbb{N}} = \{(\mathsf{E}_\lambda, \mathsf{F}_\lambda)\}_{\lambda\in\mathbb{N}}$ be an efficiently-computable collection of functions where $\mathsf{E}_\lambda\colon \mathcal{X}'_\lambda \to \mathcal{X}_\lambda$ is an encoding function and $\mathsf{F}_\lambda\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ is a keyed evaluation function. Then, we say that $\{\mathsf{F}'_\lambda\}_{\lambda\in\mathbb{N}}$ is a (t,ε)-*encoded-input PRF* (EI-PRF) if the function $\mathcal{K}_\lambda \times \mathcal{X}'_\lambda \to \mathcal{Y}_\lambda$ defined via $(k, x') \mapsto \mathsf{F}_\lambda(k, \mathsf{E}_\lambda(x'))$ is a (t,ε)-strong pseudorandom function. Moreover, we say that F'_λ is computable in $\mathcal{C}$ if F_λ is computable in $\mathcal{C}$.

While the definition of an encoded-input PRF may seem equivalent to that of a standard PRF, the important point is that the encoding function is a *keyless* procedure. This means that an honest user can evaluate for itself the encoding algorithm on an input to obtain a valid *encoded input*, and then ask for the PRF value on the encoded input. The holder of the PRF secret key only needs to evaluate F. This is the reason we define the complexity of an EI-PRF to be the complexity of its evaluation function (rather than the composition of its evaluation and encoding functions). Furthermore, we note that even though the overall function $\mathsf{F}(\cdot, \mathsf{E}(\cdot))$ is a strong PRF, the function F itself may live in a complexity class where strong PRFs do not exist.

One of the main reasons we are interested in EI-PRFs is that we can potentially use them as a drop-in replacement for strong PRFs in concrete applications. In many of these scenarios, however, it does not make sense to assume that the evaluator behaves honestly and will only evaluate the F on properly-encoded inputs. This motivates our stronger notion of a *protected encoded-input PRF* (PEI-PRF), which augments an EI-PRF with an additional verification algorithm. The inputs to a PEI-PRF consists of a point x as well as a proof w that x is a proper encoding (with respect to the encoding function of the underlying EI-PRF). The guarantee is that the output of the PEI-PRF are pseudorandom on all properly-encoded inputs, and $\perp$ on improperly-encoded inputs.

Definition 7.2 (Protected EI-PRF)**.** Let $\{\mathsf{F}'_\lambda\}_{\lambda\in\mathbb{N}} = \{(\mathsf{E}_\lambda, \mathsf{V}_\lambda, \mathsf{F}_\lambda)\}_{\lambda\in\mathbb{N}}$ be an efficiently-computable collection of functions where $\mathsf{E}_\lambda\colon \mathcal{X}'_\lambda \to \mathcal{X}_\lambda \times \mathcal{W}_\lambda$ is a protected encoding function whose range is polynomial-time checkable by $\mathsf{V}_\lambda\colon \mathcal{X}_\lambda \times \mathcal{W}_\lambda \to \{0,1\}$. That is, $\mathsf{V}_\lambda(x, w) = 1$ if and only if (x, w) is

a valid encoding. Finally, $\mathsf{F}_\lambda \colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \times \mathcal{W}_\lambda \to \mathcal{Y}_\lambda$ is a keyed evaluation function. Denote by $\perp$ a special element of $\mathcal{Y}_\lambda$. For a function $f \in \mathsf{Funs}[\mathcal{X}_\lambda, \mathcal{Y}_\lambda]$, define $\mathsf{Eval}^f_\lambda \colon \mathcal{X}_\lambda \times \mathcal{W}_\lambda \to \mathcal{Y}_\lambda$ as:

$$\mathsf{Eval}^f_\lambda(x, w) = \begin{cases} f(x) & \text{if } \mathsf{V}_\lambda(x, w) = 1 \\ \perp & \text{otherwise.} \end{cases}$$

Then, we say that $\{\mathsf{F}'_\lambda\}_{\lambda \in \mathbb{N}}$ is a (t, ε)-PEI-PRF if for all adversaries $\mathcal{A}$ running in time $t(\lambda)$, and taking $k \xleftarrow{R} \mathcal{K}_\lambda$ and $f \xleftarrow{R} \mathsf{Funs}[\mathcal{X}_\lambda, \mathcal{Y}_\lambda]$, we have that

$$\left| \Pr[\mathcal{A}^{\mathsf{F}_\lambda(k,\cdot,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{\mathsf{Eval}^f_\lambda(\cdot,\cdot)}(1^\lambda) = 1] \right| \leq \varepsilon(\lambda).$$

We say that F'_λ is *computable in a circuit class* $\mathcal{C}$ if the mapping $(k, x, w) \mapsto \mathsf{F}_\lambda(k, x, w)$ is computable in $\mathcal{C}$. Finally, we say that a PEI-PRF is *systematic* if the witness w has the form $x' \| w'$ such that $\mathsf{V}_\lambda(x, (x' \| w')) = 1$ if and only if $(x, w) = \mathsf{E}_\lambda(x')$.

Remark 7.3 (Relation between EI-PRFs and PEI-PRFs)**.** PEI-PRFs are stronger objects than EI-PRFs, since if $(\mathsf{E}, \mathsf{V}, \mathsf{F})$ is a PEI-PRF, then (E, F) is an EI-PRF.

We first show that that PEI-PRFs can be generically constructed from EI-PRFs.

Lemma 7.4 (PEI-PRFs from EI-PRFs)**.** *Let $\{(\mathsf{E}^*_\lambda, \mathsf{F}^*_\lambda)\}_\lambda$ be an EI-PRF. Then, assuming F_λ and CNF formulas can be computed by depth-d circuits in a class $\mathcal{C}$, there exists a systematic PEI-PRF $\{(\mathsf{E}_\lambda, \mathsf{V}_\lambda, \mathsf{F}_\lambda)\}_\lambda$ computable by a depth-$(d+1)$ circuit.*

Proof. The lemma follows from the fact that we can check the correctness of any Boolean circuit computation using a CNF formula. In particular, we define a variable associated with each wire in the circuit, and construct a constant-size CNF associated with each gate in the circuit (checking that the gate is implemented correctly). The conjunction of all of these gate-by-gate CNFs gives a CNF for the overall circuit. For notational convenience, we drop the λ subscripts in the description below. We now define a systematic PEI-PRF $(\mathsf{E}_\lambda, \mathsf{V}_\lambda, \mathsf{F}_\lambda)$ as follows:

- $\mathsf{E}(x') \to (x, w)$: On input a point $x' \in \mathcal{X}'$, output $(\mathsf{E}^*(x'), w)$, where w is the set of all of the wire values for the Boolean circuit computing $\mathsf{E}^*(x')$. Specifically, we can write $w = x' \| w'$, where x' is the input to E^* and w' contain the internal (and output) wire values of $\mathsf{E}^*(x')$.
- $\mathsf{V}(x, w) \to \{0, 1\}$: On input an encoded input $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$, the verification algorithm interprets $w = x' \| w'$. Then, it invokes the CNF verification procedure (for checking correct computation of E^*) to check that $\mathsf{E}^*(x') = (x, w)$.
- $\mathsf{F}(k, x, w) \to y$: On input the key $k \in \mathcal{K}$, an encoded input $x \in \mathcal{X}$, and a witness $w \in \mathcal{W}$, the evaluation algorithm outputs $y \leftarrow \mathsf{F}^*(k, x)$ if $V(x, w) = 1$, and $\perp$ otherwise. This can be implemented by computing an AND between the output of $\mathsf{V}(x, w)$ and $\mathsf{F}^*(k, x)$.

Since the verification algorithm V can be expressed as a CNF formula, and moreover, both F^* and CNFs can be computed by a circuit of depth $d > 2$, the evaluation algorithm F can be implemented by a circuit of depth $d + 1$. □

7.2 Applications of (P)EI-PRFs

We now describe several interesting applications of strong PRFs that can be based on PEI-PRFs. Certainly, we can instantiate any application of strong PRFs using an EI-PRF, since EI-PRFs are PRFs if we consider the combination of the encoding and the evaluation functions. However, we note here that our notions of EI-PRFs and PEI-PRFs allow us to obtain interesting alternative instantiations of many of the classic applications of PRFs.

Symmetric encryption with low-depth decryption. A symmetric encryption scheme is a tuple $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ such that the key-generation algorithm $\mathsf{G}(1^\lambda)$ outputs a secret key k, the encryption algorithm $\mathsf{E}(k, m)$ outputs an encryption c of a message m, and the decryption algorithm $\mathsf{D}(k, c)$ outputs a string y such that $\mathsf{D}(k, \mathsf{E}(k, m)) = m$ for all k, m. CPA-security of an encryption scheme requires that an efficient adversary given access to an encryption oracle $\mathsf{E}(k, \cdot)$ cannot distinguish an encryption of m_0 from an encryption of m_1 (for any two adversarially-chosen messages m_0 and m_1). The following theorem states how we can obtain a symmetric encryption scheme with low-depth decryption from any EI-PRF.

Theorem 7.5 (Symmetric Encryption with Low-Depth Decryption)**.** *Let $\mathcal{C}$ be a circuit class. Then, if there exists an EI-PRF computable in $\mathcal{C}$, there exists a symmetric encryption scheme with decryption in $\mathcal{C}$ (assuming $\mathcal{C}$ is closed under composition with binary XOR gates).*

Proof. This theorem follows direction from the textbook construction of symmetric encryption from weak PRFs. Let $(\mathsf{E}^*, \mathsf{F}^*)$ be an EI-PRF with key-space $\mathcal{K}$, domain $\mathcal{X}'$, encoded domain $\mathcal{X}$, and range $\{0,1\}$. Then, we define our symmetric encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ over a binary message space $\mathcal{M} = \{0,1\}$ as:

- $\mathsf{G}(1^\lambda) \to k$: On input the security parameter λ, sample and output an EI-PRF key $k \xleftarrow{R} \mathcal{K}$.
- $\mathsf{E}(k, m) \to (x, c)$: On input an EI-PRF key $k \in \mathcal{K}$ and a message $m \in \{0,1\}$, sample $r \xleftarrow{R} \mathcal{X}'$ and output $(\mathsf{E}^*(r), m \oplus \mathsf{F}^*(k, \mathsf{E}^*(r)))$.
- $\mathsf{D}(k, (x, c)) \to m$: On input an EI-PRF key $k \in \mathcal{K}$ and a ciphertext (x, c), output $m = c \oplus \mathsf{F}^*(k, x)$.

Correctness and CPA-security are immediate by definition. □

Remark 7.6 (Low-Complexity Symmetric Encryption)**.** An important property of the construction in Theorem 7.5 is that the decryption function does *not* need to evaluate the encoding function (since the ciphertext contains the encoded nonce $x = \mathsf{E}^*(x')$). Thus, assuming an EI-PRF in AC^0_d (resp., ACC^0_d), we obtain a symmetric encryption scheme where decryption can be implemented by an AC^0_{d+3} (resp., ACC^0_{d+1}) circuit.[13]

MAC with low-depth verification. A MAC is a tuple $(\mathsf{G}, \mathsf{S}, \mathsf{V})$ where the key-generation algorithm $\mathsf{G}(1^\lambda)$ outputs a secret key k, the signing algorithm $\mathsf{S}(k, m)$ outputs a MAC t of the message m, and the verification algorithm $\mathsf{V}(k, (m, t))$ outputs 1 if and only if t is a valid MAC on m. Unforgeability requires that an efficient adversary with access to the signing oracle $\mathsf{S}(k, \cdot)$ cannot produce a MAC (on any message m) that was not previously output by the signing oracle.

[13] Fan-in 2 XOR gates can be computed by a depth-3 AC^0 circuit and by a depth-1 ACC^0 circuit.

MACs are immediate from strong PRFs: let $\mathsf{G}(1^\lambda)$ outputs a PRF key k and define $\mathsf{S}(k,m) = \mathsf{F}(k,m)$. Then $\mathsf{V}(k,(m,t))$ simply checks if $t = \mathsf{F}(k,m)$. Security is immediate as any adversary that can forge a MAC can also predict the output of the PRF on a previously-unqueried input. In fact, for this basic construction, an unpredictable function suffices, so plausibly, we can realize MACs from weaker complexity classes (where strong PRFs may not exist). However, similar to the case for PRFs, the existence of a learning algorithm for a class $\mathcal{C}$ also rules out the existence of an unpredictable function. Here we show that systematic PEI-PRFs are powerful enough for building MAC. In particular, we obtain the following theorem.

Theorem 7.7 (MACs with Low-Depth Verification)**.** *Let $\mathcal{C}$ be a circuit class. Then, if there exists a systematic PEI-PRF in $\mathcal{C}$, there exists a MAC with verification in $\mathcal{C}$ (assuming $\mathcal{C}$ is closed under composition with an equality testing circuit).*

Proof. Let $\mathsf{F}' = (\mathsf{E}, \mathsf{V}', \mathsf{F})$ be a systematic PEI-PRF with key-space $\mathcal{K}$, domain $\mathcal{X}'$, encoded-domain $\mathcal{X}$, witness space $\mathcal{W}$, and range $\mathcal{Y}$. We construct a MAC with message-space $\mathcal{X}'$ as follows:

- $\mathsf{G}(1^\lambda) \to k$: On input the security parameter λ, the key-generation algorithm samples and outputs a PEI-PRF key $k \xleftarrow{R} \mathcal{K}$.
- $\mathsf{S}(k,m) \to (x,w,t)$: On input the signing key $K \in \mathcal{K}$ and a message $m \in \mathcal{X}'$, the signing algorithm computes the encoding $(x,w) = \mathsf{E}(m)$ and the tag $t = \mathsf{F}(k,(x,w))$. Finally, it outputs (x,w,t).
- $\mathsf{V}(k,m,(x,w,t)) \to \{0,1\}$: On input the signing key k, the message m, and a tag (x,w,t), the verification algorithm outputs 1 if $t = \mathsf{F}(k,(x,w))$ and that $w = (m \| w')$ for some bit-string w'. Otherwise, the verification algorithm outputs 0.

Correctness and security immediately follows from the definition of systematic PEI-PRFs, and verification only relies on evaluating F and checking for equality. □

Remark 7.8 (On Non-Systematic PEI-PRFs)**.** Note that we require the PEI-PRF to be systematic, since the verification algorithm needs to check that (x,w) is a valid encoding of the target-message m (as opposed to just a valid encoding of some message) without recomputing the encoding of m (which need not be a low-depth computation). We note that we can obtain a relaxed version of a MAC with low-depth verification from a non-systematic PEI-PRF if we instead define the message space to be the set of valid PEI-PRF encoded inputs (i.e. pairs (x,w) where $\mathsf{V}'(x,w) = 1$).

CCA-secure symmetric encryption with low-depth decryption. Combining our CPA-secure encryption scheme with low-depth decryption (Theorem 7.5) and our MAC with low-depth verification (Theorem 7.7) yields a CCA-secure symmetric encryption scheme (in fact, an authenticated encryption scheme) with low-depth decryption via the classic "encrypt-then-MAC" construction [BN00]. The decryption procedure of the resulting encryption scheme consists of evaluating the decryption operation of the underlying scheme together with the MAC verification. This increases the depth of the overall decryption circuit by 1.

Distributed SSE for malicious clients. One of our primary motivations for introducing our PEI-PRF notion is to realize a malicious-secure variant of our distributed SSE application. Specifically, in a distributed SSE application, a malicious client might issue queries that are invalid (i.e., do not correspond to the output of the random oracle). In particular, a cheating client can learn the outputs of the PRF on adaptively-chosen inputs, thereby compromising security. However, if we used a PEI-PRF, we can ensure the servers abort whenever the client tries to evaluate the PRF on an invalid input. This is done as follows: in addition to sending the servers a secret sharing of its input, it also sends the servers a secret sharing of the proof that its input is properly encoded. The servers can then jointly evaluates the PEI-PRF. This ensures that the client will learn the output of the PRF only if the client provided a valid encoding; otherwise, the client learns nothing.

7.3 Candidate Constructions of (P)EI-PRFs and Strong PRFs

We begin with a heuristic construction of PEI-PRFs from weak-PRFs using random oracles. This construction is primarily of conceptual interest and follows some basic observations on the connection between weak and strong PRFs [NR95]. We then propose a candidate PEI-PRF based on our mod-2/mod-3 weak PRF candidate (Construction 3.1) that remains MPC-friendly.

A heuristic approach to boost weak PRFs to PEI-PRFs. Suppose that F is a weak PRF. Then, the pair $(\mathcal{H}, \mathsf{F})$ is an EI-PRF, where the hash function $\mathcal{H}$ is modeled as a random oracle. Of course, this is a purely heuristic construction, since we cannot represent a random oracle (over a super-polynomial-size domain) by a polynomial-size circuit. However, we could heuristically instantiate the random oracle by a concrete hash function like $\mathsf{SHA\text{-}3}$. Then, the pair $(\mathsf{SHA\text{-}3}, \mathsf{F})$ gives a heuristic construction of an EI-PRF. This means that if F is a weak PRF computable by a depth-d circuit in a class $\mathcal{C}$ satisfying the requirements of Lemma 7.4, we obtain a heuristic construction of a PEI-PRF computable by a depth-$(d+1)$ circuit in $\mathcal{C}$. This in turn gives heuristic constructions of the applications from Section 7.2 (to symmetric encryption and MACs).

(P)EI-PRF from our candidate. Next, we propose a candidate PEI-PRF based on Construction 3.1 whose design is well-suited for our MPC applications. In particular, we need an encoding function such that the outputs of the PRF on all encoded inputs appear pseudorandom. At a high level, both the non-adaptive attack (Section 5.3) and the adaptive attack (Appendix B) on our weak PRF candidate relies on querying inputs that are heavily correlated (e.g., have low Hamming distance). This suggests that using a code with large minimal distance to encode the input x should prevent these attacks. For MPC-friendliness, we would like to use a linear code, as verifying that an input is a valid codeword can be done efficiently (by multiplying by the parity-check matrix for the code).

A natural candidate is to use a linear code $(\mathbf{G}, \mathbf{H})$ over $\mathbb{Z}_2$: the encoding of an input x' is the codeword $\mathbf{G} \cdot x'$. Unfortunately, the same attack still applies since we can always view the PRF evaluation as $\mathbf{A} \cdot (\mathbf{G} \cdot x') = (\mathbf{A} \cdot \mathbf{G}) \cdot x'$ and interpret $(\mathbf{A} \cdot \mathbf{G})$ as the key. To defend against this, we instead use a linear code over $\mathbb{Z}_3$ and define the encoded bitstring x to be the binary representation of the codeword obtained by applying the code to x' (where we interpret $x' \in \{0,1\}^{n'}$ as a vector over $\mathbb{Z}_3$). By mixing mod-2 and mod-3 operations, the encoding procedure becomes non-linear, but verification can still be expressed as a linear function. At the same time, the use of the linear code ensures that (1) encoded inputs are far from each other, (2) verification is MPC-friendly

as the code is linear, and (3) the input of the second mapping *cannot* be expressed as a read-once computation (this property is important for resisting the learning automata with multiplicity attacks from [BV96]).

Construction 7.9 (EI-PRF and PEI-PRF Candidate)**.** Let λ be a security parameter, and take $n = n(\lambda)$, $n' = n'(\lambda)$, and $m = m(\lambda)$. Let $\mathbf{G} \in \mathbb{Z}_3^{n' \times n}$ and $\mathbf{H} \in \mathbb{Z}_3^{n'-n \times n'}$ be the generator and parity-check matrices, respectively, for a linear error-correcting code over $\mathbb{Z}_3$. We use $\mathbf{G}$ and $\mathbf{H}$ to construct a candidate EI-PRF $(\mathsf{E}_\lambda, \mathsf{F}_\lambda)$ and PEI-PRF $(\mathsf{E}_\lambda, \mathsf{V}_\lambda, \mathsf{F}_\lambda)$ with domain $\{0,1\}^n$ and range $\mathbb{Z}_3$ as follows:

- **EI-PRF:** We define the encoding function $\mathsf{E}_\lambda \colon \{0,1\}^n \to \{0,1\}^{2n'}$ for our encoded-input PRF to be the mapping $x' \in \{0,1\}^n \mapsto \mathsf{bin}(\mathbf{G} \cdot x') \in \{0,1\}^{2n'}$, where bin is a "binary decomposition" operator. In words, the encoding function E_λ takes as input an element in $\{0,1\}^n$, interprets it as a binary-valued vector in $\mathbb{Z}_3^n$, maps it to a codeword $w \in \mathbb{Z}_3^{n'}$, and then outputs the binary representation of the elements of w. Namely, each element of w is a $\mathbb{Z}_3$ element which we can encode using two bits. Next, we take $\mathsf{F}_\lambda \colon \mathbb{Z}_2^{m \times 2n'} \times \mathbb{Z}_2^{2n'} \to \mathbb{Z}_3$ to be our weak PRF candidate (Construction 3.1) with input length $2n'$. The encoded-input PRF is the pair $(\mathsf{E}_\lambda, \mathsf{F}_\lambda)$.

- **PEI-PRF:** Our candidate PEI-PRF construction is defined as $(\mathsf{E}_\lambda, \mathsf{V}_\lambda, \mathsf{F}_\lambda)$ where E_λ and F_λ are defined as above (for our EI-PRF candidate). Checking that an input is properly encoded is now possible using the parity-check matrix (the input is the encoded codeword and the witness is empty). In particular, on input an encoded input $x \in \{0,1\}^{2n'}$, the verification algorithm $\mathsf{V}_\lambda \colon \{0,1\}^{2n'} \to \{0,1\}$ outputs 1 if $(\mathbf{H} \cdot \mathbf{B} \cdot x = 0^{n'-n})$ and 0 otherwise, where $\mathbf{B} \in \mathbb{Z}_3^{n' \times 2n'}$ denotes the powers-of-two matrix:

$$\mathbf{B} = (2 \;\; 1) \otimes \mathbf{I}_{n'} = \begin{pmatrix} 2 & 1 & 0 & & & \cdots & 0 \\ 0 & 0 & 2 & 1 & 0 & \cdots & 0 \\ & & & & \ddots & & \\ 0 & & \cdots & & & 0 & 2 \; 1 \end{pmatrix},$$

  where $\mathbf{I}_{n'}$ denotes the $(n' \times n')$ identity matrix. By construction, on input a binary vector $x \in \{0,1\}^{2n'}$, the mapping $\mathbf{B}x$ returns the unique $y \in \mathbb{Z}_3^{n'}$ such $x = \mathsf{bin}(y)$.

  Note that we can obtain a systematic PEI-PRF by assuming that $\mathbf{G}$ has the form $(\mathbf{I}_k | \mathbf{P})^T$ so that it transforms an input x' into a codeword of the form $(x', \mathbf{P} \cdot x')$. This is without loss of generality.

Remark 7.10 (Non-Interactive Verification)**.** The verification algorithm for our PEI-PRF candidate in Construction 7.9 is a *linear* function over $\mathbb{Z}_3$. Thus, there is an efficient distributed protocol where multiple servers, who each hold a linear secret-sharing of the encoded input, can non-interactively compute a share of the verification bit for their shared input.

Remark 7.11 (Fully-Distributed Evaluation)**.** We can easily extend our 3-party protocol for evaluating our weak PRF candidate (Construction 3.1) to obtain a similar protocol for distributed evaluation of our PEI-PRF candidate in Construction 7.9. In this setting, our objective is to defend against a malicious *client*. The servers are still assumed to be semi-honest. The only additional component we require is a way for the servers the check that the input is valid before releasing the shares of the output. One possibility is to have the client provide secret shares of the encoded input

over both $\mathbb{Z}_2$ and $\mathbb{Z}_3$ to the servers. The servers would use the $\mathbb{Z}_2$-secret shares to evaluate the PRF (using the distributed evaluation protocol from Figure 2) and the $\mathbb{Z}_3$-secret shares to verify the encodings (using the approach from Remark 7.10).

Unfortunately, a malicious client can provide inconsistent shares. Instead, we only require the client to submit a $\mathbb{Z}_2$-sharing of the encoded input and have the servers use the share-conversion protocol $\pi_{2,3}$ (Definition 6.2, Figure 1) to convert this sharing into a $\mathbb{Z}_3$-sharing (between 2 out of the 3 servers). The two servers can then non-interactively compute a secret sharing of the verification bit (since the verification algorithm is linear). Naïvely, performing the verification check introduces an additional round in the evaluation protocol since we need the servers to reveal the verification bit before publishing their shares of the output. Fortunately, in the share-conversion protocol $\pi_{2,3}$, one of the two servers that obtains a share of the $\mathbb{Z}_3$-sharing picks its share. This means that this server can output its share of the verification in the first round and its share of the PRF output in the second round. The other server then computes its share of the verification bit after the first round, checks if the verification is successful, and outputs its share of the PRF output *only* if the verification is successful. Otherwise, it does not output anything. This ensures that the client does not learn anything about the PRF value if the input is not properly encoded, thus yielding a 2-round distributed SSE protocol with security against a malicious client.

Instantiating Construction 7.9. Our main conjecture is that instantiating Construction 7.9 with a random linear code yields an (exponentially-secure) encoded-input PRF. We state our main conjecture below, and then describe several variants:

Conjecture 7.12 (Exponential Hardness of Encoded-Input PRF Candidate)**.** Let λ be a security parameter. Then, there exist constants $c_1, c_2, c_3, c_4 > 0$ such that for $n = c_1\lambda$, $n' = c_2\lambda$, $m = c_3\lambda$ and $t = 2^\lambda$, the function family $\{(\mathsf{E}_\lambda, \mathsf{F}_\lambda)\}_{\lambda\in\mathbb{N}}$ from Construction 7.9, when instantiated using a random linear code (i.e., sample a uniformly random generator matrix $\mathbf{G} \xleftarrow{R} \mathbb{Z}_3^{n'\times n}$ and publish $\mathbf{G}$ as part of the description of the EI-PRF), is a (t, ε)-EI-PRF for $\varepsilon = 2^{-c_4\lambda}$. Correspondingly, for this choice of linear code, the function family $\{(\mathsf{E}_\lambda, \mathsf{V}_\lambda, \mathsf{F}_\lambda)\}_{\lambda\in\mathbb{N}}$ is a (t, ε)-PEI-PRF.

We leave the question of further evaluating the conjecture and studying the concrete achievable tradeoffs between the parameters c_1, c_2, c_3, c_4 for further study.

Remark 7.13 (Alternative Instantiations of Construction 7.9)**.** We now describe several alternative instantiations of Construction 7.9 and its associated hardness conjecture (Conjecture 7.12):

- **Using a specific linear code:** Instead of using a random linear code for the encoding function, we can formulate Construction 7.9 (and Conjecture 7.12) for a specific choice (or family) of linear codes. For example, in our application to natural proof barriers (Remark 7.15), we rely on an instantiation using the Druk-Ishai [DI14] family of codes.
- **A simpler encoding function.** Instead of taking the binary decomposition of the mod-3 codeword in the encoding function E_λ, we can define a simpler encoding function $\mathsf{E}_\lambda\colon \{0,1\}^n \to \{0,1\}^{n'}$ to be the mapping $x' \mapsto \mathsf{lsb}(\mathbf{G}\cdot x')$, where lsb returns the *least significant bit* of each mod-3 component of the codeword $\mathbf{G}\cdot x' \in \mathbb{Z}_3^{n'}$. This encoding function has the advantage that it preserves the length of the codeword. A drawback of this encoding function is that verification is no longer a linear function (unlike bin, the lsb function is not even invertible). This makes it less suitable for distributed evaluation (Remarks 7.10 and 7.11).

Remark 7.14 (Candidate Strong PRF in Depth-3 $\mathsf{ACC}^0[2,3]$)**.** Construction 7.9 gives a strong PRF candidate if we consider the composition of the encoding function E with the evaluation function F. The strong PRF candidate starts by applying a public mod-3 linear encoding function to the input, then interprets the result as a vector over $\mathbb{Z}_2$ and applies a *secret* random mod-2 linear mapping, and finally interprets the result as a vector over $\mathbb{Z}_3$ and outputs the sum of its entries. (One could obtain a variant with a longer output by using a suitable compressive mod-3 linear mapping for the final step, as in our weak PRF candidate.) Since the encoding function E computes a *linear* function over $\mathbb{Z}_3$, it can be computed by a depth-1 $\mathsf{ACC}^0[3]$ circuit. As noted in Remark 3.10, the PRF evaluation function F can be computed by a depth-2 $\mathsf{ACC}^0[2,3]$ circuit. Thus, the composition of E and F can be computed by a *depth-3* ACC^0 circuit (note that the binary decomposition in the encoding function is easily handled via fan-in and does not increase the depth of the circuit). More precisely, under Conjecture 7.12, we obtain a *strong PRF* with exponential security in depth-3 $\mathsf{ACC}^0[2,3]$ by instantiating the error-correcting code in Construction 7.9 with a random linear code. One could also consider a seemingly more conservative variant where the mod-3 linear encoding function is secret.

Remark 7.15 (Asymptotically-Optimal Strong PRFs and Natural Proof Barriers)**.** As noted in Remark 7.14, Construction 7.9 gives a strong PRF candidate if we consider the composition of the encoding function E with the evaluation function F. If both E and F can be computed by a circuit of size $O(\lambda)$ (where all parameters are linear in λ, as in Conjecture 7.12), then we obtain a candidate strong PRF with exponential security that can be computed by linear-size circuits. This gives an "asymptotically optimal" PRF that rules out natural proofs of super-linear circuit lower bounds in the sense of Razborov and Rudich [RR94]. We now describe a variant of our EI-PRF from Construction 7.9 that gives the first candidate instantiation of an asymptotically-optimal PRF, and correspondingly, the first natural proof barrier for proving super-linear circuit lower bounds.

Evaluating our EI-PRF candidate from Construction 7.9 consists of three main steps: encoding the input over $\mathbb{Z}_3$, computing the binary decomposition of the encoded vector, and then multiplying the encoded input with the secret key $\mathbf{A}$ over $\mathbb{Z}_2$. If we instantiate the $\mathbb{Z}_3$-encoding with a linear-time encodable code over $\mathbb{Z}_3$ and then replace the key $\mathbf{A}$ with the generator matrix of a linear-time encodable code over $\mathbb{Z}_2$, then the resulting construction can be computed by a linear-size circuit. For instance, we can instantiate the code with the linear-time encodable code family proposed by Druk and Ishai [DI14] (which builds on the hash function from [IKOS08]). This family gives a randomized construction of a linear-time encodable code that has many of the combinatorial properties of a *random* linear code. Thus, we conjecture that sampling the key to be the generator matrix of a Druk-Ishai code does not compromise the security of our candidate. Putting these pieces together, we obtain a plausible candidate of a strong PRF with exponential security and which can be computed by a linear-size circuit. As far as we know, this is the first candidate instantiation of such an asymptotically-optimal strong PRF. Assuming it is indeed exponentially secure, natural proof techniques cannot prove super-linear circuit lower bounds.

Conclusions. We believe that *the conjectures we have made in this section are strong and a healthy dose of skepticism is warranted.* We hope that the applications and implications we point out will motivate further study and constructions of (P)EI-PRFs, as well as additional cryptanalysis of our concrete candidates. We also leave open the question of setting concrete parameters for our new PEI-PRF (Construction 7.9) and strong PRF candidates (Remark 7.14 and Conjecture 7.12).

Acknowledgments

We thank Benny Applebaum, Andrej Bogdanov, Arkadev Chattopadhyay, Itai Dinur, Neeraj Kayal, Sam Kim, Hart Montgomery, Alon Rosen, Adi Shamir, Alexander Sherstov, Nitzan Tur, Emanuele Viola, Gilles Zémor, and the anonymous TCC reviewers for helpful discussions and pointers. D. Boneh and D. J. Wu were supported by NSF, DARPA, a grant from ONR, and the Simons Foundation. Y. Ishai, A. Passelègue, and A. Sahai were supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, NSF-BSF grant 2015782, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. Y. Ishai was additionally supported by ERC grant 742754, ISF grant 1709/14, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

[ABB+17] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, et al. BIKE: Bit flipping key encapsulation. 2017.

[ABG+14] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in $AC^0 \circ MOD_2$. In *ITCS 2014*, pages 251–260, 2014.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[AFL+16] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *ACM CCS*, pages 805–817, 2016.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *ICALP 2011ICALP*, pages 403–415, 2011.

[AGR14] Andrew Arnold, Mark Giesbrecht, and Daniel S Roche. Sparse interpolation over finite fields via low-order roots of unity. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 27–34. ACM, 2014.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT 2016*, pages 191–219, 2016.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS*, pages 166–175, 2004.

[AIK09] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. *Journal of Cryptology*, 22(4):429–469, October 2009.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In *FOCS*, pages 120–129, 2011.

[AKK+03] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing low-degree polynomials over GF(2). In *APPROX-RANDOM*, pages 188–199, 2003.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *CRYPTO*, pages 57–74, 2013.

[App13] Benny Applebaum. Cryptographic hardness of random local functions-survey. In *TCC*, page 599, 2013.

[AR16] Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In *TCC 2016-BTCC*, pages 27–56, 2016.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT*, pages 430–454, 2015.

[ASA16] Jacob Alperin-Sheriff and Daniel Apon. Dimension-preserving reductions from LWE to LWR. Cryptology ePrint Archive, Report 2016/589, 2016. http://eprint.iacr.org/2016/589.

[Bar85] David A. Barrington. Width-3 permutation branching programs. Technical Memorandum TM-293, 1985.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *ACM CCS*, pages 896–912, 2018.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, pages 169–188, 2011.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, pages 420–432, 1992.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In *CRYPTO*, pages 97–109, 1995.

[BFKL94] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO*, pages 278–291, 1994.

[BGM+16] Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In *TCC 2016-A*, pages 209–224, 2016.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM CCS*, pages 784–796, 2012.

[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 *TCC*, pages 699–729, 2018.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *ACM STOC*, pages 435–440, 2000.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In *CRYPTO*, pages 410–428, 2013.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, pages 531–545, 2000.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *ACM STOC*, pages 1–10, 1988.

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation (extended abstract). In *ACM STOC*, pages 301–309, 1988.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, pages 353–370, 2014.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *EUROCRYPT*, pages 719–737, 2012.

[BR17] Andrej Bogdanov and Alon Rosen. Pseudorandom functions: Three decades later. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography.*, pages 79–158. Springer International Publishing, 2017.

[BV96] Francesco Bergadano and Stefano Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM J. Comput.*, 25(6):1268–1280, 1996.

[Can06] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In *ISC 2006*, pages 171–186, 2006.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *ACM STOC*, pages 11–19, 1988.

[CCF+16] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In *FSE*, pages 313–333, 2016.

[CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *TCC*, pages 342–362, 2005.

[CGH+85] Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *FOCS*, pages 396–407, 1985.

[CGKO06] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM CCS*, pages 79–88, 2006.

[Cho16] Tung Chou. Sandy2x: New Curve25519 speed records. In *SAC*, pages 145–160, 2016.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *CCC*, pages 10:1–10:24, 2016.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low ANDdepth and few ANDs per bit. In *CRYPTO*, pages 662–692, 2018.

[DGN+17] Nico Döttling, Satrajit Ghosh, Jesper Buus Nielsen, Tobias Nilges, and Roberto Trifiletti. TinyOLE: Efficient actively secure two-party computation from oblivious linear function evaluation. In *ACM CCS*, pages 2263–2276, 2017.

[DI14] Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the Gilbert-Varshamov bound and their cryptographic applications. In *ITCS*, pages 169–182, 2014.

[DLM+07] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. In *FOCS*, pages 549–558, 2007.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.

[EKM17] Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In *CRYPTO*, pages 486–514, 2017.

[EKR95] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. On learning bounded-width branching programs. In *COLT*, pages 361–368, 1995.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *CRYPTO*, pages 276–288, 1984.

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

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *ACM STOC*, pages 218–229, 1987.

[Goh03] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. http://eprint.iacr.org/2003/216.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000. http://eprint.iacr.org/2000/063.

[GRS08] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. HB$^\sharp$: Increasing the security and efficiency of HB$^+$. In *EUROCRYPT*, pages 361–378, 2008.

[GS09] Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theoretical Computer Science*, 410(27-29):2659–2662, 2009.

[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 *FSE*, pages 346–365, 2012.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.

[IKM$^+$13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC*, pages 600–620, 2013.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *ACM STOC*, pages 433–442, 2008.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In *TCC*, pages 577–594, 2009.

[JPRZ04] Charanjit S. Jutla, Anindya C. Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. In *FOCS*, pages 423–432, 2004.

[Kha93] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *ACM STOC*, pages 372–381, 1993.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *ACM STOC*, pages 20–31, 1988.

[KL01] Matthias Krause and Stefan Lucks. Pseudorandom functions in TC^0 and cryptographic limitations to proving lower bounds. *Computational Complexity*, 10(4):297–313, 2001.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In *ACM CCS*, pages 830–842, 2016.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP 2008*, pages 486–498, 2008.

[KY88] Erich Kaltofen and Lakshman Yagati. Improved sparse multivariate polynomial interpolation algorithms. In *International Symposium on Symbolic and Algebraic Computation*, pages 467–474. Springer, 1988.

[LMN89] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. In *FOCS*, pages 574–579, 1989.

[MBD+18] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *IEEE Trans. Information Theory*, 64(5):3927–3943, 2018.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In *EUROCRYPT*, pages 311–343, 2016.

[MV12] Eric Miles and Emanuele Viola. Substitution-permutation networks, pseudorandom functions, and natural proofs. In *CRYPTO*, pages 68–85, 2012.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *ACM STOC*, pages 245–254, 1999.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In *EUROCRYPT*, pages 327–346, 1999.

[NR95] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *FOCS*, pages 170–181, 1995.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM*, 51(2):231–262, 2004.

[NRR00] Moni Naor, Omer Reingold, and Alon Rosen. Pseudo-random functions and factoring (extended abstract). In *ACM STOC*, pages 11–20, 2000.

[Pan15] Pavel Panteleev. Fast systematic encoding of quasi-cyclic codes using the chinese remainder theorem. In *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*, pages 1916–1920, 2015.

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In *SOFSEM*, pages 99–114, 2012.

[PRS02] Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic boolean formulae. *SIAM Journal on Discrete Mathematics*, 16(1):20–46, 2002.

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded-depth networks over a complete basis with logical addition (russian). *Matematicheskie Zametki*, 41(4):598–607, 1987. English translation in Mathematical Notes of the Academy of Sci. of the USSR, 41(4):333-338, 1987.

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

[RR94] Alexander A. Razborov and Steven Rudich. Natural proofs. In *ACM STOC*, pages 204–213, 1994.

[Sha08] Adi Shamir. SQUASH - a new MAC with provable security properties for highly constrained devices such as RFID tags. In *FSE*, pages 144–157, 2008.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, pages 124–134, 1994.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *ACM STOC*, pages 77–82, 1987.

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[Val84] Leslie G. Valiant. A theory of the learnable. In *ACM STOC*, pages 436–445, 1984.

[Ver90] Karsten A. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *COLT*, pages 314–326, 1990.

[Vio13] Emanuele Viola. The communication complexity of addition. In *SODA*, pages 632–651, 2013.

[Wer94] Kai Werther. The complexity of sparse polynomial interpolation over finite fields. *Applicable Algebra in Engineering, Communication and Computing*, 5(2):91–103, 1994.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

[YS16] Yu Yu and John P. Steinberger. Pseudorandom functions in almost constant depth from low-noise LPN. In *EUROCRYPT*, pages 154–183, 2016.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, pages 216–226, 1979.

[Zip90] Richard Zippel. Interpolating polynomials from their values. *J. Symb. Comput.*, 9(3):375–403, 1990.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *EUROCRYPT*, pages 220–250, 2015.

A Proofs from Section 4

In this section, we give the proofs from Section 4.

A.1 Proof of Lemma 4.1

We first prove the claim for the setting where the $\mathcal{H}$ contains a single function. The claim then follows by a union bound. Consider a function $h\colon \{0,1\}^n \to \{0,\pm 1\}$ and define the random variable

$$Z(\mathbf{A}) = \Pr_x\left[\mathsf{map}(\mathbf{A}x) = h(x)\right] = \mathbb{E}_x[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x))],$$

where $\mathbb{1}(x,y)$ is the indicator function: $\mathbb{1}(x,y) = 1$ if and only if $x = y$ and 0 otherwise. Then, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{A}}[Z(\mathbf{A})] &= \mathbb{E}_{\mathbf{A}}[\mathbb{E}_x[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x))]] = \mathbb{E}_x[\mathbb{E}_{\mathbf{A}}[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x))]] \\ &= \frac{1}{2^n}\cdot\left(\mathbb{1}(0^n, h(0^n)) + \sum_{x\neq 0^n}\mathbb{E}_{\mathbf{A}}[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x))]\right)\end{aligned}$$

For a uniformly random choice of $\mathbf{A}$ and $x \neq 0^n$, we have that $\mathbf{A}x$ is uniform and independent of x. Hence, for any fixed $x \neq 0^n$, we have:

$$\mathbb{E}_{\mathbf{A}}[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x))] = \mathbb{E}_y[\mathbb{1}(\mathsf{map}(y), h(x))] = \Pr_y\left[\mathsf{map}(y) = h(x)\right].$$

We next need to use the following claim about the distribution of the mapping.

Claim A.1. For any $n > 0$ and $b \in \{0,\pm 1\}$ and taking $y \xleftarrow{\text{R}} \{0,1\}^n$,

$$\left|\Pr[\mathsf{map}(y) = b] - \frac{1}{3}\right| \leq \frac{1}{2^n}.$$

Proof. We show the following: for each n, the number of bitstring mapped to each of $(0,1,-1)$ is either a tuple of the form $(x,x,x+1)$ or $(x+1,x+1,x)$ (up to permutation) for some x. We proceed via induction. The base case $n = 1$ is immediate (take $x = 0$). Assume that for a fixed n that the number of bitstring mapped to each $(0,1,-1)$ has one of these forms. Then, if we introduce an additional bit (either 0 or 1), the number of inputs mapped to $(0,1,-1)$ is then $(2x+1, 2x, 2x+1)$ or $(2x+1, 2x+2, 2x+1)$, which has the desired structure. The claim follows. □

Hence, since $0 \leq \mathbb{1}(0^n, h(0^n)) \leq 1$ we have that $\left|\Pr_y\left[\mathsf{map}(y) = h(x)\right] - \frac{1}{3}\right| \leq \frac{1}{2^n}$ for any fixed x,

$$\mathbb{E}_{\mathbf{A}}[Z(\mathbf{A})] \leq \frac{1}{3} + \frac{1}{2^{n-1}}$$

The next step is to use the Bienaymé-Chebyshev inequality, which we recall below.

Fact A.2 (Bienaymé-Chebyshev Inequality)**.** Let X be a random variable with finite expected value μ and finite non-zero variance σ^2. Then, for any real number $k > 0$:

$$\Pr\left[|X - \mu| \geq k\sigma\right] \leq \frac{1}{k^2}.$$

Applying the inequality immediately gives us:

$$\Pr_{\mathbf{A}}\left[\Pr_x\left[\mathsf{map}(\mathbf{A}x) = h(x)\right] > \frac{1}{3} + \frac{1}{2^{n-1}} + \varepsilon\right] \leq \frac{\sigma^2}{\varepsilon^2},$$

where σ^2 denote the variance of Z, which we estimate to conclude the proof.

$$\begin{aligned}
\mathbb{E}_{\mathbf{A}}[Z(\mathbf{A})^2] &= \mathbb{E}_{\mathbf{A}}[\mathbb{E}_x[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x))]^2] \\
&= \mathbb{E}_{\mathbf{A}}[\mathbb{E}_x[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x))] \cdot \mathbb{E}_y[\mathbb{1}(\mathsf{map}(\mathbf{A}y), h(y))]] \\
&= \mathbb{E}_{x,y}[\mathbb{E}_{\mathbf{A}}[\mathbb{1}(\mathsf{map}(\mathbf{A}x), h(x)) \cdot \mathbb{1}(\mathsf{map}(\mathbf{A}y), h(y))]].
\end{aligned}$$

Again, over the randomness used to sample $\mathbf{A}$, the quantities $\mathbf{A}x$ and $\mathbf{A}y$ are uniformly random and independent over $\{0,1\}^n$, as long as $x \neq y$, $x \neq 0^n$, and $y \neq 0^n$. Since each of these events occurs with probability $1/2^n$, we have:

$$\begin{aligned}
\mathbb{E}_{\mathbf{A}}[Z(\mathbf{A})^2] &\leq \mathbb{E}_{x,y}[\mathbb{E}_s[\mathbb{1}(\mathsf{map}(s), h(x))] \cdot \mathbb{E}_t[\mathbb{1}(\mathsf{map}(t), h(y))]] + \frac{3}{2^n} \\
&\leq \left(\frac{1}{3} + \frac{1}{2^n}\right)^2 + \frac{3}{2^n},
\end{aligned}$$

and then, by definition of the variance,

$$\begin{aligned}
\sigma^2 &= \mathbb{E}_{\mathbf{A}}[Z(\mathbf{A})^2] - \mathbb{E}_{\mathbf{A}}[Z(\mathbf{A})]^2 \\
&\leq \mathbb{E}_{\mathbf{A}}[Z(\mathbf{A})^2] - \left(\frac{1}{3} - \frac{1}{2^n}\right)^2 \\
&\leq \left(\frac{1}{3} + \frac{1}{2^n}\right)^2 - \left(\frac{1}{3} - \frac{1}{2^n}\right)^2 + \frac{3}{2^n} \\
&\leq \frac{4}{3 \cdot 2^n} + \frac{3}{2^n} = \frac{13}{3 \cdot 2^n} \leq \frac{5}{2^n}.
\end{aligned}$$

Lemma 4.1 immediately follows by applying a union bound. □

A.2 Proof of Lemma 4.2

By contradiction, let us assume there exists $f' \in \mathrm{GF}(q^\ell)[X_1, \ldots, X_n]$ of degree at most d such that

$$f'(x) = \mathsf{map}_p(x) \quad \forall x \in \mathcal{X}, \tag{A.1}$$

for some subset $\mathcal{X} \subseteq \{0,1\}^n$. Consider the extension $\mathbb{F} = \mathrm{GF}(q^\ell)^{p-1}$. By definition, $\mathbb{F}$ contains an element r of order p since p divides $x^{p-1} - 1$ for any $x > 0$ provided that p does not divide x. We apply the mapping $X_i \in \{0,1\} \mapsto Y_i = 1 + (X_i \cdot (r-1)) \in \{1, r\}$ to every indeterminate. This maps 0 to 1 and 1 to r. Then, f' is transformed into a polynomial $f \in \mathbb{F}[Y_1, \ldots, Y_n]$ over the Y_i's with the same degree, $\mathsf{map}_p(X)$ becomes $\prod_{i=1}^n Y_i$ (with $z \in [p]$ being mapped to r^z), and $\mathcal{X} \subseteq \{0,1\}^n$ becomes a set $\mathcal{Y} \subseteq \{1, r\}^n$ with same cardinality. Moreover, Eq. (A.1) becomes:

$$f(y) = \prod_{i=1}^{n} y_i \ , \forall y \in \mathcal{Y}. \tag{A.2}$$

Consider the set $\mathcal{F}_{\mathcal{Y}}$ of all functions $t\colon \mathcal{Y} \to \mathbb{F}$. For any function $t \in \mathcal{F}_{\mathcal{Y}}$, we uniquely associate a function $t' : \mathcal{X} \to \mathbb{F}$ such that $t'(x) = t(y)$ using the above mapping $x_i \mapsto y_i$, and we construct a multilinear[14] polynomial $P_{\mathcal{X}}$ over $\mathbb{F}$ of total degree at most $n/2 + d$ such that $P_{\mathcal{X}}(x) = t'(x)$ for all $x \in \mathcal{X}$. Thus $|\mathcal{F}_{\mathcal{Y}}|$ is at most the number of multilinear polynomials over $\mathbb{F}$ of total degree at most $n/2 + d$, which is $|\mathbb{F}|^{\sum_{i=0}^{n/2+d} \binom{n}{i}}$. As $|\mathcal{F}_{\mathcal{Y}}| = |\mathbb{F}|^{|\mathcal{Y}|}$, we then obtain $|\mathcal{Y}| \leq \sum_{i=0}^{n/2+d} \binom{n}{i}$.

It remains to show how to construct the polynomial $P_{\mathcal{X}}$. Consider a function $t \in \mathcal{F}_{\mathcal{Y}}$. We can write it as a (possibly huge) multilinear polynomial P over $\mathbb{F}$. Indeed, as we aim at building a polynomial $P_{\mathcal{Y}}$ that matches t over $\mathcal{Y} \subseteq \{1, r\}^n$, we can interpolate t using only multilinear monomials $((r-1)^n)^{-1} \prod_{b_i=1}(r - Y_i) \cdot \prod_{b_i=r}(Y_i - 1)$, for any $b \in \{1, r\}^n$, that map b to 1 and any $b' \neq b$ to 0. This polynomial can then be expanded as $P = \sum_{I \subseteq [n]} a_I \cdot \prod_{i \in I} Y_i$, with $a_I \in \mathbb{F}$. Moreover, for any $|I| > n/2$, we have $\prod_{i \in I} Y_i = \prod_{i \in [n]} Y_i \cdot \prod_{i \in \overline{I}} Y_i^{-1}$. Since $Y_i^{-1} = (r^{-1} - 1)X_i + 1$ (it can be easily verified by computing $Y_i \cdot Y_i^{-1}$) we obtain via Eq. (A.2):

$$P = \sum_{\substack{I \subseteq [n] \\ |I| \leq n/2}} a_I \cdot \prod_{i \in I} Y_i + f \cdot \sum_{\substack{I \subseteq [n] \\ |I| > n/2}} a_I \cdot \prod_{i \in \overline{I}} Y_i^{-1},$$

over $\mathcal{Y}$. Then, replacing Y_i's and Y_i^{-1}'s by their linear expression in the X_i's, we obtain a polynomial of degree at most $n/2 + d$ that matches t' over $\mathcal{X}$. This concludes the proof of Lemma 4.2. □

B On the Non-Existence of Depth-2 Strong PRFs

Here, we show that our candidates are not strong PRFs, and moreover, that a large class of depth-2 circuits does not contain strong PRFs. The distinguisher relies on a learning algorithm for automata with multiplicity from membership queries[15] by Bergadano and Varricchio [BV96]. We also show that generalizing this attack to using only random examples (i.e. to weak PRFs) would also break the LWR assumption for any polynomial parameters.

B.1 Adaptive Insecurity of our Candidate

We first state the adaptive attack against our candidates.

Lemma B.1 (Adaptive Attacks on Construction 3.1)**.** *There exists an adaptive polynomial-time (in n) distinguisher that breaks the pseudorandomness of Construction 3.1 with input length n. In fact, for general p, q, there exists an adaptive polynomial-time (in n, p, q) distinguisher that breaks the mod-p/mod-q generalization of Construction 3.1 (Remark 3.2) with input length n.*

The above lemma follows from the learnability of automata with multiplicity [BV96], and its proof just consists of constructing a polynomial-size (in s, n) automaton with multiplicity over $\mathbb{Z}_q$ that evaluates our candidate. Before detailing the proof, we recall the definition of an automata and automata with multiplicity.

[14] By multilinear, we mean that every monomial has degree at most 1 in each indeterminate (i.e., has the form $\prod_{i \in I} X_i$ for $I \subseteq [n]$.)

[15] We recall that having access to membership queries in learning theory corresponds to having adaptive oracle access to the target function ("concept") f. Specifically, in this model, the learning algorithm can make adaptive queries on inputs x to obtain $f(x)$.

Definition B.2 (Automaton)**.** An automaton is a 5-tuple $\mathcal{A} = (\Sigma, Q, T, I, F)$, where Σ is an alphabet, Q is a set of states, $T \subseteq Q \times \Sigma \times Q$ is a set of transitions, and $I, F \subseteq Q$ are a set of initial and final states, respectively. We say that a sequence $\pi = (q_1, x_1, q_2), \ldots, (q_\ell, x_\ell, q_{\ell+1})$ is an accepting path for $x \in \Sigma^\ell$ if $q_1 \in I$, $q_{\ell+1} \in F$, and $(q_i, x_i, q_{i+1}) \in T$ for all $i \in [\ell]$. We denote by $\Pi(x)$ the set of accepting paths for x. Then, for any $x \in \Sigma^\ell$, we define the evaluation

$$\mathcal{A}(x) = \begin{cases} 1 & \text{if } \Pi(x) \neq \varnothing \\ 0 & \text{otherwise} \end{cases}.$$

Definition B.3 (Automaton with Multiplicity)**.** An automaton with multiplicity is an automaton $\mathcal{A} = (\Sigma, Q, T, I, F)$ associated with a multiplicity (λ, μ, γ) over a field $\mathbb{F}$, where $\lambda \colon I \to \mathbb{F}$, $\mu \colon T \to \mathbb{F}$, and $\gamma \colon F \to \mathbb{F}$. For any $x \in \Sigma^\ell$, $\mathcal{A}(x)$ is then defined as:

$$\mathcal{A}(x) = \sum_{\substack{\pi \in \Pi(x) \\ \pi = (q_1, x_1, q_2), \ldots, (q_\ell, x_\ell, q_{\ell+1})}} \lambda(q_1) \cdot \prod_{i=1}^{\ell} \mu((q_i, x_i, q_{i+1})) \cdot \gamma(q_{\ell+1}).$$

We can now prove Lemma B.1 using the following theorem from [BV96].

Theorem B.4 (Learning Automaton with Multiplicity [BV96])**.** *There exists an efficient learning algorithm with membership queries for automata with multiplicity; in particular, the running time of the learning algorithm is polynomial in the size of the automaton.*

Proof of Lemma B.1 To prove the lemma, we only need to construct an automaton with multiplicity that evaluates our candidate mod-p/mod-q PRF. The idea is simple: for every output of the linear mapping (i.e. an inner product over $\mathbb{Z}_p^n$), we construct an automaton with multiplicity that returns the value of the mapping modulo q. Then, doing this for every output of the linear mapping and considering the union of all such automata, we obtain an automaton with multiplicity that evaluates precisely our PRF. The output of this automaton is the sum (over $\mathbb{Z}_q$) of all of the outputs of each automaton, which is precisely evaluating map_q on the output of the linear mapping.

Concretely, we describe the automaton for computing a single inner product $\langle k, x \rangle$, where k is a row of the key. We construct an automaton of size $np + 1$ to compute the inner product $\langle k, \cdot \rangle$ (over $\mathbb{Z}_p$) as follows:[16]

- Σ: the alphabet is $\mathbb{Z}_p$.
- Q: the automaton has $np+1$ states. The initial state is denoted $(0, 0)$. There are n successive layers, each with p states. We index these states by $[n] \times \mathbb{Z}_p$.
- T: transitions are defined by $T = \{((i, j), x, (i + 1, j + k_i \cdot x \bmod p)) \mid (i, j) \in Q\}$.
- I: the only initial state is $(0, 0)$.
- F: the final states consist of the states in the last layer: every state (n, j) where $j \in \mathbb{Z}_p$.

[16] Actually, as we are only interested in the result of the inner product modulo q, we could represent it by an automaton of size $nq + 1$ if $q < p$

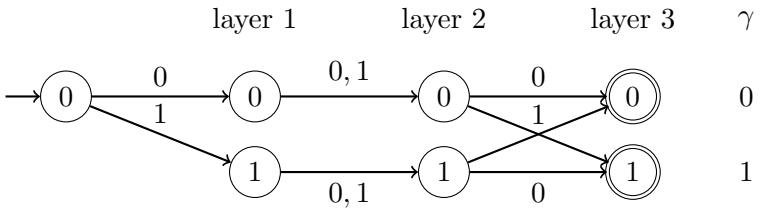

Figure 3: Automaton for computing $\langle (1,0,1), x \rangle$ over $\mathbb{Z}_2$. On the right, we show the multiplicities γ associated with the 2 final states (layer 3). The other multiplicities for the initial state and the transitions are all set to 1.

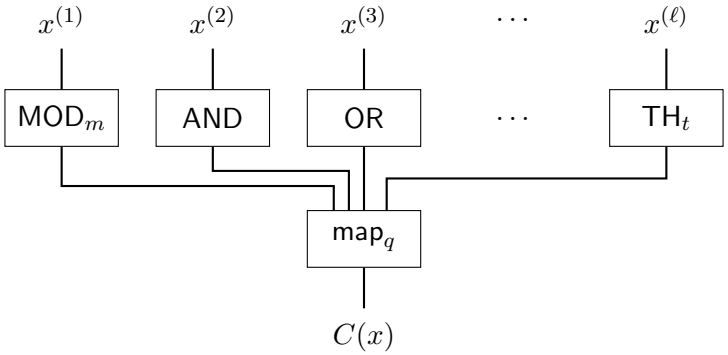

Figure 4: An example of depth-2 circuit that cannot be a strong pseudorandom function. Every $x^{(i)}$ corresponds to a subset of the input $x \in \{0,1\}^n$.

We define the multiplicity (λ, μ, γ) over $\mathbb{Z}_q$ as follows. The multiplicity of the starting states and the transitions are defined as the constant function 1. The multiplicity of the output states γ maps the node (n, j) to $j \bmod q$. It is easy to verify that on input $x \in \mathbb{Z}_p^n$, the above automaton computes the inner product $\langle k, x \rangle$ over $\mathbb{Z}_p$ and returns the output modulo q. We give a simple automaton with inputs of length $n = 3$ over a binary field $\mathbb{Z}_2$ in Figure 3. Lemma B.1 then follows from Theorem B.4. □

Remark B.5 (Generalizing the Attack)**.** This attack described in the proof of Lemma B.1 generalizes to a much larger class of depth-2 circuits. Specifically, as long as the outputs are computed as a (possibly weighted)[17] sum of values modulo some prime q and if the first layer only consists of functions that can be evaluated by automaton with multiplicity, we can mount the same attack. In particular, any gate of the form $\mathsf{AND}, \mathsf{OR}, \mathsf{MOD}_m$ for any (polynomial) m, EXACT_k, where $\mathsf{EXACT}_k(x)$ outputs 1 if $\mathsf{hw}(x) = k$ and 0 otherwise, and where hw denoting the Hamming weight, or TH_t gates, where $\mathsf{TH}_t(x)$ outputs 1 if $\mathsf{hw}(x) \geq t$ and 0, can be evaluated by an automaton with multiplicity. This rules out a large class of depth-2 circuits. An example of such a circuit is given in Figure 4.

[17]Specifically, if an input has weight c, it suffices to replicate the automaton associated to that input c times

B.2 Relation to the Learning with Rounding Assumption

While Lemma B.1 rules out strong pseudorandomness of our candidates (and more generally, a large class of depth-2 circuits computable by automaton with multiplicity), it critically relies on the adversary being able to make adaptive queries (Theorem B.4). Below, we show that strengthening the attack to also apply in the setting where the adversary only sees random samples (which would break weak pseudorandomness of our candidates) would give an attack on the learning with rounding (LWR) assumption [BPR12] with polynomial parameters. We first recall the LWR problem.

Definition B.6 (Learning with Rounding (LWR) [BPR12])**.** Let $n, p, q \in \mathbb{N}$ be positive integers. The learning with rounding (LWR) assumption $\mathsf{LWR}(n, p, q)$ assumption states that the following two distributions are computationally indistinguishable:

$$\left(\vec{a}, \lceil \langle \vec{a}, \vec{s} \rangle \rfloor_{q,p}\right) \quad \text{and} \quad \left(\vec{a}, r\right),$$

where $\vec{a} \xleftarrow{R} \mathbb{Z}_q^n$, $\vec{s} \xleftarrow{R} \mathbb{Z}_q^n$, $r \xleftarrow{R} \mathbb{Z}_p$, and $\lceil \cdot \rfloor_{q,p}$ denote the rounding operator that maps values in $\mathbb{Z}_q$ to $\mathbb{Z}_p$. The assumption naturally extends to the setting where the distinguisher is given many samples (i.e., distinguish $((\vec{a}_1, \langle \vec{a}_1, \vec{s} \rangle \rfloor_{q,p}), \ldots, (\vec{a}_m, \langle \vec{a}_m, \vec{s} \rangle \rfloor_{q,p}))$ from $((\vec{a}_1, r_1), \ldots, (\vec{a}_m, r_m))$).

Lemma B.7 (Connection to Learning with Rounding)**.** *Suppose there exists an algorithm that learns automaton with multiplicity over $\mathbb{Z}_p$ given only random samples. If the running time of the algorithm is $T(\ell, s)$, where ℓ is the size of the alphabet and s is the size of the automaton, then there exists an algorithm that breaks $\mathsf{LWR}(n, p, q)$ in time $T(q, q^2 n)$. In particular, if T is a polynomial, then LWR does not hold for any polynomial parameters $p, q, n = \mathrm{poly}(\lambda)$.*

Proof. The lemma follows from the same argument as in the proof of Lemma B.1. Specifically, in the proof of Lemma B.1, we showed how to compute an inner product over $\mathbb{Z}_q$ using an automaton with multiplicity. For LWR samples, only a single inner product needs to be computed, so only a single automaton (instead of an union in the previous lemma) is needed. The only difference with the previous proof is how we define the multiplicity (λ, μ, γ). Again, we define the multiplicity λ, μ for the input nodes and the transitions, respectively, to be the constant function equal to 1. The output multiplicity γ is defined to be the rounding function that maps a final state (n, j) with $j \in \mathbb{Z}_q$ to $\lceil j \rfloor_{q,p} \in \mathbb{Z}_p$. It is easy to verify the correctness of the evaluation. In Figure 5, we give an example for $p = 2, q = 3, n = 3, \vec{s} = (2, 0, 1)$. □

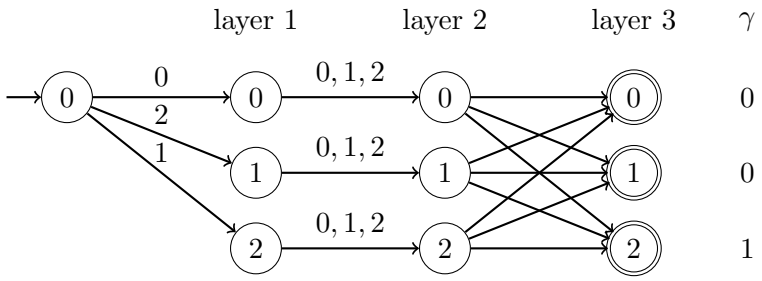

Figure 5: Automaton evaluating LWR samples for $p = 2, q = 3, n = 3, s = (2, 0, 1)$. On the right, we give the multiplicities γ associated with the 3 final states (layer 3). The rest of the multiplicities (for transitions and initial state) are set to 1.