A Stateless Cryptographically-Secure Physical Unclonable Function

Charles Herder, Ling Ren, Marten van Dijk, Meng-Day (Mandel) Yu, and Srinivas Devadas

***Abstract*—We present the first stateless construction of a cryptographically-secure Physical Unclonable Function. Our construct requires no non-volatile (permanent) storage, secure or otherwise, and its computational security can be clearly reduced to the hardness of Learning Parity with Noise (LPN) in the random oracle model. The construction is "stateless," because there is *no* information stored between subsequent queries, which mitigates attacks against the PUF via tampering. Moreover, our stateless construction corresponds to a PUF whose outputs are free of noise because of internal error-correcting capability, which enables a host of applications beyond authentication. We describe the construction, provide a proof of computational security, and present experimental evidence that this construct is viable.**

I. Introduction

A. Background and Motivation

SILICON Physical Unclonable Functions (PUFs) are a promising innovative primitive that are used for **authentication** and **secret key storage** without the requirement of secure memory or expensive tamper-resistant hardware [25], [54]. This is possible, because instead of storing secrets in digital memory, PUFs derive secrets from the physical characteristics of the integrated circuit (IC). Silicon PUFs rely on the fact that even though the mask and manufacturing process is the same among different ICs, each IC is actually slightly different due to normal manufacturing variability. PUFs leverage this variability to derive 'secret' information that is unique to the chip (a silicon 'biometric'). Due to the manufacturing variability, one cannot manufacture two chips with identical secrets, even with full knowledge of the chip's design. PUF architectures that exploit different types of manufacturing variability have been proposed. In addition to gate delay, there are PUFs that use the power-on state of SRAM, threshold voltages, and many other physical characteristics to derive the secret.

The (informal) requirements for a PUF are:

1) Upon being given a challenge, the PUF produces a response, and no other data about the internal functionality of the PUF is revealed.
2) Large enough challenge-response space such that an adversary cannot enumerate all challenge-response pairs within reasonable time.
3) An adversary given a polynomial number of challenge-response pairs cannot predict the response to a new, randomly chosen challenge.
4) Not feasible to manufacture two PUFs with the same responses to all challenges.

These requirements correspond to what has been sometimes called a strong PUF in the literature.

The silicon PUF approach is advantageous over standard secure digital storage for several reasons:

- Since the 'secret' is derived from physical characteristics of the IC, the chip must be powered on for the secret to reside in digital memory. Any physical attack attempting to extract *digital* information from the chip therefore must do so while the chip is powered on.
- Authentication of devices and secure communication to devices do not require embedding and permanently storing secrets in the devices. Devices therefore do not require non-volatile memory, which is more expensive and not available in all manufacturing processes. For example, EEPROMs require additional mask layers, and battery-backed RAMs require an external always-on power source.

PUFs can therefore serve as one way to address the growing counterfeit electronics problem [28].

For authentication, PUFs usually adopt a simple challenge-response protocol. An entity, call it the verifier, collects challenge-response pairs in a secure location when in possession of the PUF. At any later point of time, to authenticate a remote device, the verifier sends a challenge to the device and asks for the response[1]. If the response "matches" the stored response, verification is successful, else it is not.

The simplicity and power of the above protocol motivated the construction of many candidate silicon PUFs. We note that the physical system and the protocol are both *stateless* (i.e., store *no* data between subsequent queries and do not require non-volatile digital storage). Unfortunately, none of the candidate constructions have a proof of computational security, and further, most, if not all, of them have been shown to be susceptible to machine learning attacks (cf. Section III). In the context of stateless PUFs, Gassend et al. [25] write: "An important direction of research is to find a circuit that is provably hard to break ...". In this paper we accomplish this objective.

B. Physically Obfuscated Keys

Physically Obfuscated Keys (POKs) (e.g., [45]) predate silicon PUFs and have sometimes been called weak PUFs in the literature. The (informal) requirements for a POK are less stringent than the requirements for a PUF. They are:

1) Small number of challenge-response pairs (linearly related to the number of components whose behavior depends on manufacturing variation).
2) Responses are unpredictable and depend strongly on the innate manufacturing variability of the device.

Both PUFs and POKs rely on analog physical properties of the fabricated circuit to derive secret information. Naturally, these analog properties have noise and variability associated with them. As environmental parameters vary, so does the "digital fingerprint" measured by the PUF. If the parameters vary too much, the PUF or POK response will change. In the

[1]To defeat man-in-the-middle attacks, challenges should not be repeated.

PUF authentication protocol, "matching" means the response stored by the verifier and the regenerated response are within a chosen threshold.

If a POK's responses are exposed, it can be easily cloned by enumerating all challenge-response pairs and storing them in a ROM. However, a PUF can be built using *public* tamper-proof storage, error correction logic, cryptographic primitives, and a POK as we describe below. Choose a POK output or bits derived from the POK output during a *provisioning* step as the secret key, and error correct the POK outputs when they are regenerated so the key is always the same. Assuming the public helper data associated with the error correcting code does not give away (too much) information about the POK outputs and therefore the secret key, one can use the secret key as one input to a one-way hash function to build a PUF. This PUF is noise-free through error correction using helper data. The helper data can be stored on the verifier side, though it is usually more convenient to store it on the device itself. However, even if the helper data is stored off-device, there is still a requirement for storage on the PUF device.

The reason is subtle: generation of helper data can only be done once or a very limited number of times, because the helper data leaks information about the POK outputs as mentioned above. We know of no information-theoretically secure error correction scheme that can repeatedly generate helper data an arbitrary number of times under potentially different environmental conditions where POK outputs change slightly or significantly. Such a scheme is difficult to envision because the POK has finite entropy. This means that once helper data is generated, this provisioning functionality needs to be turned off; else the system can be broken. This implies an irreversible fuse, i.e., *storage that is tamper proof*. Else, an attacker with physical access just has to modify one bit of storage to potentially break the system by rerunning the provisioning step, as opposed to having to read volatile values in a stateless PUF.

C. Overview of our Approach

In this paper, we show how to build a stateless PUF with a provable security reduction. Our construction has internal error correction, and therefore the PUF outputs will be completely stable, assuming the error correction range matches the requirement posed by environmental variation. Our PUF therefore is a controlled PUF [24], which is much more powerful than a conventional PUF that can only be used for authentication (cf. Section IV-E3).

We accomplish our task in two steps. First, we show given a POK, how to set up a restricted version of a Learning Parity with Noise (LPN) problem such that a secure challenge-response protocol is enabled. The restriction of a constant (hardcoded) $\mathbf{A}$ matrix in LPN (cf. Section II-B) is crucial to our security argument. There are two modes of operation for this PUF. The first is to generate a challenge-response pair, and the second is to return a response given a challenge. We give a reduction that the challenge-response protocol is secure under the random oracle model given the hardness of LPN. The protocol fails if the POK is noisy, i.e., the PUF responses will be completely different even with a small amount of noise. We could use an enumerative approach to correct $O(\log m)$ errors in the m-bit POK output in polynomial time, as discussed by Fuller et al. in [21]. Unfortunately, we invariably have $O(m)$ errors in a POK, which will require exponential time in an enumerative approach.

In the second step, by making an assumption about the characteristics of the POK, namely, that it can provide "confidence" information, we show how we can correct $O(m)$ errors in polynomial time. Now, in the first mode of operation, all challenge-response pairs generated are reliable in that they always produce the same responses for the corresponding challenges when subsequently regenerated in the second mode of operation. The challenge itself contains necessary helper data for noise-free regeneration.

The key insight that makes the difference is that *use without storage* of confidence information from a noisy biometric source can exponentially increase error tolerance. The confidence information can thus be viewed as a *noise-avoiding trapdoor*.[2]

Abstractly, confidence information can be thought of as information measured from the biometric source in addition to the output bits that represents which bits of the output have higher/lower probability of error. This information is available from a large number of biometric sources, since many sources define their output bits in terms of whether an analog/digital value is greater/less than some threshold. In this case, the confidence information would be the distance of the analog/digital value from that threshold. One example corresponds to simply making repeated measurements of a biometric source (e.g., SRAM state [32]) and taking a majority vote to produce a bit. The degree of majority would be the confidence information. The most natural example that we know of corresponds to the ring oscillator POK (cf. Section II-A) used in our case study (cf. Section VIII), which produces bits based on the sign of the difference of oscillator frequencies.

We stress that while previous constructions leveraging confidence information published their confidence information [13], [15], [46], [47] (which requires persistent memory storage and also affects the information-theoretic security argument) the confidence information in our construction is not persistent, rather, it is regenerated on each key recovery. Our PUF remains stateless.

D. Our Contributions

We defer a detailed comparison to prior work to Section III, and elaborate on our contributions below.

1) We provide a construction of a stateless PUF using a restricted version of the LPN problem with a constant $\mathbf{A}$ matrix and show that breaking the PUF requires breaking LPN under the random oracle model (cf. Section IV).
2) We provide an error correction scheme for key recovery that is able to correct $\Theta(m)$ errors in the m bits of measured biometric data in polynomial time, allowing efficient implementation of (1). Two novel ideas enable this powerful result.
   - First, we take advantage of the confidence information (present in most biometric sources) as a trapdoor to a hard problem. The confidence information is never

[2] At first glance, our result seems contradictory to recent work in computationally secure key extraction [21], which concluded that there was no improvement to be gained beyond information-theoretic key extraction from a viewpoint of error tolerance. We have, however, made an additional assumption about the biometric source.

stored; it is regenerated (and may change at each key extraction) and then erased (cf. Section V).
- Second, compared with traditional LPN/LWE cryptosystems, we use m (the number of equations in the LPN/LWE problem) in a novel way – to provide redundancy – and show that setting $m = \Theta(n^3)$ results in a negligible failure probability for key recovery even with $\Theta(m)$ errors, where n is the size of the output key (cf. Section VI).

3) LPN is hard given i.i.d. properties of the biometric source. We give a reduction for a large class of entropy source distributions (cf. Definition VII.1) that are significantly more relaxed (cf. Section VII).
4) We provide experimental evidence that our PUF construction is viable and works under significant environmental variation (cf. Section VIII).

E. Organization

We give background for ring oscillator POKs and LPN in Section II. We discuss related work in Section III. Section IV gives formal definitions for stateless PUFs, our construction of a stateless PUF and shows a security reduction to LPN hardness under the random oracle model. This scheme only works for constituent POKs with limited noise. Section V describes how noisy POK outputs can be corrected securely. We analyze the reliability of key regeneration in Section VI, where we show that the confidence information serves as a trapdoor. We give a detailed security analysis in Section VII, and relax the assumptions on the biometric bits. We also provide quantitative analysis on various PUF parameters. We show in Section VIII that an efficient, computationally secure stateless PUF can be built using ring oscillator POKs. We conclude the paper in Section IX.

II. Background

A. Ring Oscillator POK

Our case study will be based on the Ring Oscillator (RO) POK, which generates bits by comparing the frequencies of two ring oscillators that are identical by design, yet whose frequencies vary due to manufacturing variation. Each POK output bit is simply determined by which oscillator is faster. It was first observed in [58] that if the difference in counts between the two ring oscillators is large, then one can have higher confidence that environmental changes are unlikely to cause the output bit to flip erroneously when measured at a later time. This difference will be our confidence information (cf. Figure 1). While there have been improvements in ring oscillator structures (e.g., [22]), our case study uses the basic structure of Figure 1.

B. Learning Parity with Noise

The Learning Parity with Noise (LPN) problem is a famous open problem that is widely conjectured to be hard, as the best known algorithm is slightly subexponential ($2^{\Omega(n/\log n)}$) [10], [5]. As a result, this problem has since been used as the foundation of several cryptographic primitives [33], [3], [2], [9].

The problem is posed as follows. Let $\mathbf{s} \in \{0,1\}^n$ be chosen uniformly at random. Let $\mathbf{A} \in \{0,1\}^{m \times n}$ be uniformly random, $m \geq n$. Let $\mathbf{e} \in \{0,1\}^m$ be chosen from a distribution χ. Finally, define $\mathbf{b} \in \{0,1\}^m$ as: (where $\cdot$ is a dot product)

$$\begin{aligned} \mathbf{b}_1 &= \mathbf{A}_1 \cdot \mathbf{s} + \mathbf{e}_1 \mod 2 \\ \mathbf{b}_2 &= \mathbf{A}_2 \cdot \mathbf{s} + \mathbf{e}_2 \mod 2 \\ \vdots &= \vdots \\ \mathbf{b}_m &= \mathbf{A}_m \cdot \mathbf{s} + \mathbf{e}_m \mod 2 \end{aligned}$$

The problem is to learn $\mathbf{s}$ given only the values of $\mathbf{b}$ and $\mathbf{A}$. When each $\mathbf{e}_i$ is distributed according to probability distribution χ, where χ has each bit i.i.d. with probability τ of being 1 and probability $1-\tau$ of being 0 with non-negligible τ and $1-\tau$, there is no known polynomial-time algorithm that solves this problem.

Conjecture II.1 (LPN Hardness [33])**.** *There is no algorithm that solves an LPN problem instance* $(\mathbf{A}, \mathbf{b}, \chi)$*, where* $\mathbf{s}$ *and* $\mathbf{A}$ *are uniformly random, in time* $\mathsf{poly}(n, 1/(\frac{1}{2}-\chi))$ *with non-negligible probability in* n*.*

The LPN problem can be thought of as a special case of the Learning With Errors (LWE) problems discussed by Regev [53], by allowing the equations to instead be modulo a prime number q (as opposed to 2). However, Regev's reduction to the independent shortest vector problem (ISVP) does not apply to the LPN case. Therefore, the difficulty of solving LPN is a separate conjecture from the difficulty of solving LWE. This paper will focus on a stateless PUF based on LPN, but our construction can be easily extended to the LWE case.

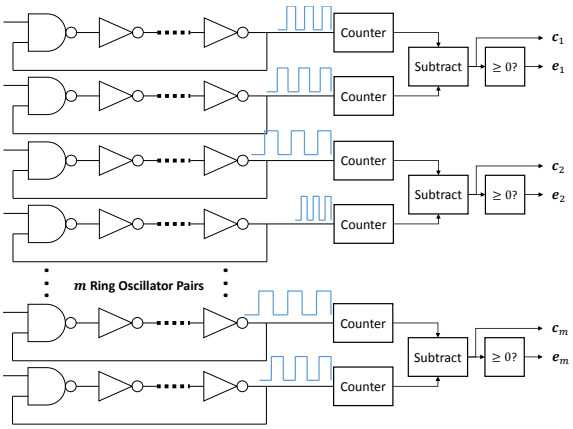


Fig. 1: A basic Ring Oscillator POK with m differential pairs. Note that in addition to the output bits $\mathbf{e}_i$, confidence values $\mathbf{c}_i$ may be made available to the surrounding logic. These confidence values are in the form of the actual differential count between the two ring oscillators, while the POK output bits $\mathbf{e}_i$ correspond to whether the differential count is greater/less than 0.

III. Related Work

A. PUF Proposals

Although many of the architectures that integrate PUFs into existing IC technology are new, it should be noted that the concepts of unclonability and uniqueness have been used extensively in the past for other applications [38]. For example, "Unique Objects" are well defined as objects with a unique set

of properties (a 'fingerprint') based on the unique disorder of the object [54]. One example of early usage of unique objects for security was proposed for the identification of nuclear weapons during the cold war [27]. One would spray a thin coating of randomly distributed light-reflecting particles onto the surface of the nuclear weapon. Since these particles are randomly distributed, the resulting interference pattern after being illuminated from various angles is unique and difficult to reproduce. Unique objects were termed Physical One-Way Functions and popularized in 2001 [51]. However, to our knowledge, none of these proposals has an associated computational security argument that shows hardness of model-building or machine learning attacks.

Silicon PUFs, introduced in [25], do not require an external measurement apparatus unlike the proposals described above. In the past several years, there have been several proposals for candidate silicon PUF architectures. These include the family of proposals corresponding to the Arbiter PUF [26], feedforward Arbiter [44] and XOR Arbiter PUF [58]. Machine learning attacks such as those of [55] and [6] have successfully attacked these constructions to create software clones. While other constructions using nonlinear circuit elements (e.g., [43], [41], [50]) have not yet been broken to our knowledge, these constructions do not as yet have clear computational security reductions.

B. Error Correction for Silicon PUFs

Silicon PUF key generation was first introduced using Hamming codes in [23] and more details were presented in [57]. The security argument is information-theoretic. Specifically, if one requires a k-bit secret from n bits generated by the PUF, then at most $n - k$ bits could be exposed. The number of correctable errors is quite limited in this approach.

C. Fuzzy Extractors for Silicon PUFs

Fuzzy extractors [20] convert biometric data (either human or silicon) into reproducible uniform random strings, which can serve as secret keys in cryptographic primitives to encrypt and authenticate user data.

Fuzzy extractors typically have two phases: a secure sketch (error correction) phase and a privacy amplification (hashing) phase. The secure sketch phase focuses on the recovery of biometric data w. It first outputs a sketch h (also called "helper data") for w. Then, given h and a future measurement w' close to w, it recovers w. The sketch is secure if it does not reveal much about w: w retains much of its entropy even if h is known. This means that h can be stored in public without compromising the privacy of w. However, in typical biometric applications, w does not have full entropy, so we need the privacy amplification phase to compress w prior to obtaining a cryptographic key. In the fuzzy extractor framework, it is possible to extract near-full-entropy keys from a biometric source while maintaining information-theoretic security.

The information-theoretic security, however, comes at a high cost in terms of the raw entropy required and the maximum tolerable error rate. The secure sketch phase is well known to lose significant entropy from the helper data h, especially as measurement noise increases. Even in cases where entropy remains after error correction (e.g. [48]), there is not enough entropy remaining to accumulate the 128-bits of entropy in an information-theoretic manner during the privacy amplification phase. According to [39], the entropy loss associated with the use of the information-theoretic entropy accumulator alone is $\geq$ 128 bits due to the leftover hash lemma.

Works on fuzzy extractors for silicon PUFs can be classified based on the additional assumptions they require:

Perfectly i.i.d. Entropy Source. There were several works that created helper data that is information-theoretically secure. [61] uses PUF error correction helper data called Index-Based Syndrome (IBS), as an alternative to Dodis' code-offset helper data. IBS is information-theoretically secure, under the assumption that PUF output bits are independent and identically distributed (i.i.d.). Given this i.i.d. assumption, IBS can expose more helper data bits than a standard code-offset fuzzy extractor construction. Efficiency improvements to IBS that maintained information-theoretic security are described in [30] and [31].

A soft-decision PUF error correction decoder based on code-offset was described in [46], [47] where the confidence information part of the helper data was proven to be information-theoretically secure under an i.i.d. assumption (the security of the remaining redundancy part associated with the code-offset was not as rigorously addressed in either paper).

We note that while these works created practical implementations based on a provably secure information-theoretic foundation, these works did not explicitly address the full key generation process (secure sketch + privacy amplification); they addressed only the error correction (secure sketch) phase. Further, they need the strong assumption on PUF output bits being i.i.d., which allows them to publicly reveal the confidence information. Indeed, silicon biometrics are not necessarily i.i.d., and attacks have therefore been performed, e.g., [7]. Our approach achieves the same advantage of using confidence information, but it does not reveal this information. Therefore, our proposal remains secure for non-i.i.d. entropy sources (cf. Definition VII.1).

Computational Security Based on Machine Learning Heuristics. There were several works [52] [62] [63] that created helper data that is heuristically secure based on results of state-of-the-art machine learning attacks on PUFs [56]. These designs used a candidate strong PUF based on XORs [58] but leak only a limited number of PUF response bits as helper data to generate a key. While several years of attacks by several groups around the world have established the heuristic security of leaking a limited number of bits from a candidate strong PUF [17], [34], [55], [56], [59], [6], there is not yet a proof to reduce this difficulty into a computational hardness assumption accepted by the cryptography community. These works are also limited in scope in that they do not explicitly address the full key generation processing, but address only the error correction phase.

Secure Sketch + Privacy Amplification. To the best of our knowledge, there is one paper that attempted to implement and address the security associated with both stages of a PUF fuzzy extractor [48]. The paper accounted for the information-theoretic loss of the error correction helper data, using code-offset syndrome [20], but did not have sufficient entropy left over from the secure sketch phase to implement an information-theoretically secure privacy amplification stage

and instead opted for a more efficient implementation using a lightweight hash called SPONGENT [11] as an entropy accumulator.

Others. There are several works [12], [60] that cite the need for a fuzzy extractor [20] but do not fully implement it in a manner that fully accounts for the information-theoretic entropy loss. There are some works that describe a PUF key generation scheme without accounting for entropy loss associated with error correction and privacy amplification [49].

We are unaware of any information-theoretically secure extractors whose key provisioning step can be repeated arbitrarily many times in different environmental conditions as we require in our stateless construction of Section IV. In prior schemes, once a key or keys have been provisioned, this functionality has to be turned off. Further, the system has to be made secure against malicious helper data (cf. Section III-F).

D. Fuzzy Extractors for Human Biometrics

Many iris biometrics that leverage fuzzy extractors [13], [15], [29] require the use of public "mask" data indicating which bits are generally more reliable. There is no mathematical justification for why these data do not reveal any information about the key or individual. In some instances, this limitation is recognized, and the mask is also required to be kept secret by the user [29], negating to a certain extent the usefulness of the biometric.

This mask information is similar to the confidence information that our approach leverages. The difference is that we do not reveal the confidence information to the adversary or require secure storage (cf. Section V).

E. Computational Fuzzy Extractors

Given our discussion above on information-theoretically secure fuzzy extractors, a more efficient key extraction framework that can be used repeatedly and which is based on an established computational hardness assumption is compelling.

Fuller et al. [21] give a computational fuzzy extractor based on LWE. In Fuller et al.'s scheme, the output entropy improves; the error correction capacity, however, does not. Indeed, Fuller et al. show in their model that computational fuzzy extractors are subject to the same error correction bounds as information-theoretic extractors. Their construction therefore requires exponential time to correct $\Theta(m)$ errors, where m is the number of bits in the biometric source.

We use Fuller et al.'s LWE construction translated to LPN, but require a restricted version of LPN with a hardcoded $\mathbf{A}$ matrix in our stateless PUF construction of Section IV. This restriction is important so we can repeatedly generate secret keys ($\mathbf{s}$'s) and expose the LPN public keys ($\mathbf{b}$'s) for the same noise ($\mathbf{e}$), while maintaining security based on hardness of LPN.

Further, our work changes the fuzzy extractor model and leverages the confidence information (common in many biometric sources) to correct $\Theta(m)$ errors in polynomial time. Fuller et al. state that it is impossible to overcome an exponential complexity in correcting a linear number of errors, since there is no place to securely store a trapdoor in a fuzzy extractor. We have solved that problem for certain kinds of biometric sources by recognizing that the biometric source itself, with its dynamically regenerated confidence information that does not require persistent storage memory can in fact serve as a trapdoor (cf. Sections V and VI). Finally, we show that security can be maintained even if the bits generated by the biometric source are correlated (cf. Definition VII.1).

F. Helper Data Manipulation

The issue of helper data manipulation has been addressed in the biometric realm with robust fuzzy extractors [14] [19]. Their use of a helper data hash do not address recent helper data manipulation attacks in [36] [18], including ones that take advantage of the linear, bitwise-XOR nature of code-offset helper data as applied to linear error correction codewords.

In our PUF of Section IV, the helper data comprises (part of) the challenge. Since in LPN-based fuzzy commitment the key is uncorrelated computationally to the helper information, our scheme can authenticate the helper information in a computationally secure manner via a keyed-hash message authentication code such as HMAC [40]. This results in a scheme that is secure in a computational sense against active adversaries that modify the helper data.

IV. Stateless PUF Construction

Following the formalization provided in [4], define a Physically Obfuscated Key (POK) as a Physical Function (PF) wherein there is only one challenge (the challenge space is of cardinality 1). A single physical function returns a single m-bit response, denoted as $\mathbf{e}$ in this paper.

If the PF response is stable, then constructing a stable secret key or strong PUF would be trivial. Unfortunately, the PF response in practice will be slightly different each time the physical function is called due to internal noise. Error-correcting the PF response to tolerate the noise is a major challenge. With this in mind, we will present the framework of our construction in this section and omit the details on error correction (as if the PF response were stable) to simplify presentation, We will address the error correction problem in Section V where we propose a novel scheme tailored to our construction that can correct up to a constant fraction of errors.

A. POK Extraction

Using the Physical Function (PF), the first step is to construct a Physical Function System (PFS) that, using helper information, reliably reconstructs a key $\mathbf{s}$.

To do this, consider the fuzzy commitment scheme described in [21]. This fuzzy extractor leverages LWE to perform fuzzy commitment, allowing the extraction of a pseudorandom string from fuzzy data. Informally, the key result (Theorem 4.7 of [21]) states that assuming the hardness of LWE, there exists a choice of parameters for which their fuzzy extractor construction is secure.

This work will begin by translating the above extractor to the learning parity with noise (LPN) problem discussed in Section II-B, and formalized in Conjecture II.1.

Construction IV.1. *Let k be a security parameter, and let $n = \mathsf{poly}(k)$, and $m \geq n$. Define $(\mathbf{A}, \mathbf{b}) \leftarrow \mathsf{Gen}(1^k)$, and $\mathbf{s} \leftarrow \mathsf{Rep}(\mathbf{A}, \mathbf{b})$ as follows:*

procedure $(\mathbf{A}, \mathbf{b}) \leftarrow \mathsf{Gen}(1^k)$
  Input $\mathbf{e} \in \{0,1\}^m$ from the PF (modeled by some distribution χ over $\{0,1\}^m$), to be indexed by $i \in 1 \dots m$.
  Sample $\mathbf{A} \in \{0,1\}^{m \times n}$ uniformly at random, to be indexed by $i \in 1 \dots m$.
  Sample $\mathbf{s} \in \{0,1\}^n$ uniformly at random.
  Compute $\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}_i$.
  return $(\mathbf{A}, \mathbf{b})$.
end procedure

procedure $\mathbf{s} \leftarrow \mathsf{Rep}(\mathbf{A}, \mathbf{b})$
  Input $\mathbf{e}'$ from the PF, to be indexed by $i \in 1 \dots m$.
  Input $\mathbf{A}$, $\mathbf{b}$.
  Let $\mathbf{s} = \mathsf{Decode}_t(\mathbf{A}, \mathbf{b}, \mathbf{e}')$.
  return $\mathbf{s}$.
end procedure

This construction is exactly analogous to Construction 4.1 of Fuller [21], translated to LPN instead of LWE (all equations are $\mod 2$ instead of $\mod q$). Therefore, we state the following theorem without proof, as it is analogous to Theorem 4.7 of [21], that the security of Construction IV.1 is equivalent to Definition 2.5 of [21] if Conjecture II.1 is true.

Theorem IV.2. *Let k be a security parameter, and let the number of errors $t = c \log k$ for some constant c.*

If Conjecture II.1 is true, then there is a setting of $n = \mathsf{poly}(k)$ for which there exists $\epsilon = \mathsf{neg}(k)$ such that the following is true: For any randomized circuit size $s = \mathsf{poly}(k)$, Construction IV.1 is a $(\{0,1\}^m, \chi, \{0,1\}^n, t)$ fuzzy extractor that is (ϵ, s)-hard, with error $\delta = e^{-\Omega(k)}$ (this is Definition 2.5 of [21]).

Decode entails solving an $n \times n$ linear system to obtain $\mathbf{s}$ after selecting n out of m rows in $\mathbf{A}$ and the corresponding rows in $\mathbf{b}$, and runs in polynomial time if the $\mathbf{e}_i$ values are regenerated exactly. If m is the number of biometric bits in the POK, one can correct for $t = \Theta(\log m)$ errors by enumerating through possible combinations (as proposed by Fuller et al. in [21]). We discuss a critical contribution in Section V that allows for the correction of $\Theta(m)$ errors in polynomial time.

The POK in combination with the above extractor represents the PFS (Physical Function System).

B. Stateless PUF Definition

A *Stateless PUF* will be a pair of functions $\mathbf{PUF} = \{\mathsf{Gen}, \mathsf{Ver}\}$, where Gen is responsible for generating and outputting challenge-response pairs, while Ver inputs a challenge, and outputs a response. The intent is for Gen to be called multiple times by a verifier over a secure channel with the Stateless PUF to obtain a collection of challenge/response pairs. At a later time, the verifier will send one of these challenges to the PUF over an insecure channel, to which the PUF must generate the correct response. A challenge-response pair therefore can only be used once by Ver.

Definition IV.3. *A (m, l, χ) stateless PUF is a pair of randomized probabilistic polynomial time procedures $\{\mathbf{b}, \mathbf{D}\} \leftarrow \mathsf{Gen}_{\mathrm{PF}}(1^k)$, and $\mathbf{D} \leftarrow \mathsf{Ver}_{\mathrm{PF}}(\mathbf{b})$ where*

- *The* challenge-response generation *algorithm $\{\mathbf{b}, \mathbf{D}\} \leftarrow \mathsf{Gen}_{\mathrm{PF}}(1^k)$ takes as the argument the security parameter k. It returns two strings $\mathbf{b} \in \{0,1\}^m$, $\mathbf{D} \in \{0,1\}^l$, with $l = \mathsf{poly}(k)$. The subscript* PF *corresponds to the POK contained within the PUF. That is, each PUF manufactured will have a unique POK according to distribution χ over $\{0,1\}^m$ due to manufacturing variation.*
- *The* verification *algorithm $\mathbf{D} \leftarrow \mathsf{Ver}_{\mathrm{PF}}(\mathbf{b})$ takes as argument the bit string $\mathbf{b} \in \{0,1\}^m$, and returns the bit string $\mathbf{D} \in \{0,1\}^l$. Just as in the above algorithm,* PF *refers to the unique POK data contained within the PUF.*

Now we define the security of the Stateless PUF ($\mathrm{s-uprd}$ refers to "strong unpredictability" as defined in [4]).

Definition IV.4 (Stateless PUF Strong Security)**.** *A stateless PUF is ϵ-secure with error δ if $\mathsf{Pr}\left[\{\mathbf{b}, \mathbf{D}\} \leftarrow \mathsf{Gen}_{\mathrm{PF}}(1^k) : \mathbf{D} = \mathsf{Ver}_{\mathrm{PF}}(\mathbf{b})\right] > 1 - \delta$ and for all PPT A, $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) < \epsilon$, which is defined in terms of the following experiment, where $H(\cdot)$ is a random oracle.*

procedure $\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A})$
  Run $\mathsf{A}^{\mathsf{Gen}_{\mathrm{PF}}(\cdot), \mathsf{Ver}_{\mathrm{PF}}(\cdot)}$
  if A returns $\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}$ such that:
    - A did not query $\mathsf{Ver}_{\mathrm{PF}}$ with $\mathbf{b}$.
    - $\mathsf{Gen}_{\mathrm{PF}}$ did not return $\{b, H(\mathbf{s}, \mathbf{b})\}$.
    - $\mathsf{Ver}_{\mathrm{PF}}(\mathbf{b}) = H(\mathbf{s}, \mathbf{b})$.

  then
    return 1
  else
    return 0.
  end if
end procedure

The s-uprd *advantage of* A *is defined as*

$$\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(A) = 1\right] \tag{1}$$

While other formalizations of PUF system security have been proposed [4], ours is slightly different in that in the above case, there is *no distinction* between helper data and challenge data. Moreover, the PUF is responsible for generating both the challenge and the response for the verifier to use later.

One key recognition in the above definition is that there is *no provisioning stage*. The algorithms Gen and Ver may be called in arbitrary order as many times as required. Put differently, there is no stage at which a secret is programmed into the device or an irreversible operation is performed on the device. This is critical, as the overall system can therefore be stateless, and not have to have any additional protections against adversaries attempting to break the provisioning logic of the device.

The formalism of manufacturing unclonability remains the same as that put forth in [4].

C. Our Construction

Given the full description of the POK and extractor given in Section IV-A and the definition of a Stateless PUF in Section IV-B, we may now provide concrete constructions that define the randomized algorithms for both Gen, Ver.

Construction IV.5 (LPN Stateless PUF)**.** *Let k be a security parameter, with $n = \mathsf{poly}(k)$, and $m \geq n$. Define $\mathbf{A} \in \{0,1\}^{m \times n}$ be a* **constant***, publicly known, uniformly random matrix row-indexed by i from 1 to m. Let both algorithms have access to the random oracle $H(\cdot)$.*

1: **procedure** $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\} \leftarrow \mathsf{Gen}_{\mathrm{PF}}(1^k)$
2: Generate $\mathbf{s} \in \{0,1\}^n$ uniformly at random.
3: Regenerate $\mathbf{e} \in \{0,1\}^m$ by using PF, the Physical Function/POK (e.g., Ring Oscillators or SRAM).
4: Generate $\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}_i$.
5: **return** $\{\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}, H(\mathbf{s})\}$.
6: **end procedure**

1: **procedure** $\mathbf{D_s} \leftarrow \mathsf{Ver}_{\mathrm{PF}}(\{\mathbf{b}, \mathbf{D_b}\})$
2: Regenerate $\mathbf{e} \in \{0,1\}^m$ by using PF, the Physical Function/POK (e.g., Ring Oscillators or SRAM)
3: Extract $\mathbf{s}$ by using argument $\mathbf{b}$ using Rep in Construction IV.1.
4: Verify that $\mathbf{D_b} = H(\mathbf{s}, \mathbf{b})$, else return $\perp$.
5: **return** $H(\mathbf{s})$
6: **end procedure**

Note that the above algorithm requires both internal randomness as well as a random oracle. This algorithm also includes an extension to check for valid challenge-response pairs. This is important, as Definition IV.4 allows for active adversaries.

Clearly, Gen runs in polynomial time. Ver also runs in polynomial time if there is no error. We will show in Sections V and VI that with our error correction step, Ver still runs in polynomial time with $\Theta(m)$ errors.

Our construction is illustrated in Figure 2. The *same* POK is used for all randomly generated secrets. If the POK outputs are noiseless it means that the same $\mathbf{e}$ is used for multiple secrets, whereas the standard LPN problem (cf. Section II-B) requires different $\mathbf{e}$. Note that if an adversary receives two sets of equations

$$\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$$
$$\mathbf{b}' = \mathbf{A}' \cdot \mathbf{s}' + \mathbf{e}$$

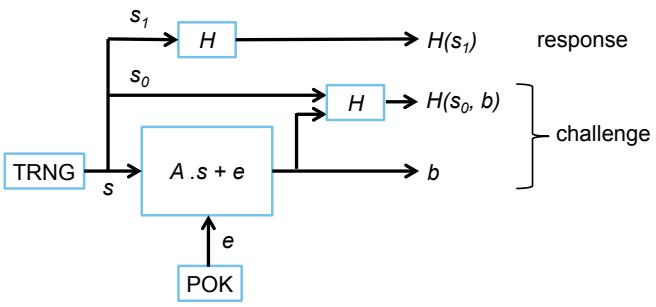
with $\mathbf{A} \neq \mathbf{A}'$, the adversary can discover $\mathbf{s}$ and $\mathbf{s}'$ by canceling out the $\mathbf{e}$ terms. However, we will show that if the $\mathbf{A}$ matrix is the same for the different secrets, discovering any individual secret requires the adversary to break standard LPN (cf. Lemma IV.10).

D. Security Proof

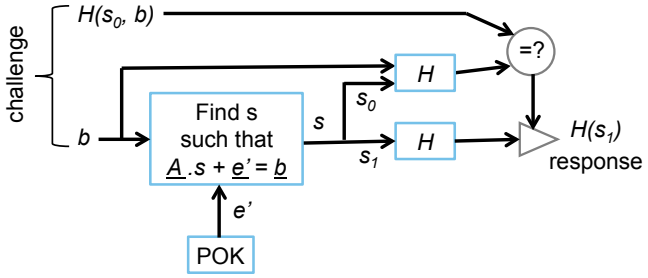
To show the reduction of the experiment in Definition IV.4, we first prove that inclusion of random oracle data and challenges do not impact the security of the system. To that end, consider the following algorithm that "simulates" the Gen algorithm from Construction IV.5.

Construction IV.6 (LPN Stateless Simulator 1)**.** *Let k be a security parameter, with $n = \mathsf{poly}(k)$, and $m \geq n$. Define $\mathbf{A} \in \{0,1\}^{m \times n}$ be a* **constant***, publicly known, uniformly random matrix row-indexed by i from 1 to m.*

Let PF *be the POK to which* Gen' *has oracle access.*



(a) Gen: Generation of challenge-response pairs. TRNG stands for True Random Number Generator.



(b) Ver: Regeneration of response when the PUF is presented with a valid challenge. The underlines on $\mathbf{A}$, $\mathbf{e}'$ and $\mathbf{b}$ indicate that a subset of n of the m rows are selected to solve for $\mathbf{s} = \{\mathbf{s}_0, \mathbf{s}_1\}$.

Fig. 2: Stateless PUF construction. Note that Gen and Ver can be called any number of times in any order. The PUF does not retain any state across invocations. In an implementation, H can be replaced with SHA-256.

1: **procedure** $\{\{\mathbf{b}, U_1)\}, U_2\} \leftarrow \mathsf{Gen}'_{\mathrm{PF}}(1^k)$
2: Run $\{\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}, H(\mathbf{s})\} \leftarrow \mathsf{Gen}_{\mathrm{PF}}(1^k)$
3: Uniformly generate $U_1,\ U_2 \in \{0,1\}^l$ (same length as output of $H(\cdot)$).
4: Construct res $= \{\{\mathbf{b}, U_1\}, U_2\}$.
5: Store res into a local table T.
6: **return** res.
7: **end procedure**

First, we show that an adversary A capable of achieving a non-negligible advantage $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A})$, can guess $\mathbf{s}$ with non-negligible advantage.

Lemma IV.7. *Given an algorithm* A *that has advantage* $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(A) = 1\right]$ *greater than* $\mathsf{neg}(k)$*, there exists an algorithm* B *that given polynomial queries to* $\mathsf{Gen}_{\mathrm{PF}}$*,* $\mathsf{Ver}_{\mathrm{PF}}$*, and $H(\cdot)$, can recover $\mathbf{s}$ with non-negligible probability.*

Proof. Construct algorithm B that runs $\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(A)$, but intercepts calls by A to $H(\cdot)$, storing all input/output pairs $\{x, H(x)\}$ into a table T.

When A succeeds with non-negligible probability in k, it returns a fresh $\{\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}, H(\mathbf{s})\}$, use table T to look up $H(\mathbf{s})$.

Because $H(\cdot)$ is a random oracle, $H(\mathbf{s})$ is guaranteed to be present if A succeeds, because A has non-negligible advantage in returning a valid $H(\mathbf{s})$.

Therefore, since $H(\mathbf{s})$ is present in the table, the other

element of the pair must be $\mathbf{s}$, which is then returned by B. □

Next, we show that the random oracle data $H(\mathbf{s}, \mathbf{b})$, $H(\mathbf{s})$ contained in the return value of Gen does not aid an adversary in breaking $\mathbf{s}$.

Lemma IV.8. *Given an algorithm A that, given oracle access to $\mathsf{Gen}_{\mathrm{PF}}$, $H(\cdot)$, recovers $\mathbf{s}$ with non-negligible probability in k, the algorithm A can also recover $\mathbf{s}$ given oracle access to ${\mathsf{Gen}'}_{\mathrm{PF}}$, $H(\cdot)$, with Gen' defined in Construction IV.6.*

Proof. The algorithm Gen generates $\mathbf{s} \in \{0,1\}^n$ (with $n = \mathsf{poly}(k)$) uniformly at random. Therefore, for polynomial queries to Gen, the probability that the same $\mathbf{s}$ is generated twice is $O(2^{-n}) = \mathsf{neg}(k)$.

Therefore, with probability $1 - \mathsf{neg}(k)$, all generated $\mathbf{s}$ are unique, implying that $H(\mathbf{s}, \mathbf{b})$, $H(\mathbf{s})$ are uniform random, since $H(\cdot)$ is a random oracle.

Therefore, the output of Gen is computationally indistinguishable from the output of Gen' with probability better than $1 - \mathsf{neg}(k)$.

Therefore, any algorithm that recovers $\mathbf{s}$ from polynomial queries to Gen with non-negligible probability in k will also recover $\mathbf{s}$ from polynomial queries to Gen' with non-negligible probability in k. □

Next, we show that access to Ver does not help an adversary, because the challenge is authenticated by the random oracle first.

Lemma IV.9. *Given a security parameter k, an algorithm A that makes polynomial oracle queries to $\mathbf{PUF} = \{\mathsf{Gen}_{\mathrm{PF}}, \mathsf{Ver}_{\mathrm{PF}}\}$, and $H(\cdot)$ (from Definition IV.3) and recovers $\mathbf{s}$ with non-negligible probability in k. One can construct algorithm B that only requires polynomial queries to $H(\cdot)$, and ${\mathsf{Gen}'}_{\mathrm{PF}}$ (Construction IV.6) and recovers $\mathbf{s}$ with non-negligible probability in k.*

Proof. Consider algorithm B that runs A, and emulates oracle queries by A as follows:

- $H(\cdot)$: Runs $H(\cdot)$ without modification.
- $\mathsf{Gen}_{\mathrm{PF}}(1^k)$: Runs $\{\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}, H(\mathbf{s})\} \leftarrow \mathsf{Gen}_{\mathrm{PF}}(1^k)$ from Definition IV.3. Records $\{\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}, H(\mathbf{s})\}$ into internal table T before returning.
- $\mathsf{Ver}_{\mathrm{PF}}(\mathbf{b}, H(\mathbf{s}, \mathbf{b})$: Looks up $\{\mathbf{b}, H(\mathbf{s}, \mathbf{b})\}$ in T. Returns $\perp$ if not found, and associated entry $H(\mathbf{s})$ otherwise.

If the oracle access to Ver helps A reveal $\mathbf{s}$, then it must be the case that A calls Ver when A does not yet have a non-negligible advantage in guessing $\mathbf{s}$ (otherwise, A does not require access to Ver to estimate $\mathbf{s}$ with non-negligible advantage).

In this case, the simulated Ver algorithm described above will return the correct result with probability $1 - \mathsf{neg}(k)$. However, A must only make a polynomial number of calls to Ver, so the above algorithm simulates the actual Ver from Definition IV.3 perfectly with probability $1 - \mathsf{neg}(k)$.

Therefore, if A extracted $\mathbf{s}$ with non-negligible probability, then B will extract $\mathbf{s}$ with non-negligible probability given only oracle access to $\mathsf{Gen}_{\mathrm{PF}}$, $H(\cdot)$.

Finally, by Lemma IV.8, B can also extract $\mathbf{s}$ with non-negligible probability given oracle access to ${\mathsf{Gen}'}_{\mathrm{PF}}$, $H(\cdot)$. □

Next, we prove the overall security of Construction IV.5.

Lemma IV.10. *Let k be a security parameter, some $n = \mathsf{poly}(k)$, and $m \geq n$.*

There is no PPT A that has advantage $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A})$ non negligible in k.

Proof. Assume that A has non-negligible advantage over Construction IV.5 in the experiment in Definition IV.4.

By Lemma IV.7, there exists an algorithm A' that reveals $\mathbf{s}$ with non-negligible probability given polynomial queries to $\mathsf{Gen}_{\mathrm{PF}}$, $\mathsf{Ver}_{\mathrm{PF}}$.

Furthermore, by Lemma IV.9, there exists an algorithm B that obtains this same non-negligible advantage in revealing $\mathbf{s}$ given only oracle access to ${\mathsf{Gen}'}_{\mathrm{PF}}$ (Construction IV.6).

Using B, construct algorithm C that violates the hardness Conjecture II.1 as follows.

Algorithm C takes as input a random LPN problem $(\mathbf{b}, \mathbf{A})$, where $\mathbf{A} \in \{0,1\}^{m \times n}$, $\mathbf{b} \in \{0,1\}^m$, and:

$$\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}_i$$

$\mathbf{s} \in \{0,1\}^n$ is uniformly random, and $\mathbf{e}$ is chosen according to distribution χ.

Algorithm C then simulates the experiment in Definition IV.4. Note that B only makes calls to ${\mathsf{Gen}'}_{\mathrm{PF}}$ (it doesn't call $\mathsf{Ver}_{\mathrm{PF}}$), so C must only simulate ${\mathsf{Gen}'}_{\mathrm{PF}}$ (Construction IV.6).

To simulate Gen', C performs the following:

1) Choose a uniformly random $\mathbf{s}_{\mathrm{DIFF}} \in \{0,1\}^n$
2) Construct:

$$\mathbf{b'}_i = \mathbf{b}_i + \mathbf{A}_i \cdot \mathbf{s}_{\mathrm{DIFF}} \tag{2}$$

$$= \mathbf{A}_i \cdot (\mathbf{s} + \mathbf{s}_{\mathrm{DIFF}}) + \mathbf{e}_i \tag{3}$$

3) Choose uniformly random $U_1,\ U_2 \in \{0,1\}^l$ (same length as output by $H(\cdot)$).
4) Return $\mathrm{res} = \{\{\mathbf{b}', U_1\}, U_2\}$.

Note first that $U_1,\ U_2$ correctly mimic the behavior of Gen'.

Furthermore, the $\mathbf{b'}_i$ terms created in Gen are correctly generated and correspond to the LPN problem with $\mathbf{s} + \mathbf{s}_{\mathrm{DIFF}}$ as the secret.

Since B reveals $\mathbf{s}$ with non-negligible advantage given oracle access to Gen', C gains non-negligible advantage in extracting $\mathbf{s}$ from a randomly chosen LPN problem with the same parameters.

This contradicts Conjecture II.1. □

Now that the security reduction is in place, we may state the main result:

Theorem IV.11. *Let k be a security parameter, and $n = \mathsf{poly}(k)$. There exists a choice of n, $m \geq n$, $l \geq k$, χ such that Construction IV.5 is a (m, l, χ) stateless PUF that is ϵ-secure with error δ, with $\epsilon = \mathsf{neg}(k)$, $\delta = \mathsf{neg}(k)$.*

Proof. First, recognize that Construction IV.5 is efficient. Second, by Lemma IV.10 there does not exist any PPT A that gains has advantage $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) > \epsilon$, where $\epsilon = \mathsf{neg}(k)$. □

Note that Theorem IV.11 holds if χ represents an i.i.d. distribution of m bits with bias $\tau \geq 1/\mathsf{poly}(k)$. This is extended in Lemma VII.2 to also support more complex distributions (e.g., with correlated bits).

Also note that Theorem IV.11 holds for any Decode algorithm of the form in Construction IV.1. In the current reductions, we assume that Decode only performs Gaussian elimination, which is only correct if there are no errors in

regenerating $\mathbf{e}$. Fuller et al. extend this methodology slightly by enumeratively searching for error-free combinations of $\mathbf{e}_i$, which is only efficient for $t = c \log n$ errors [21]. Section V introduces a construction that allows this result to be extended to the case where there are $t = \Theta(m)$ errors. This extension is consistent with Construction IV.5 (the adversary does not receive any new information), so Theorem IV.11 still holds.

E. Discussion

1) Blocking Malicious Challenges: If we did not include $H(\mathbf{s}, \mathbf{b})$ as part of the challenge when a response is queried, one can imagine an attack where the adversary applies a series of challenges that are related in some manner and attempts to determine if key recovery fails or succeeds for each challenge. Key recovery will succeed if the $\mathbf{b}_i$'s that are modified are not used to recover the secret key. This may give the adversary insight into which POK bits are stable and which ones are not, which in turn may indicate bit bias.

We prevent this attack by returning $H(\mathbf{s}, \mathbf{b})$ in the challenge-response generation in Construction IV.5, and ensure that a correct $H(\mathbf{s}, \mathbf{b})$ is received along with $\mathbf{b}$ if the PUF is to return a response. If $\mathbf{s}$ is recovered correctly (e.g., because the subsystem of equations chosen by Decode was not changed by the adversary or only changed slightly), any modification of $\mathbf{b}$ will be detected with overwhelming probability. If a different $\mathbf{s}'$ is recovered (e.g., because the adversary changed one of the equations in the subsystem of equations used to recover the secret), $H(\mathbf{s}', \mathbf{b}')$ will also not match with overwhelming probability.

2) Hash Function Requirements: $H(\cdot)$ is a random oracle that is well-approximated by the SHA-256 or SHA-3 hash functions. We require the one-way property for the hash H', since we are exposing $H'(\mathbf{s})$. Also, given two $\mathbf{b_1}$, $\mathbf{b_2}$ vectors, an adversary can calculate $\mathbf{s_1} + \mathbf{s_2}$. This does not help the adversary in discovering $\mathbf{s_1}$ or $\mathbf{s_2}$. As Lemma IV.10 shows, discovering the individual secrets requires the solution of LPN for a constant $\mathbf{A}$ matrix. However, to ensure that an adversary cannot impersonate a PUF, we require non-malleability of H'. That is, the adversary should not be able to generate $H'(\mathbf{s_1} + \mathbf{s}_{\text{DIFF}})$ given $H'(\mathbf{s_1})$ and $\mathbf{s}_{\text{DIFF}}$. These properties are obviously also required because of the use of $H(\mathbf{s}, \mathbf{b})$ in the construction.

Finally, we note that in Figure 2 we show the secret $\mathbf{s}$ split into two parts with the two parts being used in the challenge and response. We do this since $\mathbf{s}$ in our experiments of Section VIII is over 512 bits.

3) Controlled PUF: We have described a "vanilla" scheme for authentication where responses are returned in the clear when challenges are applied. However, all the controlled PUF (CPUF) protocols of [24] with small modifications are enabled by our construction once it has been made noise-free (cf. Section V). Briefly, the verifier obtains a *single* challenge-response pair securely, i.e., no eavesdroppers, as before. When the PUF receives a challenge, it does not return the response, but merely generates it internally and bit-exactly. Now, the verifier who knows the response, can use it as a shared secret for repeated nonce-based authentication or secure communication. Other verifiers can use completely different shared secrets.

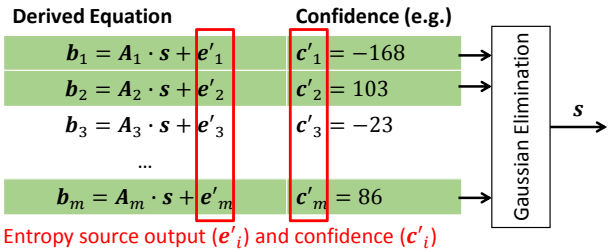


Fig. 3: Overview of LPN key extraction algorithm. The $\mathbf{e}'_i$ values are regenerated and the $\mathbf{c}'_i$ values with high absolute value identify the $\mathbf{e}'_i$ with low probability of error (since $\mathbf{c}'_i$ values don't change dramatically between measurements, and $\mathbf{e}'_i = \mathsf{Sign}(\mathbf{c}'_i)$). Gaussian elimination is then used on these selected equations to extract the secret key.

V. Fuzzy Extractor Using LPN

We will construct a computationally secure extractor from the LPN problem. Before presenting the extractor formally, we present an intuitive description.

A. Intuitive Description

Recall the description of the LPN problem in Section II-B, which is also depicted in Figure 3. A key intuition behind the construction is that one can use biometric data as the $\mathbf{e}_i$ values. Therefore, an adversary learns the equations with probability of error being $\mathsf{Pr}(\mathbf{e}_i = 1) = \tau$, where τ relates to the entropy of the biometric data. Having access to the biometric data allows one to regenerate $\mathbf{e}'_i$, where $\mathsf{Pr}(\mathbf{e}'_i = 1) = \tau$ and $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) = \tau' \ll \tau$. The regeneration is imperfect due to intrinsic noise of the biometric data as well as environmental changes. However, the LPN problem remains hard even for small τ' implying that key recovery will run in exponential time for $\Theta(m)$ number of errors.

The key intuition is that access to confidence information during the *regeneration* of the biometric bits allows the extractor to selectively choose which bits are stable. Initially, this may sound similar to using the "mask" data for iris or silicon biometrics. However, this approach is fundamentally different and has far superior security properties, as we recognize the biometric source itself to be a hidden trapdoor to a hard problem. This reduces the exponential complexity of error correction to polynomial.

A critical enabling property of LPN/LWE that has not been previously exploited is that if one can identify at least n different biometric bits ($\mathbf{e}_i$ values) that are correct with high probability, then one can use Gaussian elimination to solve for $\mathbf{s}$. *Any* set of n bits can be used. Therefore, each time one measures the biometric bits, one generates fresh confidence information from the measurement. This confidence information is used to identify the most stable bits (which may be different from measurement to measurement), and Gaussian elimination is performed on the subset of equations corresponding to these stable bits. The algorithm then *discards* the confidence information. Of course, one must architect the system such that there are enough stable biometric bits during each measurement with high probability. To this end, the LPN (or LWE) algorithm allows *arbitrary redundancy* in the number of equations supplied. Therefore, we can supply

enough equations such that with high probability (see Section VI) the recovery succeeds.

The security proof for this new construction is identical to that of the original LPN fuzzy extractor, Construction IV.1, since the adversary receives identical information.

B. Detailed Construction

In the following description, we will refer to the bits of the biometric data (measured at PUF challenge-response pair generation) as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \dots \mathbf{e}_m\}$, ($\mathbf{e}_i \in \{0,1\}$) and their noisy counterparts (measured during response verification) as $\mathbf{e}' = \{\mathbf{e}'_1, \mathbf{e}'_2, \dots \mathbf{e}'_m\}$, (once again, $\mathbf{e}'_i \in \{0,1\}$). Moreover, the confidence information associated with these noisy measurements (where applicable) will be denoted as $\mathbf{c}' = \{\mathbf{c}'_1, \mathbf{c}'_2 \dots \mathbf{c}'_m\}$, where $\mathbf{c}'_i \in \mathbb{Z}$ (as shown in Figure 3).

We describe the algorithms associated with the key extraction below. Typically, a fuzzy extractor, information-theoretic or computational, has the functions Gen and Extract, where Gen produces the public helper information, and Extract takes the biometric data and public helper information and returns the error-corrected key. This paper expands this construction to include four functions in the extraction process:

$\mathsf{Fab}(1^k, \epsilon_1, \epsilon_2, \eta)$: Represents the fabrication step. It takes the security parameter k, the desired probabilities of failure ϵ_1, ϵ_2 (see Section VI for further description), and performs the following:

1) Set $n = f(k, \eta)$. The η term is defined in Definition VII.1 and relates to the correlation present in the biometric source. n will be the size of the secret vector, and n will have to increase as the biometric bits become more biased (η goes to 0 or 1). Section VII will elaborate on the computation of η and n.
2) Compute m such that with probability greater than $1-\epsilon_1$, at least $m' = n^2$ of the m bits of biometric data are "stable" over relevant noise/environmental parameters. This choice of m' will be justified in Section VI. We define "stable" to be $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) \leq \epsilon_2$ (details also in Section VI).[3]
3) Generate m unique uniformly random $\mathbf{A}_i \in \{0,1\}^n$. (This is the same $\mathbf{A}$ in the stateless PUF construction of Section IV).
4) Manufacture devices that each produce m biometric bits internally.

$\mathsf{Gen}(n) \rightarrow (\mathbf{A}, \mathbf{b})$: Represents the challenge-response pair generation of the device manufactured in Fab. Gen takes the size of the secret vector n (calculated in Fab) and returns different public helper data $\mathbf{b}$, each time it is invoked.

1) Measure m bits of biometric data as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \dots \mathbf{e}_m\}$.
2) Generate a uniformly random secret vector $\mathbf{s} \in \{0,1\}^n$. (We differ from a conventional fuzzy extractor in that we are using a fuzzy commitment scheme [35]).
3) Compute $\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}_i$ for i between 1 and m.
4) Publish $\mathbf{b}$.
5) Discard $\mathbf{s}$ and $\mathbf{e}$.

We have not described challenge verification in Gen and outputting of the response $H(\mathbf{s})$ as is required in the stateless PUF construction of Section IV to avoid repetition.

$\mathsf{Project}(\mathbf{c}', \epsilon_2) \rightarrow S$: Represents the algorithm that determines the "stable bits" of biometric data to be used in Recovery.

1) Use measured confidence information $\mathbf{c}'_i = \mathbf{c}'_{i,\mathrm{meas}}$ to find $m' = n^2$ stable bits of the biometric data.[4] For each of the m' stable bits, $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i | \mathbf{c}'_i = \mathbf{c}'_{i,\mathrm{meas}}) < \epsilon_2$ for some small ϵ_2. (See Section VI for how this is done).
2) Let S be the set of these stable bits. Return S.

$\mathsf{Recovery}(\mathbf{e}', S) \rightarrow \mathbf{s}$: Represents the augmented key recovery algorithm. In addition to the noisy biometric data $\mathbf{e}'$, this function also takes S, the polynomially-sized set of stable bits in $\mathbf{e}'$. The steps of the algorithm are as follows:

1) Divide these m' bits into m'/n groups of n bits each.
2) Use Gaussian elimination on each group of n equations to solve for $\mathbf{s}$.
3) We now have m'/n different estimates for $\mathbf{s}$. Return the $\mathbf{s}$ that occurs most frequently.

Before looking into the effects of errors on the above algorithm, there are several notes to be made.

First, the Fab algorithm can be viewed as system design steps, while the Gen algorithm will be executed multiple times for each instantiation of the design, each time a new challenge-response pair is generated. The matrix $\mathbf{A}$ are public system parameters, and only $\mathbf{b}$ is the per-instance helper data that become the challenges of our stateless PUF. Project and Recovery together correspond to Extract in a fuzzy extractor, and are run each time a response (key) needs to be regenerated (recovered) given a challenge (helper data).

Second, if the LPN problem is hard, then an adversary in possession of $(\mathbf{A}, \mathbf{b})$ cannot compute $\mathbf{s}$ as shown in Theorem IV.2; the adversary obtains no new information. Furthermore, due to the simultaneous hardcore bits of $\mathbf{s}$ in the LPN problem, $\mathbf{s}$ is pseudorandom. This is a corollary of Theorem IV.2, and is explicitly proven in [21].

Third, the confidence information $\mathbf{c}'_i$ acts as a trapdoor for identifying "stable" bits in key recovery. Therefore, the key recovery algorithm is faced with a much lower error rate and hence a much easier problem than an adversary without the trapdoor, and can finish in polynomial time as we will show in Section VI.

VI. Noise-Avoiding Trapdoors

In Section V, the Fab and Project functions in our construction require knowledge of the error rate of bits of biometric data. This section will explore how these functions are implemented in the case where the biometric data is provided by ring oscillator POKs.

Fab uses a priori knowledge of the statistical distribution of the bits generated by the biometric source. The parameters given to Fab are public information. Project leverages confidence information that a bit is regenerated correctly ($\mathbf{e}'_i = \mathbf{e}_i$). Confidence data are extracted upon measurement of a biometric bit, and are never persistently stored.

$\mathbf{c}_i$ are random variables representing the confidence information of the i'th biometric bit (e.g., ring oscillator pair). They have the same probability distribution, shown in Figure 4. Define the output biometric bit to be a random variable $\mathbf{e}_i = \mathsf{Sign}(\mathbf{c}_i)$. Define $\mathbf{c}_{i,\mathrm{meas}}$ to be the result of measuring $\mathbf{c}_i$

[3] $m' = n^2$ is useful for showing polynomial asymptotic behavior. Improvement to $m' = n$ is shown in Sections VI, VIII.

[4] See footnote 3.

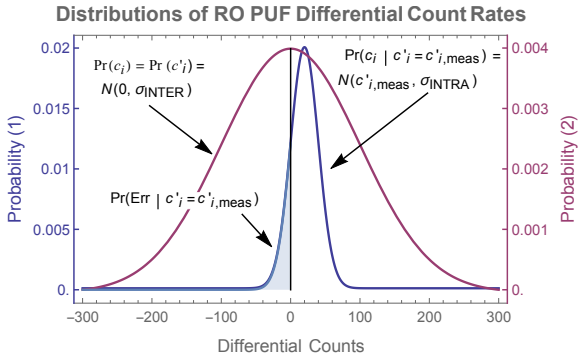


Fig. 4: Distribution of confidence information for different bits from a biometric source and with the same bit measured repeatedly over time/environmental parameters. The magenta curve corresponds to the distribution of confidence information across different devices. The blue curve corresponds to the distribution of measured confidence information from the same device in different conditions. We show the probability of error given a confidence measurement $\mathbf{c'}_{i,\text{meas}}$ as the integral of the shaded region.

at the time of initial challenge-response generation (a unique $\mathbf{c}_{i,\text{meas}}$ for each $\mathbf{c}_i$).

Next, $\mathbf{c'}_i$ are random variables representing the confidence information of the i'th biometric bit at some point in time later. They have the same probability distribution as $\mathbf{c}_i$. Define the output biometric bit to be a random variable $\mathbf{e'}_i = \mathsf{Sign}(\mathbf{c'}_i)$. Define $\mathbf{c'}_{i,\text{meas}}$ to be the result of measuring $\mathbf{c'}_i$ during key recovery. We note that if the noise is low for a particular bit, $\mathsf{Pr}(\mathbf{e'}_i = \mathbf{e}_i) \approx 1$. This is formalized below.

To provide concrete analysis, we consider the probability distribution of the $\mathbf{c}_i$ and $\mathbf{c'}_i$ random variables. Among different bits generated by the biometric source (i.e., the distribution of $\mathbf{c}_i$ with no prior information), this distribution can be taken to be a zero-mean Gaussian with variance σ_{INTER}, shown in Figure 4. Note that this directly implies that $\mathsf{Pr}(\mathbf{e}_i = 1) = \tau = 1/2$ for the LPN problem. In actual physical systems, there will be a bias towards 1 or 0, but we will see that assuming a 50% bias represents a "worst-case" *from the standpoint of error correction*. (Note that we will use a different worst-case bias for other purposes, e.g., to determine n given the security parameter in Section VII.)

Now, given that $\mathbf{c}_i$ and $\mathbf{c'}_i$ represent the random variables for measuring the *same* bit at different times and environmental parameters, the conditional distribution $\mathsf{Pr}(\mathbf{c}_i|\mathbf{c'}_i = \mathbf{c'}_{i,\text{meas}})$ is much narrower (where $\mathbf{c'}_{i,\text{meas}}$ is the actual measured value of $\mathbf{c'}_i$ at regeneration). This distribution is defined to be a Gaussian distribution with mean $\mathbf{c'}_{i,\text{meas}}$ and variance σ_{INTRA}. Both distributions are shown in Figure 4.

Note that since $\mathbf{c}_i$ and $\mathbf{c'}_i$ are random variables representing the same component of the biometric source, they have the same distribution (with no prior knowledge). Therefore, for any c, one can recognize that $\mathsf{Pr}(\mathbf{c}_i|\mathbf{c'}_i = c) = \mathsf{Pr}(\mathbf{c'}_i|\mathbf{c}_i = c)$.

In other words, one can use the confidence information collected during the fabrication step to reason about the probability of error at key extraction, or vice-versa. We may now define the probability of error given confidence information. Since $\mathbf{e}_i = \mathsf{Sign}(\mathbf{c}_i)$, we recognize that the error probability given a measurement ($\mathbf{c'}_{i,\text{meas}}$) of the confidence information is the integral of the shaded region in Figure 4. In particular, the CDF up to 0:

$$\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i|\mathbf{c'}_i = \mathbf{c'}_{i,\text{meas}}) = \frac{1}{2}\left(1 + \mathsf{Erf}(-|\mathbf{c'}_{i,\text{meas}}|/(\sqrt{2}\sigma_{\text{INTRA}}))\right) \quad (4)$$

First note that the unconditional probability of error $\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i)$ is the integral of Equation 4 over the distribution of $\mathbf{c'}_i$. This integral depends *only* on σ_{INTRA} and σ_{INTER} (not on n, m, etc.). Therefore the error rate for m bits is clearly $\Theta(m)$.

Once again, due to the symmetry of the distributions of $\mathbf{c}_i$ and $\mathbf{c'}_i$, we know that for any value of c, $\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i|\mathbf{c'}_i = c) = \mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i|\mathbf{c}_i = c)$. With this in mind, we may now describe how Fab and Project work.

A. Fabrication/Provisioning

In addition to the bound on bias, the Fab algorithm is given a security parameter k, and two error rates $\epsilon_1, \epsilon_2 > 0$ that are defined as follows. Fab must compute m such that with probability greater than $1 - \epsilon_1$, at least m' of the m bits will be stable. Recall a random bit $\mathbf{e}_i$ is defined to be stable if $\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i) < \epsilon_2$ over relevant environmental parameters.

To do this, we recognize that requiring $\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i|\mathbf{c}_i = \mathbf{c}_{i,\text{meas}}) < \epsilon_2$ sets a threshold c_T. If a ring oscillator has $|\mathbf{c}_{i,\text{meas}}| > c_T$, then the probability of an error in this bit (when measured later) is less than ϵ_2 (the bit is stable). We plug these requirements into Equation 4 and solve for c_T: (define Erf^{-1} as the inverse of Erf)

$$c_T = \sqrt{2}\sigma_{\text{INTRA}}\mathsf{Erf}^{-1}(1 - 2\epsilon_2) \quad (5)$$

Therefore, the probability that a given bit is stable (has a $\mathbf{c}_{i,\text{meas}}$ value above the threshold) can be computed by integrating the PDF of $\mathsf{Pr}(\mathbf{c}_i)$ shown in Figure 4 in the region $|\mathbf{c}_i| > c_T$. This is equal to:

$$P_{\text{ST}} = \mathsf{Pr}(|\mathbf{c}_i| > c_T) > 1 - \mathsf{Erf}\left(c_T/(\sqrt{2}\sigma_{\text{INTER}})\right)$$

The inequality is because the probability of a bit being stable is *smallest* when the bit bias is 50% (the Gaussian is centered at 0). One can see that as the center of the Gaussian shifts, more probability density falls in the region of $|\mathbf{c}_i| > c_T$. Therefore, we know that the probability of a stable bit can only be higher than our calculation here expects. Plugging in c_T from Equation 5 gives the completed formula:

$$P_{\text{ST}} > 1 - \mathsf{Erf}\left(\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}\mathsf{Erf}^{-1}(1 - 2\epsilon_2)\right) \quad (6)$$

The final step is to compute m such that at least m' POK bits will be stable ($|\mathbf{c'}_{i,\text{meas}}| > c_T$), with probability $1 - \epsilon_1$. This is a binomial distribution and is subject to a Chernoff bound. Define X as the random variable for the number of stable bits observed.

$$\mathsf{Pr}(X \leq m') \leq \exp\left(-\frac{1}{2}\left(1 - \frac{m'}{mP_{\text{ST}}}\right)^2 mP_{\text{ST}}\right) \leq \epsilon_1$$

Using this bound and our requirement of ϵ_1, we find that to ensure the probability that m' of the m POK bits are stable

with probability $1-\epsilon_1$, we need the following relationship between m' and m:

$$m \geq \frac{1}{P_{\mathrm{ST}}}\left(m' - \log(\epsilon_1) + \sqrt{\log(\epsilon_1)\left(\log(\epsilon_1) - 2m'\right)}\right) \quad (7)$$

Since P_{ST} is a function of ϵ_2, given m', ϵ_1, and ϵ_2, one can compute m such that at least m' of the POK bits are stable with probability $1-\epsilon_1$, as is required in the Fab algorithm.

B. Projection/Extraction

The extension of the above analysis to the Project algorithm is comparatively simple. Given an ϵ_2, we just use Equation 5 to compute a threshold c_T value, and select m' bits that have measurements $|\mathbf{c'}_{i,\mathrm{meas}}| > c_T$, which guarantees $\mathsf{Pr}(\mathbf{e'}_i \neq \mathbf{e}_i | \mathbf{c'}_i = \mathbf{c'}_{i,\mathrm{meas}}) < \epsilon_2$. Because of the Fab algorithm, we can be confident that we will find at least m' stable bits with high probability. These bits are then used to extract the secret key using Gaussian elimination and majority voting.

C. Showing the "Trapdoor"

In this section, we show that with this trapdoor, we only need $m \in \Theta(n^3)$ bits ($n \in \mathsf{Poly}(k)$), and our Project and Recovery algorithms finish in polynomial time with negligible failure probability.

At a high level, we want at least $1-\epsilon_1$ probability to have $m' = n^2$ stable bits, each stable bit with an error probability of at most ϵ_2. Then, we divide these n^2 stable bits into n groups, and solve for $\mathbf{s}$ using Gaussian elimination for each of the n groups. As long as there exist at least two groups that contain n error-free bits, we can solve $\mathbf{s}$ and recover all $\mathbf{e}_i$'s from one group and verify on the other group. (We note that performing Gaussian elimination on two groups with errors is very unlikely to yield the same solution because each group is associated with independently chosen $\mathbf{A}_i$'s). Since each bit has an error probability of at most ϵ_2, the probability that a group of n bits is not error-free is bounded by $n\epsilon_2$. Thus, the probability that all the n groups have errors is less than $\approx n(n\epsilon_2)^{n-1}(1-n\epsilon_2)$. Taking ϵ_1 into account, the failure probability of our construction is bounded by $\epsilon_1 + (1-\epsilon_1)n(n\epsilon_2)^{n-1}(1-n\epsilon_2)$. If we set ϵ_1 to be negligible (e.g., 2^{-n}) and $\epsilon_2 \in \Theta(1/n)$, the above failure probability is negligible in n.

We now show that to achieve $m' = n^2$, $\epsilon_1 = 2^{-n}$ and $\epsilon_2 \in \Theta(1/n)$, we only need $m = O(n^3)$. To see this, first recognize that P_{ST} (probability of a bit being stable across environment/temperature) depends *only* on $\sigma_{\mathrm{INTRA}}/\sigma_{\mathsf{INTER}}$ and ϵ_2. We consider $\sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}}$ to be a constant defined by the manufacturing process. We consider a *worst-case*, where $\sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}} = 1$. In this case, Equation 6 reduces to $P_{\mathrm{ST}} = 2\epsilon_2$. Plug them into Equation 7, and one obtains:

$$m = \frac{1}{2\epsilon_2}\left(n^2 + n + \sqrt{n(n+n^2)}\right) = \Theta\left(n^3\right) \in \mathsf{Poly}(n)$$

This shows that even in pessimistic scenarios, $m \in \Theta(n^3)$. In reality, the ratio $\sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}} < 1$, so the growth is much smaller, as will be seen in Section VI-D, VIII.

We remark again that without the confidence trapdoor, the LPN hardness states exactly that it is infeasible to compute the key in polynomial time with non-negligible success probability. Therefore, while an adversary requires exponential time to calculate the key, the owner of the fuzzy extractor requires only polynomial time. This is the definition of a trapdoor.

D. Improving on the Trapdoor

Note that the above analysis requires $m = O(n^3)$ biometric bits total. It will be seen in Section VII that for the LPN problem, typical n are on the order of 500. This translates to a bound of $> 10^8$ biometric bits, which is very large.

While the above analysis provides simple analytic insight, it is not a tight bound. The primary reason for this is that the ratio $\sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}}$ is much less than assumed. Therefore, ϵ_2 can be much less than $1/n$ without incurring blowup of m. This means that the number of errors in a set of n stable bits is one or less with high probability. One can then use a brute-force guess-and-check search to identify any erroneous bits using a simple scheme that we will describe later in the section.

In Section VIII, an implementation is described with $m \ll n^3$. This implementation leverages several tricks to achieve this performance. These tricks can be generalized for an arbitrary application as follows.

First, one must define several design parameters. Note that $\epsilon_1 + (1-\epsilon_1)P_{\mathrm{FAIL}}$ is the overall failure probability.

- LPN key size n which is based on the security parameter (see Section VII).
- A maximum extraction complexity t such that each extraction requires n^t Gaussian eliminations.
- The max failure rate of Project: ϵ_1 (see Section VI).
- The max failure rate of Recovery: P_{FAIL}.

One can compute m, the overall number of biometric bits required, by first computing ϵ_2 (cf. Section VI) so that n stable bits have at most t errors with probability $> 1 - P_{\mathrm{FAIL}}$:

$$P_{\mathrm{FAIL}} = 1 - \sum_{k=0}^{t} (\epsilon_2)^k (1-\epsilon_2)^{n-k} \quad (8)$$

Next, set $m' = n$ instead of n^2 (recall that m' is the number of stable bits in each measurement). This is possible because ϵ_2 is small. Finally, plug the result from Equation 8 into Equations 6, 7 to obtain m.

The extraction algorithm brute-force searches for at most t errors in the string of n stable bits through a guess-and-check approach. This requires n^t Gaussian eliminations (to compute the secret key $\mathbf{s}$) and n^t checks to see if $\mathbf{s}$ is correct.

An efficient way of checking key extraction is as follows. Since $m \gg n$, the computation of $\mathbf{s}$ by Gaussian elimination of the n most-confident bits can be simply checked by measuring the error rate of the remaining $m-n$ equations. An error rate of $\approx 50\%$ or more implies that the derived key is incorrect. A lower error rate (e.g., 25% or lower) indicates the key is correct. If $\mathbf{s}$ has been deemed to be generated correctly using this technique, the internally-computed $H(\mathbf{s_0}, \mathbf{b})$ is then checked against the corresponding part of the challenge to ensure that $\mathbf{b}$ has not been tampered with (cf. Figure 2).

Note that n^t is a tunable parameter based on extractor compute power. In Section VIII, $t = 1$ is chosen because the compute power is generally limited for PUF applications. Higher t values may be chosen where this is less of a factor.

VII. Security Analysis and Assumptions

A. Assumptions on POK Outputs

The POK outputs are used as the noise term in the LPN problem, and our PUF construction is secure if the POK outputs are i.i.d. We now provide a significantly relaxed definition

of POK source entropy under which our PUF construction remains secure. In particular, the following definition describes the class of sources that are secure with this construction.

Definition VII.1. *Define a set of L different m-bit entropy sources whose probability distribution may be constructed in the following way:*

1) *Begin with a set X of $m \times L$ bits that are i.i.d. with $\mathsf{Pr}(X_i = 1) = \eta$, $1 > \eta > 0$.*
2) *Select a set of affine linear transformations $F = \{F_0, F_1, \ldots, F_k\}$ (where $F(X) = M \cdot X + N$ for some $mL \times mL$ matrix M, and mL-dimensional vector N). Select a k-bit string f according to an arbitrary[5] distribution over $\{0,1\}^k$.*
3) *Return $F_k^{f_k}(F_{k-1}^{f_{k-1}}(\cdots F_1^{f_1}(F_0^{f_0}(X))\cdots))$, where $F_i^1 = F_i$ and F_i^0 is the identity transformation.*

This distribution is clearly much more general that an i.i.d. distribution, as it allows for certain bits from the same/different entropy sources to be correlated. For example, consider the i'th bit of LPN problem A, and the j'th bit of LPN problem B, this distribution can support a non-zero correlation coefficient between these bits, namely, $\mathsf{Corr}(\mathbf{e}_{A,i}, \mathbf{e}_{B,j})$. However, it is tighter than min-entropy, as min-entropy allows for individual bits to be "stuck" at one or zero. In this distribution, bits cannot be perfectly correlated (e.g., $\mathsf{Corr}(\mathbf{e}_{A,i}, \mathbf{e}_{B,j}) = 1$). We will see that η in effect sets the "maximum correlation", and η (as well as $1 - \eta$) must not be 0 or negligible in the security parameter.

Note also that knowledge of which bits are correlated is *public* (it is assumed that the adversary knows the transformations that are applied). Furthermore, note that the set of bits X is the set of bits *across* different sources. For this discussion, each source has m bits. If there are L different sources, then X is the set of all $L \times m$ bits. As a result, correlations between bits on different sources is allowable in the definition. Under this assumption, Lemma VII.2 proves the security of the system.

Lemma VII.2. *If the entropy sources for a collection of LPN fuzzy extractors have a joint distribution that can be described by Definition VII.1 for some η, then an algorithm that can extract $\mathbf{s}$ from any of the fuzzy extractors in polynomial time with non-negligible advantage can be used to solve the traditional LPN problem with bias η in polynomial time with non-negligible advantage.*

Lemma VII.2 can be proved by recognizing that a set of LPN problems with i.i.d. bits for their e_i values can be converted into a collection of LPN problems with bits described by Definition VII.1 by probabilistically applying the identified sequence of linear transformations F to their public keys $(\mathbf{A}, \mathbf{b})$. The proof is given:

Proof. Consider a collection of L different m-bit entropy sources. Let X be the set of all $m \times L$ bits, and let the joint distribution of X be described by Definition VII.1. Specifically, Definition VII.1 takes several parameters. Let the initial bias be η. Let $F = \{F_0, F_1, \ldots, F_k, \ldots\}$ be the set of affine transformations. Let $P = \{P_0, P_1, \ldots, P_k, \ldots\}$ be the set of random bits that determines which subset of F_i are applied. Let P have some joint distribution. The definition states that we can sample from this distribution in polynomial time.

[5]that can be sampled in polynomial time

$$
\begin{aligned}
&\text{Problem 1}: \begin{cases} \mathbf{b}_1^1 = \mathbf{A}_1^1 \cdot \mathbf{s}^1 + \mathbf{e}_1^1 \\ \mathbf{b}_2^1 = \mathbf{A}_2^1 \cdot \mathbf{s}^1 + \mathbf{e}_2^1 \\ \vdots \\ \mathbf{b}_m^1 = \mathbf{A}_m^1 \cdot \mathbf{s}^1 + \mathbf{e}_m^1 \end{cases} \\
&\text{Problem 2}: \begin{cases} \mathbf{b}_1^2 = \mathbf{A}_1^2 \cdot \mathbf{s}^2 + \mathbf{e}_1^2 \\ \mathbf{b}_2^2 = \mathbf{A}_2^2 \cdot \mathbf{s}^2 + \mathbf{e}_2^2 \\ \vdots \\ \mathbf{b}_m^2 = \mathbf{A}_m^2 \cdot \mathbf{s}^2 + \mathbf{e}_m^2 \end{cases} \\
&\quad\vdots \\
&L \text{ Problems}
\end{aligned}
\tag{9}
$$

Now, consider the set of L corresponding LPN problems (each using a distinct set of m bits from X as its e_i values). Let adversary $\mathcal{A}$ take as argument the public parameters of this set of LPN problems: $(\mathbf{A}_i^j, \mathbf{b}_i^j)$, for i from 1 to m (there are m equations in a single LPN problem), and j from 1 to L (the set of L LPN problems). Assume that there exist parameters η, F, and a distribution over P such that $\mathcal{A}$ calculates at least one of the secret keys of the set of LPN problems with non-negligible probability.

Using $\mathcal{A}$, we construct algorithm $\mathcal{B}$ that takes as argument the public parameters of L different LPN problems whose $\mathbf{e}_i$ bits are i.i.d. with bias η. $\mathcal{B}$ will return the secret vector of at least one of the LPN problems with non-negligible probability. Note that $\mathcal{B}$ is equivalent to breaking the LPN problem, as each LPN problem is independent.

First, consider a single LPN problem where $\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots \mathbf{A}_m\}$, $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_m\}$, $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_m\}$, and $\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}_i$. The bits $\mathbf{e}_i$ have some distribution. The key recognition is that the act of applying an affine transformation to the set of bits $\mathbf{e}$ is equivalent to applying the same transformation to $\mathbf{A}$ and $\mathbf{b}$. If one wants to transform the distribution by applying $F(e) = M \cdot e + N$ (M is an $m \times m$ dimensional matrix and N is an m-dimensional vector), then one can derive a different LPN problem:

$$
\begin{aligned}
F(\mathbf{b})_i =& F(\mathbf{A} \cdot \mathbf{s} + \mathbf{e})_i \\
(M \cdot \mathbf{b} + N)_i =& (M \cdot \mathbf{A})_i \cdot \mathbf{s} + F(\mathbf{e})_i
\end{aligned}
$$

By setting $\mathbf{b}' = M \cdot \mathbf{b} + N$ and $\mathbf{A}' = M \cdot \mathbf{A}$, we have a new LPN problem: $\mathbf{b}'_i = \mathbf{A}'_i \cdot \mathbf{s} + \mathbf{e}'_i$, where $\mathbf{e}'_i = F(\mathbf{e})_i$. By modifying *only the public parameters*, we have transformed the distribution of $\mathbf{e}_i$ by an affine transformation.

We generalize this to multiple LPN problems by recognizing that the above technique can be applied to the set of equations that comprise multiple LPN problems by simply concatenating the vectors, resulting in Equation 9.

Now, we recognize (where $|$ is concatenation) that to transform a set of problems with $\mathbf{e}^1|\mathbf{e}^2|\cdots\mathbf{e}^L$ into a set of problems with $F(\mathbf{e}^1|\mathbf{e}^2|\cdots\mathbf{e}^L)$, one can simply concatenate the aforementioned observation (note that M is now an $mL \times mL$ sized matrix, and N is a vector of dimension mL):

$$
\begin{aligned}
\mathbf{b}'^1|\mathbf{b}'^2|\cdots\mathbf{b}'^L =& M \cdot (\mathbf{b}^1|\mathbf{b}^2|\cdots\mathbf{b}^L) + N \\
\mathbf{A}'^1|\mathbf{A}'^2|\cdots\mathbf{A}'^L =& M \cdot (\mathbf{A}^1|\mathbf{A}^2|\cdots\mathbf{A}^L)
\end{aligned}
$$

We now return to the discussion of algorithm $\mathcal{B}$. The algorithm $\mathcal{B}$ is the probabilistic application of the above fact multiple times. The steps of $\mathcal{B}$ are as follows:

1) Sample p_i from the distribution of each P_i.
2) Set $\mathbf{b}_{\mathrm{TOT}} = \mathbf{b}^1|\mathbf{b}^2|\cdots\mathbf{b}^L$.
3) Set $\mathbf{A}_{\mathrm{TOT}} = \mathbf{A}^1|\mathbf{A}^2|\cdots\mathbf{A}^L$.
4) For j from 0 to k, Define $F_j(x) = M_j \cdot x + N_j$. If $p_j = 1$, set $\mathbf{b}_{\mathrm{TOT}} = M_j \cdot \mathbf{b}_{\mathrm{TOT}} + N$, and set $\mathbf{A}_{\mathrm{TOT}} = M \cdot \mathbf{A}_{\mathrm{TOT}}$. Otherwise do nothing.
5) Call $\mathcal{A}$ using the newly created public parameters for the set of LPN problems. Return the secret vector that $\mathcal{A}$ computes.

The final value of the public parameters corresponds to a set of LPN problems where the statistics of $\mathbf{e}_i^j$ are equal to those that can be solved by $\mathcal{A}$. Moreover, the set of secret vectors has not changed, and we obtained this problem by modifying bits of the public parameters *only*. Therefore, if $\mathcal{A}$ exists, it will recover at least one $\mathbf{s}$ with non-negligible advantage. Then, $\mathcal{B}$ can be used to break an i.i.d. LPN problem with bias η. This is a contradiction if LPN is hard, so $\mathcal{A}$ cannot exist. □

Note that the key step in the above algorithm is that $\mathcal{B}$ applies the affine transformation to the public parameters of the set of LPN problems. This operation produces a new set of LPN problems that are *statistically identical* to the case where the POK bits themselves have a transformed distribution. This allows $\mathcal{B}$ to transform a set of i.i.d. LPN problems into a set of LPN problems with correlated POK data by modifying only public parameters, and without affecting the secret vectors.

Also note that a corollary of Lemma VII.2 is that $\mathbf{s}$ remains pseudorandom even in the presence of correlated bits. This is due to the fact that LPN's secret has $n - o(n)$ simultaneous hardcore bits [1], and is proven for uncorrelated LWE in [21]. The proof is similar for the correlated LPN construction, as it is independent of the transformations performed in Lemma VII.2.

B. Security Parameter Derivation

The security goal is that an adversary given helper data must perform $\Omega(2^k)$ operations (security parameter $k = 128$) to discover the secret key. As we have shown in Theorem IV.11, discovering the secret key requires solving the LPN problem.

There is significant evidence to justify the LPN hardness conjecture [53], [10], [8]. Moreover, our construction can be easily extended to depend on the LWE problem, which has been reduced to the ISVP problem [53].

For $k = 128$, we describe the choice of LPN parameters with respect to best known attacks. However, POK bits must be distributed in a way that LPN is still hard. Definition VII.1 and Lemma VII.2 formalize this requirement.

In our construction, we derive the size n of the secret vector $\mathbf{s}$ as a function of the security parameter k. In typical LPN-based cryptosystems, the error rate $\tau = \Pr(\mathbf{e}_i = 1)$ must be kept low (e.g., $\tau = 0.0024$ [16]) to ensure correct decryption. We, on the other hand, do not use any LPN encryption/decryption algorithm, and therefore *do not have the same restriction on* τ.

In fact, we would like τ to be 0.5, representing perfect entropy in the POK data. Yet, to be conservative, we pessimistically assume $\tau = 0.45$. We set $\eta = 0.4$ (as discussed in Definition VII.1). As such, we use the analysis presented in [8] to discover an estimate for n based on k in order to obtain a security parameter of 128. We obtain for this set of parameters, $n \approx 540$.

Parm.	n	τ	a	b	l	W	q
Value	540	0.4	94	4	1	4	4

TABLE I: Table of parameters for [8] that minimize computation time for 10MB memory limit achieving a security parameter of 128.

Temp.	Bias
$-40°C$	54%
25°C	52%
105°C	53%

Temp.	σ_{INTRA}
$-40°$C	24.3 ± 1.3
0°C	8.9 ± 0.40
70°C	17.4 ± 0.64
85°C	24.0 ± 1.0
105°C	33.7 ± 1.4

TABLE II: (Left) Measured bias of 320 RO pairs at varying temperatures. (Right) Measured σ_{INTRA} for varying temperatures.

We follow the concrete attack approach put forth in [8], which combines ideas from canonical LPN attacks [10], [37], [42]. We restrict the memory of the attack to 2^{23}, as this corresponds to roughly 10MB of memory which must be accessed at a single-cycle latency. The algorithm in [8] requires parameters of a, b, l, W, and q. These are new parameters that are not related to any other variables in this paper. In particular, we optimize over $1 \leq a \leq 100$, $1 \leq b \leq 100$, $1 \leq l \leq 10$, and $1 \leq W \leq 4$, and $1 \leq q \leq 500$.

The cryptanalysis uses τ to represent the i.i.d. bit bias of the LPN problem. We set $\tau = \eta = 0.40$ (due to Lemma VII.2). We find that for $n = 540$, the expected number of bit operations is on the order of 2^{131} for parameters in Table I. Using the breakdown from [8], we recognize that an individual iteration is required of several steps: The extraction of the required additional queries requires $2^{23.7}$ bit operations. The filtering/clearing step requires roughly $2^{26.2}$ bit operations. The Walsh transform requires roughly $2^{25.7}$ bit operations. Finally, the dot product computation dominates with roughly $2^{33.6}$ bit operations. Therefore, a single iteration of the algorithm requires roughly $2^{33.7}$ bit operations. However, one iteration of the algorithm has probability $2^{-97.7}$ of success, and so it must be repeated an expected number of $2^{97.7}$ times. This yields an overall expected number of bit operations equal to $2^{131.3}$.

VIII. Case Study using a Ring Oscillator POK

We will use Ring Oscillator POKs as a case study because of the easy availability of confidence information (cf. Figure 1). In the case of the RO POK, the differential counts between the ring oscillators is the confidence information $\mathbf{c'}_i$, and the output bit $\mathbf{e'}_i = \mathsf{Sign}(\mathbf{c'}_i)$ described in Section II-A.

We have provided a theory explaining the resilience of the LPN construction to noise and environmental parameters using this confidence information in Section VI. Now, we use this theory and collected data from a set of 320 pairs of ring oscillators measured across temperature and voltage ranges to demonstrate the efficiency of the LPN fuzzy extractor construction in a concrete fashion. Experiments were conducted on a Xilinx Virtex 7 Series Field Programmable Gate Array (FPGA). We measured the differential counts of a set of 320 ring oscillator pairs in a wide (beyond industrial)

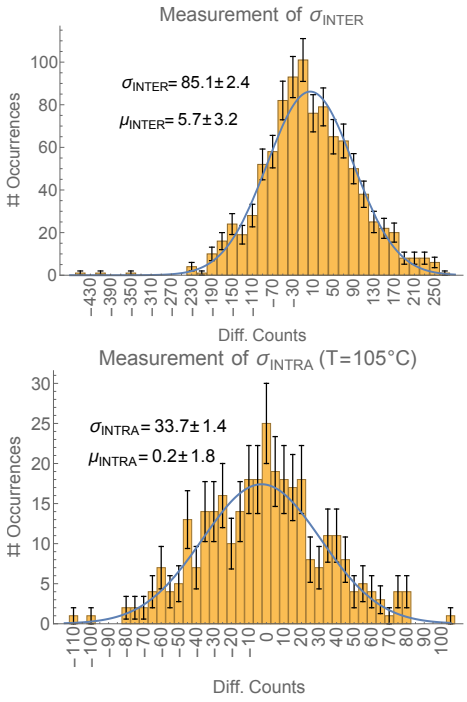


Fig. 5: (Top) Measurement of σ_{INTER} through the estimation of the distribution of differential counts across 320 RO pairs across room temperature and the fast and slow voltage/temperature corners. (Bottom) Measurement of σ_{INTRA} by subtracting differential counts at $25°\text{C}@1V$ from $105°\text{C}@1.05V$.

range of temperature and voltage. Three interesting points are $-40°\text{C}@0.95V$, $25°\text{C}@1.00V$, and $105°\text{C}@1.05V$. Other ranges that we will use are the differential count values at commercial ($0°\text{C}$ to $70°\text{C}$) and extended industrial ($-40°\text{C}$ to $85°\text{C}$). The $\sigma_{\text{INTRA}}/\sigma_{\text{INTER}}$ ratios improve as the temperature range is reduced.

We note that 24% of the ring oscillator pairs produce different responses in the environmental range; this is the typical $O(m)$ error case for such circuits under environmental stresses in the ranges shown.

We first measured the bias of the RO counts across temperature as shown in Table II. Therefore, our pessimistic estimate of bias ignoring correlation effects as 45% (or 55% equivalently) is correct.

These differential count values are distributed according to the distribution discussed in Section VI with variance σ^2_{INTER}. We verified for each of these temperatures that the distribution of differential counts was Gaussian, as we assumed in Section VI. Each of the fits from which we derived parameters have a χ^2 value corresponding to $> 95\%$ confidence. Moreover, neither the mean nor variance of the distribution changed significantly over temperature or voltage. Therefore, we describe the distribution in terms of a single mean, variance (μ_{INTER}, σ_{INTER}) shown in Figure 5. To measure μ_{INTRA} and σ_{INTRA}, one must measure the distribution of how these differential counts change regardless of the differential count measured at provisioning. This distribution is $\mathsf{Pr}(\mathbf{c'}_i - \mathbf{c}_i)$. We can calculate this distribution by using data from different ring oscillators. We then recognize that the standard deviation of this distribution is σ_{INTRA}.

Temp.	Erroneous Bits	$\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$	# Ring Osc. Pairs
$0°\text{C} - 70°\text{C}$	9%	0.20	1780
$-40°\text{C} - 85°\text{C}$	21%	0.29	3230
$-40°\text{C} - 105°\text{C}$	24%	0.40	8740

TABLE III: Summary of $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ and resources required for an LPN fuzzy extractor over the specified temperature range. The percentage of erroneous bits over environmental conditions and associated ratio is displayed. Key extraction succeeds with error probability $< 10^{-6}$ and a security parameter of 128.

To accomplish this, we used room temperature as a baseline (this would be the conditions in which the challenge-response pairs would be initially generated), and measured how the differential counts change as temperature/voltage vary for each of the 320 ring oscillator pairs. These data provide a statistical distribution of how much the differential count value will change with a change in environmental parameters (the distribution described by σ_{INTRA}, μ_{INTRA} in Section VI).

The distribution at $105°\text{C}$ is shown in Figure 5. The measurements at various temperatures are shown in Table II. It is important to note that although in Section VI we did not present any theoretical justification for the reason why the distribution of counts of a single ring oscillator pair over relevant environmental conditions would be Gaussian, this does turn out to be the case within experimental error as demonstrated in Figure 5.

We now present an analysis of the resource requirements (number of RO pairs) to construct a LPN fuzzy commitment scheme with a security parameter of 128, and probability of error 10^{-6} over the above temperature ranges. Note that LPN's secret has $n - o(n)$ simultaneous hardcore bits [1], therefore the key entropy is ≥ 512 bits.

First, using these measurements, we calculate the ratio ($\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$) required in Section VI for commercial ($0°\text{C}$ to $70°\text{C}$) as 0.20, extended industrial ($-40°\text{C}$ to $85°\text{C}$) as 0.29, and the maximum temperature range our experiment could support ($-40°\text{C}$ to $105°\text{C}$) as 0.40. This is shown in Table III.

We choose $\epsilon_1 = 10^{-6}$ and $P_{\text{FAIL}} = 10^{-6}$, and used Equation 8 to compute $\epsilon_2 \approx 3 \times 10^{-6}$. Using $n = 540$ (a security parameter of 128) and these values for ϵ_1 and $\sigma_{\text{INTRA}}/\sigma_{\text{INTER}}$ with Equations 6, 7 we may compute m (the number of RO pairs) to achieve $m' = 540$ stable bits. These values are are shown for various temperature ranges in Table III.

Note that in Section V, our algorithm description gives $m' = n^2$, as this achieves negligible probability of error. For practical purposes, this is far too conservative. In the above case, we choose $m' = n$. If we perform a single Gaussian elimination with the most stable bits, the system error probability would be $n \times 10^{-6}$. However, through a simple binomial distribution analysis, we recognize that the probability of 2 or more errors in the set of m' stable bits is $< 10^{-6}$. Therefore, one exhaustively searches over these possibilities (n Gaussian elimination operations given that $t = 1$ as described in Section VI).

This approach is possible because the ratio $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ for RO POKs ($\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}} \leq 0.4$) is significantly less than pessimistic assumption in Section VI ($\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}} = 1$). This smaller ratio allows us to set ϵ_2 to a much smaller value than $1/n$ (as is

used in Section VI), which in turn allows us to decrease m' while maintaining an overall probability of error of $< 10^{-6}$.

Note that our analysis is still pessimistic (e.g., assuming that all stable bits have error probability ϵ_2 even though most bits have much lower error probability) and our construction is unoptimized. Even with an unoptimized implementation, these results compare well, e.g., 1780 helper data bits for commercial range of operation, with the works described in Section III. Unlike most prior work on information theoretic extractors, our LPN fuzzy extractor can be scaled to higher noise settings simply by changing n and m without affecting the security argument. Most important, our construction allows for repeated generation and extraction of random keys (albeit with known relationships) enabling a stateless PUF construction.

IX. Conclusion

We have presented the first secure construction of a stateless Physical Unclonable Function in this paper. This has been an open problem for over thirteen years since silicon PUFs were introduced in 2002 [25].

Our construction is secure under the random oracle model and under well-established assumptions, namely, the difficulty of Learning Parity with Noise (LPN). Our construction is noise-free; the responses during challenge-response generation and successful verification match exactly. This means that an entity with a single challenge-response pair can authenticate the PUF any number of times by treating the response as a shared secret. All the protocols described in [24] are enabled by our construction.

In order to construct a noise-free PUF, we have to error correct the constituent POK responses. Fuzzy extractors have served as a useful construction to generate secret keys from noisy biometric sources. However, practical issues such as the level of noise, and the size of the helper data, have held back fully information-theoretically secure constructions. Further, these constructions can only be invoked once (or a very limited number of times) for security to hold. Computational fuzzy extractors are thus attractive, provided they can address efficiency problems under established computational hardness assumptions. Prior work in computational fuzzy extractors has been unable to address the noise level issue, as they would run in exponential time for $\Theta(m)$ errors. Especially in the silicon POK realm, small noise is virtually impossible to guarantee especially under high environmental stress.

To solve the noise level problem, we presented the first construction of a computational fuzzy commitment scheme with a trapdoor in this paper, using the LPN problem as the hard problem in our construction. The trapdoor is unusual in that it is part of the biometric source and is used to avoid noise. Due to the exponential reduction in key recovery complexity enabled by the trapdoor, our construction is able to correct $\Theta(m)$ errors in polynomial time. We show how error profiles obtained from a Field Programmable Gate Array implementation of PUFs subject to wide environmental variation can be efficiently corrected.

Finally, we relaxed the i.i.d. assumptions on the POK outputs showing that if correlation can be estimated, the only change to the construction is in the selection of parameters.

Acknowledgement: This research was partially supported by the National Science Foundation. Marten van Dijk was supported in part by AFOSR MURI under award number FA9550-14-1-0351.

References

[1] Akavia, A., Goldwasser, S., and Vaikuntanathan, V. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography*. Springer, 2009, pp. 474–495.

[2] Applebaum, B., Barak, B., and Wigderson, A. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing* (2010), ACM, pp. 171–180.

[3] Applebaum, B., Cash, D., Peikert, C., and Sahai, A. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *Advances in Cryptology - CRYPTO 2009*, S. Halevi, Ed., vol. 5677 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 595–618.

[4] Armknecht, F., Maes, R., Sadeghi, A., Standaert, O.-X., and Wachsmann, C. A Formalization of the Security Features of Physical Functions. In *IEEE Symposium on Security and Privacy (S&P)* (2011), pp. 397–412.

[5] Arora, S., and Ge, R. New algorithms for learning in presence of errors. In *Automata, Languages and Programming*. Springer, 2011, pp. 403–415.

[6] Becker, G. T. The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs. In *Cryptographic Hardware and Embedded Systems 2015 (CHES 2015)* (2015).

[7] Becker, G. T., Wild, A., and Güneysu, T. Security Analysis of Index-Based Syndrome Coding for PUF-Based Key Generation. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015* (May 2015).

[8] Bernstein, D. J., and Lange, T. Never trust a bunny. In *Radio Frequency Identification. Security and Privacy Issues*. Springer, 2013, pp. 137–148.

[9] Blum, A., Furst, M., Kearns, M., and Lipton, R. Cryptographic Primitives Based on Hard Learning Problems. In *Advances in Cryptology - CRYPTO 93*, D. Stinson, Ed., vol. 773 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1994, pp. 278–291.

[10] Blum, A., Kalai, A., and Wasserman, H. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM) 50*, 4 (2003), 506–519.

[11] Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varici, K., and Verbauwhede, I. SPONGENT: A Lightweight Hash Function. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, vol. 6917 of *Lecture Notes in Computer Science*. 2011, pp. 312–325.

[12] Bösch, C., Guajardo, J., Sadeghi, A.-R., Shokrollahi, J., and Tuyls, P. Efficient Helper Data Key Extractor on FPGAs. In *Proceeding Sof the 10th International Workshop on Cryptographic Hardware and Embedded Systems* (2008), CHES '08, pp. 181–197.

[13] Bowyer, K. W., Hollingsworth, K., and Flynn, P. J. Image understanding for iris biometrics: A survey. *Computer vision and image understanding 110*, 2 (2008), 281–307.

[14] Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., and Smith, A. Secure Remote Authentication Using Biometric Data. In *EUROCRYPT'05* (2005), pp. 147–163.

[15] Bringer, J., Chabanne, H., Cohen, G., Kindarji, B., and Zémor, G. Optimal iris fuzzy sketches. In *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on* (2007), IEEE, pp. 1–6.

[16] Damgård, I., and Park, S. Is Public-Key Encryption Based on LPN Practical? *IACR Cryptology ePrint Archive 2012* (2012), 699.

[17] Delvaux, J., and Verbauwhede, I. Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise. In *6th IEEE International Symposium on Hardware-Oriented Security and Trust - HOST 2013* (2013), pp. 137 – 142.

[18] Delvaux, J., and Verbauwhede, I. Attacking PUF-Based Pattern Matching Key Generators via Helper Data Manipulation. In *Topics in Cryptology - CT-RSA 2014*, vol. 8366 of *Lecture Notes in Computer Science*. 2014, pp. 106–131.

[19] Dodis, Y., Kanukurthi, B., Katz, J., Reyzin, L., and Smith, A. Robust Fuzzy Extractors and Authenticated Key Agreement From Close Secrets. *Information Theory, IEEE Transactions on 58*, 9 (Sept 2012), 6207–6222.

[20] Dodis, Y., Reyzin, L., and Smith, A. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - Eurocrypt 2004* (2004).

[21] Fuller, B., Meng, X., and Reyzin, L. Computational fuzzy extractors. In *Advances in Cryptology-ASIACRYPT 2013*. Springer, 2013, pp. 174–193.

[22] GAO, M., LAI, K., AND QU, G. A Highly Flexible Ring Oscillator PUF. In *The 51st Annual Design Automation Conference 2014, DAC '14* (2014), pp. 89:1–89:6.

[23] GASSEND, B. Physical random functions. Master's thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., Jan. 2003.

[24] GASSEND, B., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. Controlled Physical Random Functions . In *Proceedings of* 18^{th} *Annual Computer Security Applications Conference* (Silver Spring, MD, December 2002), Applied Computer Security Associates (ACSA).

[25] GASSEND, B., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security (CCS)* (2002).

[26] GASSEND, B., CLARKE, D., VAN DIJK, M., AND DEVADAS, S. Delay-based circuit authentication and applications. In *Proceedings of the 2003 ACM Symposium on Applied Computing* (March 2003). Extended version in Concurrency and Computation: Practice and Experience.

[27] GRAYBEAL, S., AND MCFATE, P. Getting out of the STARTing block. *Scientific American 261*, 6 (1989).

[28] GUIN, U., HUANG, K., DIMASE, D., CARULLI, J. M., TEHRANIPOOR, M., AND MAKRIS, Y. Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proceedings of the IEEE 102*, 8 (2014), 1207–1228.

[29] HAO, F., ANDERSON, R., AND DAUGMAN, J. Combining crypto with biometrics effectively. *Computers, IEEE Transactions on 55*, 9 (2006), 1081–1088.

[30] HILLER, M., MERLI, D., STUMPF, F., AND SIGL, G. Complementary IBS: Application Specific Error Correction for PUFs. In *IEEE Int. Symposium on Hardware-Oriented Security and Trust* (2012), IEEE.

[31] HILLER, M., WEINER, M., RODRIGUES LIMA, L., BIRKNER, M., AND SIGL, G. Breaking Through Fixed PUF Block Limitations with Differential Sequence Coding and Convolutional Codes. In *Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices* (2013), TrustED '13, pp. 43–54.

[32] HOLCOMB, D., BURLESON, W., AND FU, K. Power-up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers 58*, 9 (September 2009), 1198–1210.

[33] HOPPER, N. J., AND BLUM, M. Secure human identification protocols. In *Advances in cryptology: ASIACRYPT 2001*. Springer, 2001, pp. 52–66.

[34] HOSPODAR, G., MAES, R., AND VERBAUWHEDE, I. Machine Learning Attacks on 65nm Arbiter PUFs: Accurate Modeling poses strict Bounds on Usability. In *4th IEEE International Workshop on Information Forensics and Security (WIFS 2012)* (2012), pp. 37 – 42.

[35] JUELS, A., AND WATTENBERG, M. A Fuzzy Commitment Scheme. In *Proceedings of the* 6^{th} *ACM Conference on Computer and Commmunications Security* (1999), pp. 28–36.

[36] KARAKOYUNLU, D., AND SUNAR, B. Differential template attacks on PUF enabled cryptographic devices. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on* (Dec 2010), pp. 1–6.

[37] KIRCHNER, P. Improved Generalized Birthday Attack. *IACR Cryptology ePrint Archive 2011* (2011), 377.

[38] KIROVSKI, D. Anti-counterfeiting: Mixing the physical and the ditigal world. In *Towards Hardware-Intrinsic Security* (2010), A.-R. Sadeghi and D. Naccache, Eds., Springer, pp. 223–233.

[39] KOEBERL, P., LI, J., RAJAN, A., AND WU, W. Entropy loss in PUF-based key generation schemes: The repetition code pitfall. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on* (May 2014), pp. 44–49.

[40] KRAWCZYK, H., BELLARE, M., AND CANETTI, R. HMAC: Keyed-Hashing for Message Authentication.

[41] KUMAR, R., AND BURLESON, W. On design of a highly secure PUF based on non-linear current mirrors. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014, Arlington, VA, USA, May 6-7, 2014* (2014), pp. 38–43.

[42] LEVIEIL, É., AND FOUQUE, P.-A. An improved LPN algorithm. In *Security and Cryptography for Networks*. Springer, 2006, pp. 348–359.

[43] LIM, D. Extracting secret keys from integrated circuits. Master's thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., May 2004.

[44] LIM, D., LEE, J. W., GASSEND, B., SUH, G. E., VAN DIJK, M., AND DEVADAS, S. Extracting secret keys from integrated circuits. *IEEE Trans. VLSI Syst. 13*, 10 (2005), 1200–1205.

[45] LOFSTROM, K., DAASCH, W. R., AND TAYLOR, D. IC Identification Circuit Using Device Mismatch. In *Proceedings of ISSCC 2000* (February 2000), pp. 372–373.

[46] MAES, R., TUYLS, P., AND VERBAUWHEDE, I. Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)* (2009), pp. 332–347.

[47] MAES, R., TUYLS, P., AND VERBAUWHEDE, I. Soft Decision Helper Data Algorithm for SRAM PUFs. In *Proceedings of the 2009 IEEE International Conference on Symposium on Information Theory - Volume 3* (2009), ISIT'09, pp. 2101–2105.

[48] MAES, R., VAN HERREWEGE, A., AND VERBAUWHEDE, I. PUFKY: A Fully Functional PUF-based Cryptographic Key Generator. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems* (2012), CHES'12, pp. 302–319.

[49] MATHEW, S. K., SATPATHY, S. K., ANDERS, M. A., KAUL, H., HSU, S. K., AGARWAL, A., CHEN, G. K., PARKER, R. J., KRISHNAMURTHY, R. K., AND DE, V. A 0.19 pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International* (2014), IEEE, pp. 278–279.

[50] ORSHANSKY, M. Physically unclonable functions based on non-linearity of sub-threshold operation, 2015. US Patent 8,938,069.

[51] PAPPU, R. *Physical One-Way Functions*. PhD thesis, Massachusetts Institute of Technology, 2001.

[52] PARAL, Z., AND DEVADAS, S. Reliable and efficient PUF-based key generation using pattern matching. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* (2011), pp. 128–133.

[53] REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM) 56*, 6 (2009), 34.

[54] RÜHRMAIR, U., DEVADAS, S., AND KOUSHANFAR, F. Security based on Physical Unclonability and Disorder. In *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. Springer, 2012, ch. 4, pp. 65–102.

[55] RÜHRMAIR, U., SEHNKE, F., SÖLTER, J., DROR, G., DEVADAS, S., AND SCHMIDHUBER, J. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security (CCS)* (2010), ACM, pp. 237–249.

[56] RÜHRMAIR, U., SÖLTER, J., SEHNKE, F., XU, X., MAHMOUD, A., STOYANOVA, V., DROR, G., SCHMIDHUBER, J., BURLESON, W., AND DEVADAS, S. PUF Modeling Attacks on Simulated and Silicon Data. *Information Forensics and Security, IEEE Transactions on 8*, 11 (Nov 2013), 1876–1891.

[57] SUH, G. E. *AEGIS: A Single-Chip Secure Processor*. PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., Aug. 2005.

[58] SUH, G. E., AND DEVADAS, S. Physical unclonable functions for device authentication and secret key generation. In *ACM/IEEE Design Automation Conference (DAC)* (2007).

[59] TOBISCH, J., AND BECKER, G. T. On the Scaling of Machine Learning Attacks on PUFs with Application to Noise Bifurcation. In *Proceedings of RFIDSec 2015* (2015).

[60] VAN DER LEEST, V., PRENEEL, B., AND VAN DER SLUIS, E. Soft Decision Error Correction for Compact Memory-Based PUFs Using a Single Enrollment. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings* (2012), pp. 268–282.

[61] YU, M.-D. M., AND DEVADAS, S. Secure and robust error correction for physical unclonable functions. *IEEE Design and Test of Computers 27* (2010), 48–65.

[62] YU, M.-D. M., M'RAÏHI, D., SOWELL, R., AND DEVADAS, S. Lightweight and Secure PUF Key Storage Using Limits of Machine Learning. In *Cryptographic Hardware and Embedded Systems (CHES)*. 2011, pp. 358–373.

[63] YU, M. M., HILLER, M., AND DEVADAS, S. Maximum-likelihood decoding of device-specific multi-bit symbols for reliable key generation. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015* (2015), pp. 38–43.