Trapdoor Computational Fuzzy Extractors and Stateless Cryptographically-Secure Physical Unclonable Functions

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

Abstract—We present a fuzzy extractor whose security can be reduced to the hardness of Learning Parity with Noise (LPN) and can efficiently correct a constant fraction of errors in a biometric source with a "noise-avoiding trapdoor." Using this computational fuzzy extractor, we present a 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 reduced to the hardness of an LPN variant under 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, analysis of the security parameter for system parameter choices, and present experimental evidence that the construction is practical and reliable under a wide environmental range.

Index Terms—Fuzzy extractor, physical unclonable function, learning parity with noise, ring oscillators, physically obfuscated keys

1 Introduction

1.1 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 [26], [53]. 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 [29].

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

- C. Herder, L. Ren, and S. Devadas are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: {cherder, renling}@mit.edu, devadas@csail.mit.edu.
- M. van Dijk is with the Electrical and Computer Engineering, University of Connecticut, Storrs, CT. E-mail: martenvdijk@gmail.com.
- M.-D. (Mandel) Yu is with the R&D, Verayo Inc., San Jose, CA. E-mail: myu@verayo.com.

Manuscript received 9 Aug. 2015; revised 4 Nov. 2015; accepted 18 Jan. 2016. Date of publication 1 Mar. 2016; date of current version 18 Jan. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2016.2536609

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

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). The stateless property implies that there is no separate "provisioning" stage: the interface exposed by the PUF is static, and any valid query can be made at any time. 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 3). In the context of stateless PUFs, Gassend et al. [26] write: "An important direction of research is to find a circuit that is provably hard to break ...". In this paper we accomplish this objective.

1.2 Physically Obfuscated Keys

Physically Obfuscated Keys (e.g., [44]) 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: (1) a small number of challenge-response pairs, and (2) these 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/POK. If the parameters vary too much, the PUF or POK response will change. In the 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 table. 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 that can be stored on the verifier side. However, even when the helper data is stored off-device, there is still a requirement for storage on the PUF device for information-theoretically secure constructions.

The reason is subtle: generation of helper data can only be done once or a limited number of times, because the helper data leaks information about the POK outputs as mentioned above. To repeatedly generate helper data an arbitrary number of times under potentially different environmental conditions where POK outputs change significantly requires strong independence assumptions on POK outputs. If these assumptions cannot be made, 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.

1.3 Overview of Our Approach

In this paper, we show how to build a stateless PUF with a computational 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 [25], which is much more powerful than a conventional PUF that can only be used for authentication (cf. Section 7.3.3).

We accomplish our task in two steps. First, by making an assumption about the characteristics of the POK, namely, that it can provide "confidence" information, we demonstrate a computational fuzzy extractor that can correct $O(m)$ errors in polynomial time. The confidence information is never stored or exposed, and can thus be viewed as a *noise-avoiding trapdoor*.

Abstractly, confidence information can be thought of as information measured from the POK 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 POK 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 POK (e.g., SRAM state [33]) 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 is the ring oscillator POK (cf. Section 2.1) used in our case study (cf. Section 9), 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 and then discarded. Our PUF remains stateless.

Second, we show given a POK, how to set up a restricted version of a Learning Parity with Noise (LPN) problem such that a sateless secure PUF is enabled. 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. Using our error correction scheme, all challenge-response pairs are reliable in that a challenge contains necessary helper data for regeneration to always reproduce the same response. We prove that the stateless PUF protocol is secure under the random oracle model given the hardness of LPN and the strong assumption that the confidence information is independent of the actual bit values. We then relax this assumption by augmenting the construction to selectively inject random noise during challenge-response generation and show that this construction requires a variant of LPN hardness assumption under the random oracle model.

1.4 Our Contributions

We defer a detailed comparison to prior work to Section 3, and elaborate on our contributions below.

1) We propose a fuzzy extractor that is able to correct $\Theta(m)$ errors in an m-bit POK output and recover an n-bit key in polynomial time. Two novel ideas enable this result.
   - First, we take advantage of the confidence information as a trapdoor to a hard problem. The confidence information is never stored; it is regenerated (and may change on each key extraction) and then erased (cf. Section 4).
   - Second, compared with traditional LPN/LWE cryptosystems, we use the number of equations m in a novel way—to provide redundancy—and show that setting $m = \Theta(n^2)$ results in a negligible failure probability for key recovery even with $\Theta(m)$ errors (cf. Section 5).
2) LPN is hard given independent, identically distributed (i.i.d.) noise (which corresponds to POK bits in our construction). To provide more flexibility with respect to POK distributions, we give a reduction for a large class of noise distributions that are significantly more relaxed (cf. Section 6).
3) We provide stateless PUF constructions and show that breaking the PUF requires breaking LPN or a variant of LPN under the random oracle model (cf. Sections 7 and 8).
4) We provide experimental evidence that our fuzzy extractor and our PUF construction are efficient and work under significant environmental variation (cf. Section 9).

1.5 Organization

We give background for ring oscillator POKs and LPN in Section 2. We discuss related work in Section 3. Section 4 describes how noisy POK outputs can be corrected securely using a computational fuzzy extractor. We analyze the reliability of key regeneration in Section 5, where we show that the confidence information serves as a trapdoor. Section 6 gives a security analysis of our fuzzy extractor, relaxes the assumptions on the POK bits, and also provides quantitative analysis on parameters. Section 7 gives formal definitions and our construction of a stateless PUF, with the corresponding security proofs given in Section 8. We show in Section 9 that our constructions can be efficiently built using ring oscillator POKs. We conclude the paper in Section 10.

2 Background

2.1 Ring Oscillator POK

Following the formalization provided in [4], we first define a Physically Obfuscated Key (POK) as a physical function wherein there is only one challenge. A POK returns a single m-bit response, denoted as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m\}$ in this paper.

If the POK response is completely stable across measurements, then constructing a stable secret key or strong PUF would be trivial: just use the POK output as the secret key.

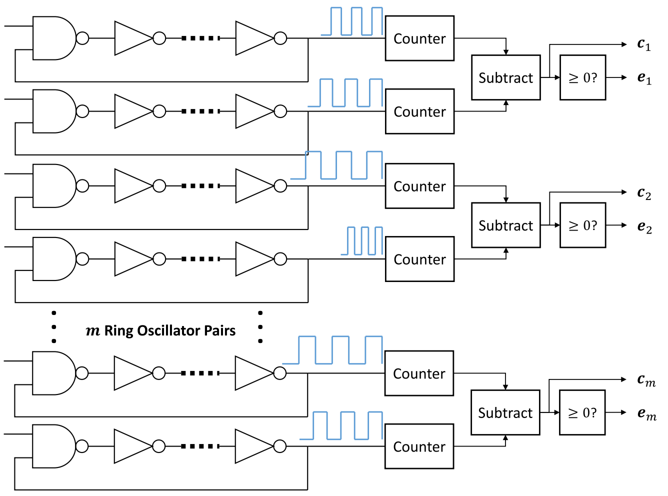


Fig. 1. A basic Ring Oscillator POK with m differential pairs. Note that in addition to the output bits $\mathbf{e}_i$, confidence values $\mathbf{c}_i$ may be made available to the surrounding logic. These confidence values are in the form of the actual differential count between the two ring oscillators,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

Unfortunately, the POK response in practice will be slightly different each time due to internal noise, i.e.,

$$\mathbf{e} = \mathbf{e}_{\text{const}} + \mathbf{e}_{\text{noise}},$$

where $\mathbf{e}_{\text{const}}$ is the same for each call to the POK, and $\mathbf{e}_{\text{noise}}$ is sampled at random from some distribution over $\{0,1\}^m$. Error-correcting the POK response to tolerate the noise is a major challenge.

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 [57] 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. Fig. 1). While there have been improvements in ring oscillator structures (e.g., [23]), our case study uses the basic structure of Fig. 1.

2.2 Learning Parity with Noise

The Learning Parity with Noise problem is a famous open problem that is widely conjectured to be hard [52], as the best known algorithm is slightly subexponential ($2^{\Omega(n/\log n)}$) [5], [8], [10], [30], [41]. As a result, this problem has since been used as the foundation of several cryptographic primitives [2], [3], [9], [34].

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 $\mathbf{b} \in \{0,1\}^m$ (where $\cdot$ is a dot product) as:

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

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.

Conjecture 2.1 (LPN Hardness [34]). 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 $\mathsf{poly}(n, 1/(\frac{1}{2} - \tau))$ with non-negligible probability in n, where χ is an Bernoulli distribution with bias τ.

The LPN problem can be thought of as a special case of the Learning With Errors (LWE) problems discussed by Regev [52], by allowing the equations to instead be modulo a prime number q (as opposed to 2). However, Regev's reduction to the shortest independent vector problem (SIVP) 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 fuzzy extractor and a stateless PUF based on LPN, but our constructions can be extended to the LWE case.

3 Related Work

3.1 PUF/POK Proposals

Although many of the architectures that integrate POKs and 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 [37]. 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 [53]. One example of early reported usage of unique objects for security was proposed for the identification of nuclear weapons during the cold war [28]. 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 [50]. However, to our knowledge, none of these proposals has an associated computational security argument that shows hardness of model-building or machine learning attacks.

Unlike the proposals described above, silicon PUFs, introduced in [26], do *not* require an external measurement apparatus. 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 [27], feedforward Arbiter [43] and XOR Arbiter PUF [57]. Machine learning attacks such as those of [54] and [6] have successfully attacked these constructions to create software clones. While other constructions using nonlinear circuit elements (e.g., [42], [40], [49]) have not yet been broken to our knowledge, these constructions do not as yet have clear computational security reductions.

3.2 Error Correction for Silicon POKs

Silicon POK key generation was first introduced using Hamming codes in [24] and more details were presented in [56]. The security argument is information-theoretic. Specifically, if one requires a k-bit secret from n bits generated by the POK, then at most $n - k$ bits could be exposed. The number of correctable errors is quite limited in this approach.

3.3 Fuzzy Extractors for Silicon POKs

Fuzzy extractors [21] convert noisy biometric data (either human or silicon) into reproducible uniform random strings, which can then serve as secret keys in cryptographic applications. 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 noisy 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 POK 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 POK 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 [38], the entropy loss associated with the use of the information-theoretic entropy accumulator alone is ≥ 128 bits due to the leftover hash lemma.

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

Perfectly i.i.d. Entropy Source. There are several works that created helper data that is information-theoretically secure. [59] uses POK 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 POK 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 [31] and [32].

A soft-decision POK 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, they 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 POK 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 6.1).

Computational security based on machine learning heuristics. There were several works [51] [60] [61] that created helper data that is heuristically secure based on results of state-of-the-art machine learning attacks on PUFs [55]. These designs used a candidate strong PUF based on XORs [57] but leak only a limited number of PUF response bits as helper data to generate a key. After several years of attacks by several groups around the world [18], [35], [54], [55], [58], the basic XOR PUF was "broken" in 2015 [6]. There has not been a strong PUF architecture with a reduction from a formal strong PUF security definition (e.g., "strong unpredictability" in [4]) to 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 POK fuzzy extractor [48]. The paper accounted for the information-theoretic loss of the error correction helper data, using code-offset syndrome [21], 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.

Under the assumption that confidence values are independent of the measurement values, information-theoretically secure extractors can also produce a stateless construction as we do in Section 7. However, in our construction, we show how this assumption can be relaxed through a computational hardness assumption of a variant LPN problem.

3.4 Computational Fuzzy Extractors

Fuller et al. [22] 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 secure sketches 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 output by the POK.

Fuller et al., expect that the exponential complexity in correcting a linear number of errors is unlikely to be overcome, since there is no place to securely put a trapdoor in a fuzzy extractor. We recognize that certain kinds of silicon biometric sources have dynamically regenerated confidence information that does not require persistent storage memory and can in fact serve as a trapdoor (cf. Sections 4 and 5). We show that security can be maintained even if the bits generated by the biometric source are correlated (cf. Definition 6.1).

3.5 Helper Data Manipulation

The issue of helper data manipulation has been addressed with robust fuzzy extractors [14], [20]. Their use of a helper data hash do not address recent helper data manipulation attacks in [19], [36], including ones that take advantage of the linear, bitwise-XOR nature of code-offset helper data as applied to linear error correction codewords.

In the stateless PUF of Section 7, the helper data comprises (part of) the challenge. Since in an LPN-based fuzzy extractor 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 [39]. This results in schemes that are secure in a computational sense against active adversaries that modify the helper data in our fuzzy extractor or stateless PUF construction.

4 Fuzzy Extractor Using LPN

This section considers how to reliably reconstruct a key $\mathbf{s}$ from a noisy POK (or a noisy biometric source). We start with the fuzzy extractor scheme described in [22], which leverages LWE to extract a pseudorandom string from fuzzy data. We will begin by translating that work from LWE to LPN discussed in Section 2.2.

Construction 4.1. *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:*

```
1: procedure $(\mathbf{A}, \mathbf{b}) \leftarrow \mathsf{Gen}\ (1^k)$
2:     Input $\mathbf{e} \in \{0,1\}^m$ from the POK (modeled by some
       distribution $\chi$ over $\{0,1\}^m$).
3:     Sample $\mathbf{A} \in \{0,1\}^{m \times n}$ uniformly at random.
4:     Sample $\mathbf{s} \in \{0,1\}^n$ uniformly at random.
5:     Compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.
6:     return $(\mathbf{A}, \mathbf{b})$.
7: end procedure
1: procedure $\mathbf{s} \leftarrow \mathsf{Rep}(\mathbf{A}, \mathbf{b})$
2:     Input $\mathbf{e}' \in \{0,1\}^m$ from the POK.
3:     Let $\mathbf{s} = \mathsf{Decode}_t(\mathbf{A}, \mathbf{b}, \mathbf{e}')$.
4:     return $\mathbf{s}$.
5: end procedure
```

This construction is exactly analogous to Construction 4.1 of Fuller et al. [22], translated to LPN from LWE (all equations are $\bmod 2$ instead of $\bmod q$). Therefore, we state the following theorem without proof, as it is analogous to Theorem 4.7 of [22] except under the LPN hardness conjecture.

Theorem 4.2. *Let k be a security parameter. If Conjecture 2.1 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 $s = \mathsf{poly}(k)$ and $t = O(\log n)$ bit errors, Construction 4.1 is a $(\{0,1\}^m, \chi, n - o(n), t)$ fuzzy extractor that is (ϵ, s)-hard, with failure rate $\delta = e^{-\Omega(k)}$ (cf. Definition 2.5 of [22]).*

In the above construction, Decode keeps picking random sets of n equations $\mathbf{b}_i = \mathbf{A}_i\mathbf{s} + \mathbf{e}_i$ and solves for $\mathbf{s}$ (see $\mathsf{Recovery}$ in Algorithm 1 for details). If $t = O(\log n)$, Decode succeeds with overwhelming probability after a polynomial number of trials. We will now describe a new extractor algorithm based on LPN that can correct $\Theta(m)$ errors in polynomial time. Before presenting the extractor formally, we present an intuitive description.

Algorithm 1. The LPN Trapdoor Fuzzy Extractor Algorithm.

```
1: procedure Fab($1^k, \delta, \eta$)    // Represents the fabrication step. It takes the security parameter $k$, the desired probability of recovery failure $\delta$, a $\eta$ term that characterizes correlation in the POK bits (defined in Definition 6.1)
2:     Select the size of the secret vector $n$ for the desired security level $k$ based on $\eta$ (details in Section 8).
3:     Compute $m$ such that with probability greater than $1 - \epsilon_1$, at least $m' = \Theta(n)$ of the $m$ POK bits are "stable" over relevant noise/environmental parameters. Define "stable" to be $\Pr(\mathbf{e}'_i \neq \mathbf{e}_i) \leq \epsilon_2$. The choice of $m', \epsilon_1, \epsilon_2$ along with other details will be presented in Section 5.
4:     Manufacture POKs that each produce $m$ bits internally.
5: end procedure
6:
7: procedure($\mathbf{A}, \mathbf{b}$) ← Gen ($n$) //Gen takes the size of the secret vector $n$ (calculated in Fab)
8:     Measure the $m$ POK bits as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_m\}$.
9:     Generate a uniformly random secret vector $\mathbf{s} \in \{0,1\}^n$.
10:    Compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.
11:    Discard $\mathbf{s}$ and $\mathbf{e}$.
12:    Return $\mathbf{b}$.
13: end procedure
14:
15: procedure $T$ ← Project ($\mathbf{c}'$) // Determines the "stable" POK bits to be used in Recovery.
16:    Use measured confidence information $\mathbf{c}'_i$ to find $m' = \Theta(n)$ stable POK bits.
17:    Let $T$ be the set of these stable bits. Return $T$.
18: end procedure
19:
20: procedure s ← Recovery ($\mathbf{e}', T$)    // Represents the augmented key recovery algorithm. In addition to the noisy POK measurement $\mathbf{e}'$, this function also takes $T$, the set of stable bits in $\mathbf{e}'$.
21:    Randomly select $n$ out of the $m'$ stable bits.
22:    Use Gaussian elimination to solve for $\mathbf{s}$ on the $n$ selected bits.
23:    Check if $\mathbf{b}_i = \mathbf{A}_i \cdot \mathbf{s} + \mathbf{e}'_i$ on the remaining $m - n$ equations. An error rate of $\approx 50\%$ implies that the derived $\mathbf{s}$ is incorrect. A significantly lower error rate (e.g., 25%) indicates $\mathbf{s}$ is correct.
24:    If $\mathbf{s}$ is incorrect, go back to step 21); else output $\mathbf{s}$.
25: end procedure
```

4.1 Intuitive Description

Recall the description of the LPN problem in Section 2.2, which is also depicted in Fig. 2. The above construction uses the POK output as the $\mathbf{e}_i$ values. Therefore, an adversary learns the equations with probability of error being $\Pr(\mathbf{e}_i = 1) = \tau$, where τ relates to the entropy of the POK. Having access to the POK allows one to regenerate $\mathbf{e}'_i$, where $\Pr(\mathbf{e}'_i = 1) = \tau$ and $\Pr(\mathbf{e}'_i \neq \mathbf{e}_i) = \tau' \ll \tau$. The regeneration is imperfect due to intrinsic noise of the POK as well as environmental changes. The LPN problem remains hard even for small τ' implying that key recovery will run in exponential time for $\Theta(m)$ number of errors.

A critical enabling property of LPN/LWE is that if one can identify *any* set of n bits that are correct ($\mathbf{e}'_i = \mathbf{e}_i$),

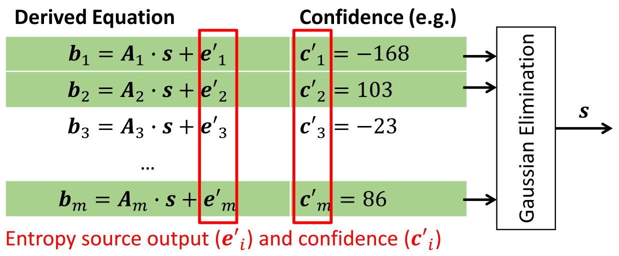

Derived Equation
Confidence (e.g.)
$b_1 = A_1 \cdot s + e'_1$
$c'_1 = -168$
$b_2 = A_2 \cdot s + e'_2$
$c'_2 = 103$
$b_3 = A_3 \cdot s + e'_3$
$c'_3 = -23$
...
$b_m = A_m \cdot s + e'_m$
$c'_m = 86$
Gaussian Elimination
s
Entropy source output (e'_i) and confidence (c'_i)

Fig. 2. 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 $\mathbf{e}'_i = \text{Sign}(\mathbf{c}'_i)$). Gaussian elimination is then used on these selected equations to extract the secret key.

then one can use Gaussian elimination to solve for $\mathbf{s}$. Therefore, our key intuition is that access to confidence information during the *regeneration* of the POK bits helps the extractor decide which bits are more likely to be stable (the set of stable bits may be different from measurement to measurement). Then, Gaussian elimination is performed on the set of equations corresponding to these stable bits.

Of course, one must architect the system such that there are enough stable POK bits during each measurement. To this end, the LPN/LWE problem allows *arbitrary redundancy* in the number of equations supplied. Therefore, we can supply enough equations such that with high probability (see Section 5) the recovery succeeds.

Initially, this may sound similar to using the "mask" data in some POK implementations. However, this approach is fundamentally different and has superior security properties, as we recognize the biometric source itself to be a hidden trapdoor to a hard problem. The confidence information is *discarded* after use and never exposed. The security proof for this new construction will therefore be identical to that of Construction 4.1, since the adversary receives identical information.

4.2 Detailed Construction

In the following description, we will refer to the POK bits as $\mathbf{e} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_m\}$, ($\mathbf{e}_i \in \{0,1\}$) and their noisy counterparts (measured during response verification) as $\mathbf{e}' = \{\mathbf{e}'_1, \mathbf{e}'_2, \ldots \mathbf{e}'_m\}$, (once again, $\mathbf{e}'_i \in \{0,1\}$). Moreover, the confidence information associated with these noisy measurements will be denoted as $\mathbf{c}' = \{\mathbf{c}'_1, \mathbf{c}'_2 \ldots \mathbf{c}'_m\}$, where $\mathbf{c}'_i \in \mathbb{Z}$ (as shown in Fig. 2).

We describe the algorithms associated with the key extraction below. Typically, a fuzzy extractor, information-theoretic or computational, has the functions Gen and Rep, where Gen produces the public helper information, and Rep takes the noisy biometric bits and public helper information and returns the error-corrected key. This paper expands this construction to include four functions in the extraction process as shown in Algorithm 1.

Before looking into the effects of errors on Algorithm 1, there are several notes to be made.

First, the Fab algorithm can be viewed as system design steps that choose parameters for the desired security and reliability. Project and Recovery together correspond to Rep in a fuzzy extractor, and Recovery is exactly the same as Decode_t in Construction 4.1.

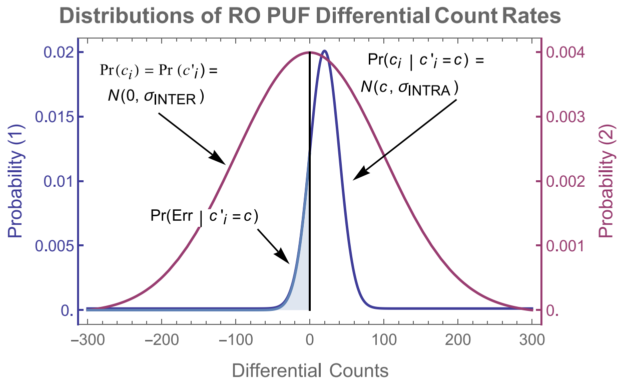


Fig. 3. Distribution of confidence information for different POK bits when measured repeatedly over time/environmental parameters. The magenta curve corresponds to the distribution of confidence information across different devices. The blue curve corresponds to the distribution of measured confidence information from the same device in different conditions. We show the probability of error given a confidence measurement c as the integral of the shaded region.

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 4.2. Furthermore, due to the simultaneous hardcore bits of $\mathbf{s}$ in the LPN problem, $\mathbf{s}$ has $n - o(n)$ pseudorandom bits [1].

Third, the matrix $\mathbf{A}$ can be made a public global system parameter as opposed to per-device output to reduce helper data size; this leaves $\mathbf{b}$ as the only per-device helper data. This will be the same global $\mathbf{A}$ in the stateless PUF construction of Section 7, though there exists a much more fundamental security reason to make $\mathbf{A}$ global there.

Lastly, 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 easier problem and can finish in polynomial time. This will be the focus of the next section.

5 Noise-Avoiding Trapdoors

In Section 4, Project leverages confidence information that a bit is regenerated correctly. This section will explore the asymptotic noise tolerance and efficiency of our system, and the required properties of the POKs (biometric source) to provide confidence information.

$\mathbf{c}_i$ are random variables representing the confidence information of the ith POK bit at the time of initial challenge-response generation. Next, $\mathbf{c}'_i$ are random variables representing the confidence information of the ith POK bit at some point in time later. We note again that confidence data are extracted upon measurement of a POK bit, and are never persistently stored.

Define the corresponding POK bit to be a random variable $\mathbf{e}_i = \mathsf{Sign}(\mathbf{c}_i)$, and $\mathbf{e}'_i = \mathsf{Sign}(\mathbf{c}'_i)$. Crucially, if the confidence is high for a particular bit, $\Pr(\mathbf{e}'_i = \mathbf{e}_i) \approx 1$.

To provide concrete analysis, we consider the probability distribution of the $\mathbf{c}_i$ and $\mathbf{c}'_i$ random variables, and assume they follow the same zero-mean Gaussian with variance σ_{INTER}, shown in Fig. 3. Note that this directly implies that $\Pr(\mathbf{e}_i = 1) = \tau = 1/2$ for the LPN problem. In actual physical systems, there will be a bias towards 1 or 0, but we will see that assuming a 0.5 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 8.)

Now, given that $\mathbf{c}_i$ and $\mathbf{c}'_i$ represent the random variables for measuring the *same* bit, the conditional distribution $\Pr(\mathbf{c}_i|\mathbf{c}'_i = c)$ is much narrower (where c is the actual measured value of $\mathbf{c}'_i$ at regeneration). This distribution is modeled to be a Gaussian distribution with mean c and variance σ_{INTRA}, also shown in Fig. 3.

Also note that $\mathbf{c}_i$ and $\mathbf{c}'_i$ represent the same POK bit measured at different times, so they have the same distribution (with no prior knowledge). Therefore, $\forall c$, $\Pr(\mathbf{c}_i|\mathbf{c}'_i = c) = \Pr(\mathbf{c}'_i|\mathbf{c}_i = c)$. In other words, one can use the confidence information collected during the fabrication step to reason about the probability of error at regeneration, 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 of the confidence information is the integral of the shaded region in Fig. 3, in particular, the CDF up to 0:

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

5.1 Fabrication/Provisioning

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 $\Pr(\mathbf{e}'_i \neq \mathbf{e}_i) < \epsilon_2$ over relevant environmental parameters.

To do this, we recognize that requiring $\Pr(\mathbf{e}'_i \neq \mathbf{e}_i|\mathbf{c}'_i = c) < \epsilon_2$ sets a threshold c_T on $|c|$ in Equation (1). If for a particular bit $|c| > c_T$, then the bit is stable. Plug these requirements into Equation (1) and solve for c_T (define Erf^{-1} as the inverse of Erf):

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

Therefore, the probability that a given bit is stable (has $|c| > c_T$) can be computed by integrating the PDF of $\Pr(\mathbf{c}'_i)$, or equivalently $\Pr(\mathbf{c}_i)$:

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

The inequality is because the probability of a bit being stable is *smallest* when the bit bias is 0.5 (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 (2) gives:

$$P_{\text{ST}} > 1 - \mathsf{Erf}\left(\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}\mathsf{Erf}^{-1}(1 - 2\epsilon_2)\right). \quad (3)$$

The final step is to compute m such that at least m' POK bits will be stable 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.

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

Rearranging,

$$m \geq \frac{1}{P_{\mathrm{ST}}}\Big(m' - \log(\epsilon_1) + \sqrt{\log(\epsilon_1)(\log(\epsilon_1) - 2m')}\Big). \quad (4)$$

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.

5.2 Projection/Extraction and Showing the "Trapdoor"

The extension of the above analysis to the Project algorithm is comparatively simple. Project simply selects a set T of $m' = \Theta(n)$ bits that have measured confidence $\mathbf{c}'_i = c$ where $|c| > c_T$. Because of the Fab algorithm, we can be confident that we will find m' such bits with overwhelming probability.

We need "truly stable" bits to perform Gaussian elimination, but the bits in T defined above only guarantee $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i | \mathbf{c}'_i = c) < \epsilon_2$ (i.e., likely correct but not certainly). Define t' as the number of bits that are not truly stable in T. If we set $\epsilon_2 = \Theta(1/n)$, then $\mathsf{E}(t') = \Theta(1)$, and a Chernoff bound gives $\mathsf{Pr}(t' > \alpha \log n) < e^{-\Theta(\log^2 n)}$. So $t' = o(\log n)$ with overwhelming probability.

In Recovery, we randomly select n out of the m' bits to perform Gaussian elimination. If the n selected bits are all "truly stable," Gaussian elimination on them will yield the correct $\mathbf{s}$ and Recovery succeeds. The above probability is given by

$$\frac{\binom{m'-t'}{n}}{\binom{m'}{n}} > \left(1-\frac{t'}{m'-n}\right)^n \approx \exp\left(-\frac{nt'}{m'-n}\right) = \frac{1}{\mathsf{poly}(n)}.$$

Therefore, after $\mathsf{poly}(n)$ number of iterations, Recovery finds the correct $\mathbf{s}$ with overwhelming probability. The overall failure probability—accounting for all types of failures (have less than m' stable bits, $t' = \omega(\log n)$, or fail to select n truly stable bits in all iterations)—is at most $\epsilon_1 + (1-\epsilon_1)\mathsf{negl}(n)$. We can set $\epsilon_1 = \Theta(2^{-n})$ to get overall negligible failure probability.

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

5.3 Setting m

We have set $\epsilon_1 = \Theta(2^{-n})$, $\epsilon_2 = \Theta(1/n)$ and $m' = \Theta(n)$. To compute m from Equations (3) and (4), we need to characterize $\sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}}$, which decides P_{ST}.

Define $\sigma_r = \sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}}$. We first consider a *worst-case*: $\sigma_r = 1$. In this case, Equation (3) reduces to $P_{\mathrm{ST}} = 2\epsilon_2$. Plug them into Equation (4), and one obtains:

$$m = \frac{1}{2\epsilon_2}\Big(m' + n + \sqrt{n(n+2m')}\Big) = \Theta(n^2).$$

In reality, the ratio $\sigma_r < 1$, so the hidden constant in $\Theta(n^2)$ is small, as will be seen in Section 9.

5.4 Improving on the Trapdoor

The above asymptotic result will be improved if we assume $\sigma_r = o(1)$. For example, let us pick σ_r such that P_{ST} is asymptotically constant.[2]

To accomplish this, recognize that $\mathsf{Erf}^{-1}(1-2\epsilon_2) \leq \sqrt{-\log(\epsilon_2)}$ as $\epsilon_2 \to 0$ [16]. Therefore, if $\sigma_r \leq c/\sqrt{-\log(\epsilon_2)}$ for some constant c, then $P_{\mathrm{ST}} \geq 1 - \mathsf{Erf}(c)$ for all n, and therefore $m = \Theta(n)$.

Note that the bound of $\sigma_r \leq c/\sqrt{-\log(\epsilon_2)}$ is very close to constant. For example, set $c = 1$ and $\epsilon_2 = 1/n$. For $n = 128, 256$, we find $\sigma_r < 0.45, 0.42$, respectively.

Finally, consider the effect of $\sigma_r \leq c/\sqrt{-\log(\epsilon_2)}$ on the number of correctable errors of the fuzzy extractor. Integration of the conditional probability distribution in Equation (1) (cf. Fig. 3) results in the associated marginal distribution (the error probability):

$$\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) = \frac{1}{2} - \frac{1}{\pi}\tan^{-1}(1/\sigma_r).$$

A constant σ_r as in the previous subsection clearly implies $\Theta(m)$ errors. For $\sigma_r = o(1)$, $\mathsf{Pr}(\mathbf{e}'_i \neq \mathbf{e}_i) = \Theta(\sigma_r)$ as $\sigma_r \to 0$. This implies that with $m = \Theta(n)$ one can no longer correct $\Theta(m)$ errors asymptotically; instead, the maximum number of correctable errors is $O(m\sigma_r)$. For practical key sizes the impact on error correction is minimal.

6 LPN Fuzzy Extractor Security Analysis and Assumptions

The proof of security for an LPN fuzzy extractor using confidence information is identical to Theorem 4.2. This is because the additional confidence information (which may or may not be correlated with the actual value of the POK bit) described in Section 5 that is used to help extract the key is never revealed.

6.1 Assumptions on POK Outputs

The POK outputs are used as the noise term in the LPN problem, and our fuzzy extractor construction is secure if the POK outputs are i.i.d. We now provide a significantly relaxed definition of POK source entropy under which the fuzzy extractor construction remains secure. In particular, the following definition describes the class of sources that are secure with LPN and hence our fuzzy extractor.

Definition 6.1. *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 $mL \times mL$ full rank matrix M, and mL-dimensional*

2. Note that one can make $\epsilon_2 = \Theta(\log(n)/n)$ and still brute-force correct in polynomial time. This does not impact the asymptotic analysis later in this section, so we ignore it.

vector N). Select a k-bit string f according to an arbitrary distribution over $\{0,1\}^k$ that can be sampled in polynomial time.

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 than an i.i.d. distribution, as it allows for certain bits from the same/different entropy sources to be correlated. For example, consider the ith bit of LPN problem A, and the jth 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 6.2 proves the security of the system.

Lemma 6.2. *If the entropy sources for a collection of LPN fuzzy extractors have a joint distribution that can be described by Definition 6.1 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 6.2 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 6.1 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 6.1. Specifically, Definition 6.1 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.

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 $F(\mathbf{e}) = M \cdot \mathbf{e} + N$ (M is an $m \times m$ dimensional matrix and N is an m-dimensional vector), then one can derive a different LPN problem:

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

$$\begin{aligned} &\text{Problem 1}: \begin{cases} \mathbf{b}_1^1 = \mathbf{A}_1^1 \cdot \mathbf{s}^1 + \mathbf{e}_1^1 \\ \mathbf{b}_2^1 = \mathbf{A}_2^1 \cdot \mathbf{s}^1 + \mathbf{e}_2^1 \\ \vdots \\ \mathbf{b}_m^1 = \mathbf{A}_m^1 \cdot \mathbf{s}^1 + \mathbf{e}_m^1 \end{cases} \\ &\text{Problem 2}: \begin{cases} \mathbf{b}_1^2 = \mathbf{A}_1^2 \cdot \mathbf{s}^2 + \mathbf{e}_1^2 \\ \mathbf{b}_2^2 = \mathbf{A}_2^2 \cdot \mathbf{s}^2 + \mathbf{e}_2^2 \\ \vdots \\ \mathbf{b}_m^2 = \mathbf{A}_m^2 \cdot \mathbf{s}^2 + \mathbf{e}_m^2 \end{cases} \\ &\quad\vdots \\ &\quad L \quad \text{Problems.} \end{aligned} \tag{5}$$

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 (5).

Now, we recognize (where $|$ is concatenation) that to transform a set of problems with $\mathbf{e}^1|\mathbf{e}^2|\ldots\ \mathbf{e}^L$ into a set of problems with $F(\mathbf{e}^1|\mathbf{e}^2|\cdots\mathbf{e}^L)$, one can simply concatenate the aforementioned equations (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 $\mathbf{b}_{\text{TOT}} = \mathbf{b}^1|\mathbf{b}^2|\cdots\mathbf{b}^L$.
3) Set $\mathbf{A}_{\text{TOT}} = \mathbf{A}^1|\mathbf{A}^2|\cdots\mathbf{A}^L$.

4) For j from 0 to k, define $F_j(x) = M_j \cdot x + N_j$. If $p_j = 1$, set $\mathbf{b}_{\text{TOT}} = M_j \cdot \mathbf{b}_{\text{TOT}} + N$, and set $\mathbf{A}_{\text{TOT}} = M \cdot \mathbf{A}_{\text{TOT}}$. 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 $\mathcal{A}$, and we obtained this problem by modifying bits of the public parameters *only*. Moreover, since the matrix M in each transformation is full rank, the original secret vectors remain the only solutions to the new LPN problems. Therefore, if $\mathcal{A}$ exists, and recovers at least one $\mathbf{s}$ with non-negligible advantage, then $\mathcal{B}$ can output that $\mathbf{s}$ to break the 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 parts of the set of LPN problems. This operation produces a new set of LPN problems that are *statistically identical* to LPN problems with correlated noise bits, while the secret vectors remain the only solutions.

Also note that a corollary of Lemma 6.2 is that $n - o(n)$ bits of $\mathbf{s}$ are pseudorandom, even in the presence of correlated bits of $\mathbf{e}$. 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 [22]. The proof is similar for the correlated LPN construction, as it is independent of the transformations performed in Lemma 6.2.

6.2 Security Parameter Derivation

The security goal is that an adversary given helper data must perform $\Omega(2^k)$ operations (k is the security parameter) to discover the secret key. We show below that for our system a key size of $n = 128$ results in a security parameter of $k = 128$ against the best known attacks. The equality of key size and security parameter is unusual for security constructions with formal hardness reduction, and is especially unusual for LPN cryptosystems.

There are two key factors enabling this property. First, recognize that typical LPN-based cryptosystems must have a low error rate (e.g., $\tau = \Pr(\mathbf{e}_i = 1) = 0.0024$ [17]) to ensure correct decryption/verification. 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 full entropy in the POK data. However, real POK data is not ideal and may not have full entropy. To be conservative, we pessimistically assume $\tau = \eta = 0.4$ and that the POK bits are correlated in a way that LPN is still hard (formalized in Definition 6.1 and Lemma 6.2).

The second factor is that number of equations in our construction is limited to $m \in O(n^2)$. Current best LPN algorithms are based on the BKW algorithm [10], which requires $m = 2^{O(n/\log n)}$. In order to successfully attack the LPN fuzzy extractor, one would have to use the technique from Lyubashevsky [45], which works with $m = O(n^{1+\epsilon})$ equations, but immediately increases the runtime to $2^{O(n/\log\log n)}$.

TABLE 1
Comparison of Performance of LPN Algorithms against an LPN Fuzzy Extractor with $\tau_L = \frac{1}{2} - 1.31 \times 10^{-48}$, $n = 128$

	Time Complexity	Security Parameter
BKW	$na(2^{b+1}(1-2\tau_L)^{-2^a}\ln(\frac{b}{\theta}) + (a-1)2^b)$	2^{247}
LF1	$b2^b + na(8\ln(\frac{2^b}{\theta})(1-2\tau_L)^{-2^a} + (a-1)2^b)$	2^{135}
LF2	$3 \cdot 2^b na + b2^b$	N/A

Set $\theta = 1/3$ to achieve 50 percent success probability [12]. The security parameter is taken for optimal choices of a, b (not shown). The security parameter of LF2 *is N/A, because there is no setting of parameters that results in the algorithm converging.*

The idea of Lyubashevsky's algorithm is to generate more equations from the given $m = O(n^{1+\epsilon})$ equations, increasing the noise rate to

$$\tau_L = \frac{1}{2} - \frac{1}{2}\left(\frac{1-2\tau}{4}\right)^{\frac{2n}{\epsilon\log n}}, \tag{6}$$

and then using other LPN algorithms, such as BKW [10], LF1, LF2 [41] as a black box with the increased error rate. For $m = \Theta(n^2)$ ($\epsilon = 1$), $n = 128$ and $\tau = 0.4$, $\tau_L = \frac{1}{2} - 1.31 \times 10^{-48}$.

The recent analysis from [12] shows that the LF1, LF2 algorithms empirically have the best performance in the limit of high noise ($\tau_L \to 0.5$). Table 1 compares BKW, LF1, LF2. Note that each of the above algorithms performs worse than brute-force or does not succeed at all. Therefore, we take $n = 128$ for a security parameter of $k = 128$.

7 Stateless PUF Construction

7.1 Stateless PUF Definition

A *Stateless PUF* is a pair of functions $\mathbf{PUF} = \{\mathsf{Gen}_{\text{POK}}, \mathsf{Ver}_{\text{POK}}\}$ with access to a POK, where $\mathsf{Gen}_{\text{POK}}$ is responsible for generating and outputting challenge-response pairs, while $\mathsf{Ver}_{\text{POK}}$ takes a challenge as input, and outputs a response. The intent is for $\mathsf{Gen}_{\text{POK}}$ to be called multiple times by a verifier over a secure channel 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 $\mathsf{Ver}_{\text{POK}}$.

Definition 7.1. *A (m, χ) stateless PUF is a pair of randomized probabilistic polynomial time procedures $\{\mathbf{c}, \mathbf{r}\} \leftarrow \mathsf{Gen}_{\text{POK}}(1^k)$, and $\mathbf{r} \leftarrow \mathsf{Ver}_{\text{POK}}(\mathbf{c})$ where*

- *The challenge-response generation algorithm $\mathsf{Gen}_{\text{POK}}(1^k)$ takes as argument the security parameter k. It returns a challenge-response pair $\{\mathbf{c}, \mathbf{r}\}$, with $\mathbf{c}, \mathbf{r} \in \{0,1\}^*$ and $|\mathbf{c}|, |\mathbf{r}| \in \mathsf{poly}(k)$. The subscript* POK *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 $\mathbf{r} \leftarrow \mathsf{Ver}_{\text{POK}}(\mathbf{c})$ takes as input a challenge $\mathbf{c}$, and returns the corresponding response $\mathbf{r}$. Again,* POK *refers to the unique POK 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 7.2 (Stateless PUF Strong Security). *A stateless PUF is ϵ-secure with error δ if $\Pr[\{\mathbf{c},\mathbf{r}\} \leftarrow \mathsf{Gen}_{\mathrm{POK}}(1^k) : \mathbf{r} = \mathsf{Ver}_{\mathrm{POK}}(\mathbf{c})] > 1-\delta$ and for all PPT A, $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) < \epsilon$, which is defined in terms of the following experiment.*

```
1: procedure $\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A})$
2:     Make polynomial queries to $\mathsf{Gen}_{\mathrm{POK}}(\cdot), \mathsf{Ver}_{\mathrm{POK}}(\cdot)$
3:     if A returns $\{\mathbf{r},\mathbf{c}\}$ such that:
           • $\mathsf{Gen}_{\mathrm{POK}}$ did not return $\{\mathbf{r},\mathbf{c}\}$.
           • $\mathsf{Ver}_{\mathrm{POK}}(\mathbf{c}) = \mathbf{r}$.
4:     then return 1
5:     else return 0.
6: end procedure
```

The $\mathrm{s-uprd}$ *advantage of* A *is defined as*

$$\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) = \Pr[\mathsf{Exp}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) = 1]. \tag{7}$$

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 $\mathsf{Gen}_{\mathrm{POK}}$ and $\mathsf{Ver}_{\mathrm{POK}}$ 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.

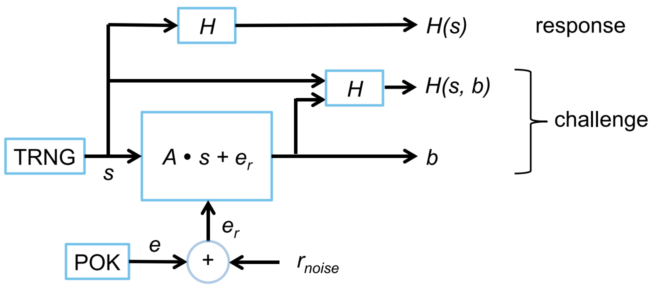
The formalism of manufacturing unclonability remains the same as that put forth in [4].

7.2 Our Construction

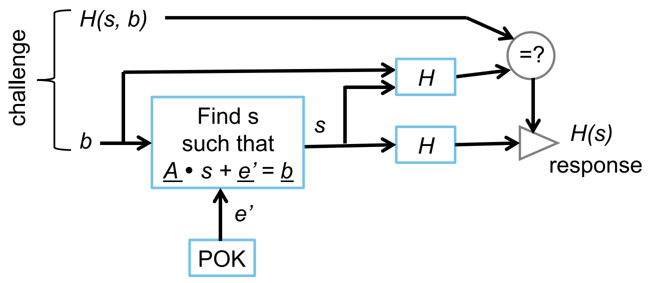
We provide concrete constructions for $\mathsf{Gen}_{\mathrm{POK}}$ and $\mathsf{Ver}_{\mathrm{POK}}$ below, which are also illustrated in Fig. 4.

Construction 7.3 (LPN Stateless PUF). *Let k be a security parameter, with $m, n \in \mathsf{poly}(k)$, and $m > n$. Let $\mathbf{A} \in \{0,1\}^{m\times n}$ be a uniformly random but* **constant** *and publicly known matrix row-indexed by i from 1 to m. Let both algorithms have access to the random oracle $H(\cdot)$.*

```
1: procedure $\{\{\mathbf{b},\mathbf{D_b}\},\mathbf{D_s}\} \leftarrow \mathsf{Gen}_{\mathrm{POK}}(1^k)$
2:     Generate $\mathbf{s} \in \{0,1\}^n$ uniformly at random.
3:     Regenerate $\mathbf{e} \in \{0,1\}^m$ from POK.
4:     Compute $\mathbf{b} = \mathbf{A}\cdot\mathbf{s} + \mathbf{e}$.
5:     return $\{\{\mathbf{b}, H(\mathbf{s},\mathbf{b})\}, H(\mathbf{s})\}$.
6: end procedure
1: procedure $\mathbf{D_s} \leftarrow \mathsf{Ver}_{\mathrm{POK}}(\{\mathbf{b},\mathbf{D_b}\})$
2:     Regenerate $\mathbf{e}', \mathbf{c}'$ from POK.
3:     Run Recovery (Section 4) to extract $\mathbf{s}$ from $\mathbf{b}$.
4:     Verify that $\mathbf{D_b} = H(\mathbf{s},\mathbf{b})$, else return $\perp$.
5:     return $H(\mathbf{s})$.
6: end procedure
```



(a) $\mathsf{Gen}_{\mathrm{POK}}$: Generation of challenge-response pairs. TRNG stands for True Random Number Generator. r_{noise} is random noise injected into low-confidence bits in our variant construction ($\mathsf{Gen\_Noisy}_{\mathrm{POK}}$) and is 0 for our basic construction ($\mathsf{GEN}_{\mathrm{POK}}$).



(b) $\mathsf{Ver}_{\mathrm{POK}}$: 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 $\mathbf{s}$.

Fig. 4. Stateless PUF construction. Note that $\mathsf{Gen}_{\mathrm{POK}}$ and $\mathsf{Ver}_{\mathrm{POK}}$ can be called any number of times in any order. The PUF does not retain any state across invocations.

Note that the above construction requires both internal randomness as well as a random oracle.

7.3 Remarks

7.3.1 Blocking Malicious Challenges

We have included a binding $H(\mathbf{s},\mathbf{b})$ in the challenge-response generation, and let $\mathsf{Ver}_{\mathrm{POK}}$ check if $\mathbf{D_b} = H(\mathbf{s},\mathbf{b})$ before returning a response. This is important, as Definition 7.2 allows for active adversaries. Without this check, an attacker can trivially win the security experiment by returning an output $\mathbf{b}$ by $\mathsf{Gen}_{\mathrm{POK}}$ with one bit modified; the modified bit is likely not used in the recovery of $\mathbf{s}$ at all, and $\mathsf{Ver}_{\mathrm{POK}}$ will accept, trivially violating strong unpredictability. With this check, if $\mathbf{s}$ is recovered correctly, any modification to $\mathbf{b}$ will be detected with overwhelming probability.

7.3.2 Hash Function Requirements

$H(\cdot)$ is a random oracle that is well-approximated by the SHA-256 or SHA-3 hash functions, which we denote $H'(\cdot)$. We require H' to be one-way, since we are exposing $H'(\mathbf{s})$. 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 $H'(\mathbf{s_1} + \Delta\mathbf{s})$ given $H'(\mathbf{s_1})$ and $\Delta\mathbf{s}$. These properties are required because of the use of $H(\mathbf{s},\mathbf{b})$ in the construction.

7.3.3 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 [25] with small modifications are enabled by our construction. 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.

8 Stateless PUF Security Analysis and Assumptions

In the Stateless PUF construction, $\mathsf{Gen}_{\mathrm{POK}}$ is run multiple times with roughly the same noise term $\mathbf{e} = \mathbf{e}_{\mathrm{const}} + \mathbf{e}_{\mathrm{noise}}$. This deviates from the LPN problem, where the noise term for each equation is required to be independent. Therefore, we will need additional assumptions. We begin by showing a reduction from our construction to LPN, assume that the confidence information (i.e., bias of $\mathbf{e}_{\mathrm{noise}}$) is independent of the *actual measurement* of the constant component $\mathbf{e}_{\mathrm{const}}$. This assumption is equivalent to requiring the POK have independent noise. As discussed in Section 3, this assumption is strong, and not necessarily representative of actual POK behavior. Therefore, we then relax this assumption in Section 8.2 on the POK distribution and show that our construction can be reduced to a new conjecture we call Partial-Error-Reuse LPN (PER_LPN, cf. Conjecture 8.4), which says informally that LPN is hard even when part of the error bits are reused.

8.1 Reduction to LPN Assuming Independence Between Confidence and $\mathbf{e}_{\mathrm{const}}$

We start by noting that in order for the construction to reduce to LPN, $\mathsf{Gen}_{\mathrm{POK}}$ must use the same matrix $\mathbf{A}$ on every query to it. Otherwise, an adversary receives two sets of equations with the same $\mathbf{e}$ (we do not want to rely on the small noise $\mathbf{e}_{\mathrm{noise}}$ in POK output for security),

$$\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \mod 2,$$
$$\mathbf{b}' = \mathbf{A}' \cdot \mathbf{s}' + \mathbf{e} \mod 2.$$

The adversary can add up the equations $\bmod 2$, thereby canceling out the $\mathbf{e}$ terms, and trivially recovers both $\mathbf{s}$ and $\mathbf{s}'$. However, we will show in Lemma 8.2 that if the $\mathbf{A}$ matrix is the same for the different secrets, discovering any individual secret requires breaking standard LPN. Intuitively, this means access to $\mathsf{Gen}_{\mathrm{POK}}$ does not help the adversary.

Next we show that access to $\mathsf{Ver}_{\mathrm{POK}}$ does not help an adversary.

Lemma 8.1. *Given an adversary* A *that has non-negligible* $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathrm{PUF}}(\mathsf{A})$, *there exists an algorithm* B *that makes no queries to* $\mathsf{Ver}_{\mathrm{POK}}$ *and still has non-negligible* $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathrm{PUF}}(\mathsf{B})$.

Proof. Let algorithm B run A, simulating calls to $\mathsf{Gen}_{\mathrm{POK}}$, $\mathsf{Ver}_{\mathrm{POK}}$ with the following $\mathsf{Gen}_{\mathsf{B},\mathrm{POK}}$ and $\mathsf{Ver}_{\mathsf{B},\mathrm{POK}}$: Responses of $\mathsf{Gen}_{\mathrm{POK}}$ are faithfully relayed to A after being recorded. Queries to $\mathsf{Ver}_{\mathrm{POK}}$ are simulated by always returning $\perp$ (unless the query is made with an output of $\mathsf{Gen}_{\mathsf{B},\mathrm{POK}}$, in which case the recorded value is returned).

```
1: procedure $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\} \leftarrow \mathsf{Gen}_{\mathsf{B},\mathrm{POK}}(1^k)$
2:     Run $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\} \leftarrow \mathsf{Gen}_{\mathrm{POK}}(1^k)$.
3:     Store $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\}$ to table $T$.
4:     return $\{\{\mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\}$.
5: end procedure
6: procedure $\mathbf{D_s} \leftarrow \mathsf{Ver}_{\mathsf{B},\mathrm{POK}}(\{\mathbf{b}, \mathbf{D_b}\})$
7:     if $\{\mathbf{b}, \mathbf{D_b}\} \in T$ then return $D_s$.
8:     else return $\perp$.
9:     end if
10: end procedure
```

By definition, A generates with non-negligible probability a query for which $\mathsf{Ver}_{\mathrm{POK}}$ would *not* return $\perp$. Therefore, A *can distinguish* $\mathsf{Ver}_{\mathsf{B},\mathrm{POK}}$ from $\mathsf{Ver}_{\mathrm{POK}}$. However, regardless of this fact, A must always emit at least one query to $\mathsf{Ver}_{\mathsf{B},\mathrm{POK}}$ for which $\mathsf{Ver}_{\mathrm{POK}}$ would not return $\perp$.[3]

Given that A makes at most a polynomial number of queries to $\mathsf{Ver}_{\mathsf{B},\mathrm{POK}}$, B may choose any of the queries made by A to $\mathsf{Ver}_{\mathrm{POK}}$ at random and have a non-negligible advantage of returning the "correct" query that would be accepted by $\mathsf{Ver}_{\mathrm{POK}}$. Therefore, B has non-negligible $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathrm{PUF}}(\mathsf{B})$. □

Now we present the security reduction to LPN.

Lemma 8.2. *Let k be a security parameter, $n = \mathsf{poly}(k)$, and $m \geq n$. If Conjecture 2.1 is true, 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 PPT algorithm A has non-negligible advantage in the experiment in Definition 7.2. According to Lemma 8.1, there exists a PPT algorithm B that has non-negligible advantage in the experiment without making queries to $\mathsf{Ver}_{\mathrm{POK}}$. Using B, we will construct an algorithm C that violates the hardness Conjecture 2.1.

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 $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, where $\mathbf{s} \in \{0,1\}^n$ is uniformly random, and $\mathbf{e}$ is chosen according to distribution χ. While in standard LPN, χ represents an i.i.d. distribution of m bits, the reduction here also applies to the correlated LPN in Lemma 6.2.

When B makes calls $H(\cdot)$, C faithfully returns the output of $H(\cdot)$, but records all queries to and responses from $H(\cdot)$. When B makes calls to $\mathsf{Gen}_{\mathrm{POK}}$, C responds using the following simulated version $\mathsf{Gen}_{\mathsf{C},\mathrm{POK}}$:

```
1: procedure $\{\{\mathbf{b}', \mathbf{D}'_\mathbf{b}\}, \mathbf{D}'_\mathbf{s}\} \leftarrow \mathsf{Gen}_{\mathsf{C},\mathrm{POK}}(1^k)$
2:     Generate uniformly random $\Delta\mathbf{s}$.
3:     $\mathbf{b}' = \mathbf{b} + \mathbf{A} \cdot \Delta\mathbf{s} + \mathbf{e}_{\mathrm{noise}} = \mathbf{A}(\mathbf{s} + \Delta\mathbf{s}) + \mathbf{e} + \mathbf{e}_{\mathrm{noise}}$
4:     Uniformly generate $U_1,\ U_2 \in \{0,1\}^l$.
5:     Insert $\{\{\mathbf{b}', U_1\}, U_2\}$ into a local table $T$.
6:     return $\{\{\mathbf{b}', U_1\}, U_2\}$.
7: end procedure
```

The output $\mathbf{b}'$ by $\mathsf{Gen}_{\mathsf{C},\mathrm{POK}}$ corresponds to the LPN problem with the random secret $(\mathbf{s} + \Delta\mathbf{s})$, and is indistinguishable from the output of $\mathsf{Gen}_{\mathrm{POK}}$. Note that the

3. After this query, A may have "distinguished" that it is querying $\mathsf{Ver}_{\mathsf{B},\mathrm{POK}}$ instead of $\mathsf{Ver}_{\mathrm{POK}}$, so the behavior of A is undefined.

added $\mathbf{e}_{\text{noise}}$ models the noisy POK output.[4] Assume $\mathbf{e}_{\text{noise}}$ does not depend on the constant component of the noise term ($\mathbf{e}$ here), so C can sample the confidence information from $N(0, \sigma_{\mathsf{INTER}})$, and then sample $\mathbf{e}_{\text{noise}}$ from $N(c, \sigma_{\mathsf{INTRA}})$ on its own according to the distributions in Fig. 3. Note that this implies that the POK noise is i.i.d.

Furthermore, since $H(\cdot)$ is a random oracle, the output U_1, U_2 by $\mathsf{Gen}_{\mathsf{C},\text{POK}}$ are computationally indistinguishable from $\mathbf{D_b}, \mathbf{D_s}$ by $\mathsf{Gen}_{\text{POK}}$. Therefore, C precisely mimics the behavior of $\mathsf{Gen}_{\text{POK}}$ for B, and with non-negligible probability, B outputs $\{\{\mathbf{b}', \mathbf{D}'_\mathbf{b}\}, \mathbf{D}'_\mathbf{s}\}$ that is not in table T and makes $\mathsf{Ver}_{\text{POK}}$ accept

$$\mathbf{b}' = \mathbf{A} \cdot \mathbf{s}' + \mathbf{e}'$$
$$\mathbf{D}'_\mathbf{b} = H(\mathbf{s}', \mathbf{b}')$$
$$\mathbf{D}'_\mathbf{s} = H(\mathbf{s}').$$

Since $H(\cdot)$ is a random oracle, B must have queried $H(\cdot)$ with $\mathbf{s}'$ before; otherwise, the probability of $\mathbf{D}'_\mathbf{s} = H(\mathbf{s}')$ must be negligible. C has recorded all the queries to and responses from $H(\cdot)$, and thus can retrieve $\mathbf{s}'$ and compute $\mathbf{e}'$.

In order for $\mathsf{Ver}_{\text{POK}}$ to accept B's output, $\mathbf{e}'$ must be distributed according to the confidence information C sampled. C can then recover $\mathbf{e}$ in the same way Recovery does, and then solve for $\mathbf{s}$. If $\mathsf{Ver}_{\text{POK}}$ accepts B's output with non-negligible probability, then C recovers $\mathbf{e}$ and $\mathbf{s}$ with non-negligible probability. This contradicts Conjecture 2.1. □

8.2 Reduction to PER_LPN

The above security proof requires the confidence information and $\mathbf{e}_{\text{noise}}$ be independent of $\mathbf{e}_{\text{const}}$. Consider a joint distribution for m bits, where a subset of m' bits are always 0, and the remaining $m - m'$ bits are i.i.d. These m' bits correspond to the set of stable bits T from Section 5, and there is perfect correlation between the confidence of a bit and its value. In this case, the above reduction cannot hold, since $\mathbf{e}_{\text{noise}}$ is not independent of $\mathbf{e}$. In this section, we relax the above requirement to allow dependence between $\mathbf{e}_{\text{noise}}$ and $\mathbf{e}_{\text{const}}$.

Let us use the intuition from Section 5 that $\mathsf{Gen}_{\text{POK}}$ and $\mathsf{Ver}_{\text{POK}}$ are able to detect a bit's confidence information. Specifically, let us abstract the notion of "stability" and require that a POK be of the form in Definition 8.3.

Definition 8.3. *A "(ϵ_2)-threshold POK" is a function $\{S_0, T_0, S_1, T_1\} \leftarrow \text{POK}$ such that $|\mathbf{e}| = m$, $T_0 \subset S_0 \subset [m]$, $T_1 \subset S_1 \subset [m]$ with the following properties:*

- *There exist disjoint sets S_0, S_1 with $S_0 \cup S_1 = [m]$ that may be different upon each measurement of POK.*
- *There exist subsets $T_0 \subset S_0$, $T_1 \subset S_1$ such that $\mathsf{Pr}(i \in S'_1 | i \in T_0) < \epsilon_2$ and $\mathsf{Pr}(i \in S'_0 | i \in T_1) < \epsilon_2$. Define S'_1 and S'_0 respectively as the sets S_1 and S_0 during a* different *measurement of POK.*

Finally, require that $|T_0| = \theta(m)$ and $|T_1| = \theta(m)$, and $\epsilon_2 \in \mathsf{neg}(k)$.[5]

In previous sections of this paper, we set bits in S_0 to '0' and bits in S_1 to '1'. T_0 and T_1 were then "stable '0'"; and "stable '1'" respectively. Instead, consider that for each measurement of POK, bits in S_0 are assigned to '0', and bits in S_1 are assigned to *uniformly random values* (cf. GEN_Noisy in Algorithm 2). The set T_1 now corresponds to bits that are uniformly random with high probability on each measurement. This is illustrated in Fig. 4a using r_{noise}.

Algorithm 2. Noisy Generate and Verify

```
1: procedure $\{\{\mathbf{A}, \mathbf{b}, \mathbf{D_b}\}, \mathbf{D_s}\} \leftarrow \mathsf{Gen\_Noisy}_{\text{POK}}(1^k)$
2:     for $i$ from 1 to $L$ do
3:         Query $\{S_0, T_0, S_1, T_1\} \leftarrow \text{POK}$.
4:         Generate $\mathbf{s}^i \in \{0,1\}^n, A^i \in \{0,1\}^{m \times n}$ uniformly at random.
5:         Set $\mathbf{e}^i_j$ to '0' for all $j \in S_0$.
6:         Set $\mathbf{e}^i_j$ to uniform random $\{0,1\}$ for all $j \in S_1$.
7:         Compute $\mathbf{b}^i = \mathbf{A}^i \cdot \mathbf{s}^i + \mathbf{e}^i$.
8:         Store $\{\{\mathbf{A}^i, \mathbf{b}^i, H(\mathbf{s}^i, \mathbf{A}^i, \mathbf{b}^i)\}, H(\mathbf{s}^i)\}$ into Tab.
9:     end for
10:    return Tab.
11: end procedure
1: procedure $\mathbf{D_s} \leftarrow \mathsf{Ver\_Noisy}_{\text{POK}}$ (Tab)
2:     Set $L = \text{Length(Tab)}$.
3:     Initialize $\text{ErrSum} = \{0\}^m$.
4:     Query $\{S_0, T_0, S_1, T_1\} \leftarrow \text{POK}$.
5:     for each $\{\mathbf{A}, \mathbf{b}, \mathbf{D_b}\}$ in Tab do
6:         Run Recovery (Section 4) to extract $\mathbf{s}$ from $\mathbf{b}$ using $T_0$.
7:         if $\mathbf{D_b} \neq H(\mathbf{s}, \mathbf{A}, \mathbf{b})$ then
8:             return $\perp$.
9:         end if
10:        Add $H(\mathbf{s})$ to HsTab.
11:        $\mathbf{e_{meas}} = \mathbf{A} \cdot \mathbf{s} - \mathbf{b} \bmod 2$.
12:        $\text{ErrSum} = \text{ErrSum} + \mathbf{e_{meas}}$.
13:    end for
14:    Set $\text{PrErr}_i = 1/2 - |1/2 - \text{ErrSum}_i / L|$.
15:    Set $\hat{T}_0$ to be the set of indices corresponding to the $m'$
       minimum $\text{PrErr}_i$.
16:    Run Recovery on $\hat{T}_0$ to recover $\hat{\mathbf{s}}$.
17:    if $\hat{\mathbf{s}} \neq \mathbf{s}$ then
18:        return $\perp$.
19:    else
20:        return HsTab.
21:    end if
22: end procedure
```

Consider $\mathsf{Gen\_Noisy}_{\text{POK}}$ and $\mathsf{Ver\_Noisy}_{\text{POK}}$, modified according to the discussion above. For technical reasons pertaining the proof of Lemma 8.5, we require that Gen_Noisy and Ver_Noisy generate/verify a polynomial number of $\{\{\mathbf{A}, \mathbf{b}, H(\mathbf{s}, \mathbf{A}, \mathbf{b})\}, H(\mathbf{s})\}$. The reason for this will become apparent in the proof. Further, we modify $\mathsf{Ver\_Noisy}_{\text{POK}}$ to check that the stable bits of the POK (the set T_0) are not too different from the bits that are stable in the provided samples (computed as $\hat{T}_0$, cf. lines 14-16 of

4. A POK with i.i.d. noise (assumed in this reduction) is modeled by a constant set of bits ($\mathbf{e}$ in the algorithm) plus some i.i.d. "noise" ($\mathbf{e}_{\text{noise}}$ in the algorithm) with some Bernoulli parameter τ. Therefore, the summation of $\mathbf{e} + \mathbf{e}_{\text{noise}}$ accurately models if the Bernoulli parameter of $\mathbf{e}_{\text{noise}}$ is τ.

5. ϵ_2 has the same interpretation as in Section 5. However, in Section 5, $\epsilon_2 = 1/n$. This is not sufficient for this proof, and we must set $\epsilon_2 = \mathsf{neg}(k)$.

Algorithm 2). Namely, Recovery must succeed when using either T_0 or $\hat{T}_0$.

With this modification, many of the bits are thrown away, so we can ignore their distribution. We require only that a large enough subset of the bits are replaced with random bits. We conjecture the following modified LPN problem to be hard, and name it "Partial Error Reuse LPN" problem, or PER_LPN.

Conjecture 8.4 (PER_LPN$_{n,m,u,L}$). Consider L LPN problems $\{\mathbf{b}^j = \mathbf{A}^j \cdot \mathbf{s}^j + \mathbf{e}^j\}_{j=1}^{L}$, where L is polynomial in n. Let $\mathbf{s}^j \in \{0,1\}^n$, $\mathbf{e}^j \in \{0,1\}^m$, $\mathbf{A}^j \in \{0,1\}^{m\times n}$, and let $\{\mathbf{e}^j\}_j$ follow the joint distribution χ_U below:

1) Randomly select $U \subset [m]$ of size u;
2) Select $\mathbf{e}_i^j$ with $i \notin U$ according to some joint distribution χ for each j.
3) Select $\mathbf{e}_i^j$ with $i \in U$ uniformly from $\{0,1\}$ for each j.

There does not exist a PPT algorithm for any χ that finds all s^j in Poly(n,u).

It is important to note that a polynomial number of outputs from Gen_Noisy concatenated is a PER_LPN problem with overwhelming probability. The bits in T_1 for a certain measurement will remain in S_1 across measurements and are thus made i.i.d. uniform random except with $\epsilon_2 \in \mathsf{neg}(k)$ probability. This will be the set U in the PER_LPN conjecture. The remaining bits can be arbitrarily distributed according to the conjecture.

We now are ready to show that **PUF_Noisy** $= \{$Gen_Noisy, Ver_Noisy$\}$ comprises a Stateless PUF according to Definition 7.1.

Lemma 8.5. *Let k be a security parameter, $n = \mathsf{poly}(k)$, and $m \geq n$. If Conjecture 8.4 is true, there is no PPT* A *that has advantage* $\mathsf{Adv}^{\text{s-uprd}}_{\textbf{PUF\_Noisy}}(\mathsf{A})$ *non-negligible in k.*

Proof. Given A, we construct B that takes a PER_LPN problem as an argument (PerLPNTab) and returns the secret vectors of *all* members in this set.

Whenever A queries Gen_Noisy, B answers with a batch of size L in PerLPNTab (so we write PerLPNTab as A's input in Line 2). Since A makes a polynomial number of queries to Gen_Noisy, PerLPNTab has polynomial instances.

Recognize that Lemma 8.1 still applies, so we do not need to give A access to Ver_Noisy. B also records A's calls to $H(\cdot)$ (and can therefore extract each $\mathbf{s}$ from each $\{H(\mathbf{s},\mathbf{A},\mathbf{b}), H(\mathbf{s})\}$ returned by A).

As mentioned earlier, there exists a PerLPNTab that precisely mimics outputs of Gen_Noisy. Therefore, A finally produces Tab that will make Ver_Noisy accept with non-negligible probability. Similar to the proof of Lemma 8.2, B can then use the recorded queries to $H(\cdot)$ to recover the error vectors in A's output.

Next, B recovers the secret subset $\hat{T}_0 \subset [m]$, i.e., the stable bits in PerLPNTab, by looking at the distribution of these errors vectors in A's output and estimating $\Pr(\mathbf{e'}_i = 1)$ (Line 9,10 in Algorithm 3). This explains why the protocol was modified earlier to incorporate a polynomial number of challenge/response pairs: they are needed in this reduction to accurately characterize the distribution of each bit. Since A makes Ver_Noisy accept with non-negligible probability, Recovery when called with $\hat{T}_0$ will succeed with non-negligible probability, in which case B solves each instance in PerLPNTab. This contradicts Conjecture 8.4. □

Algorithm 3. PER_LPN Reduction Algorithm B

```
1: procedure B (PerLPNTab)
2:    Tab ← A(PerLPNTab)
3:    Initialize ErrSum = $0^m$.
4:    for each $\{\mathbf{A}', \mathbf{b}'\} \in$ Tab do
5:       Find $\mathbf{s}'$ from recorded queries by A to $H(\cdot)$.
6:       Compute $\mathbf{e}' = \mathbf{b}' - \mathbf{A}' \cdot \mathbf{s}' \bmod 2$
7:       ErrSum = ErrSum + $\mathbf{e}'$.
8:    end for
9:    Set $\mathrm{PrErr}_i = 1/2 - |1/2 - \mathrm{ErrSum}_i/L|$.
10:   Set $\hat{T}_0$ to be the set of indices corresponding to the $m'$
         minimum $\mathrm{PrErr}_i$.
11:   Run Recovery on each instance in PerLPNTab using $\hat{T}_0$
         as the set of stable bits, and return the solutions.
12: end procedure
```

Lemma remarks. The above lemma proves *security* of the scheme. This proof does not directly rely on the distribution of the bits in **e**. However, the *correctness* of the protocol does rely on the distribution. I.e., if $\hat{T}_0$ does not approximate T_0 (and therefore, Recovery fails when called with $\hat{T}_0$), then Ver_Noisy does not accept with high probability.

In the case of an i.i.d. Gaussian distribution of **e** (as in Section 8.1), one may prove correctness by showing explicitly that $\hat{T}_0$ will with high probability cause Recovery to succeed. We omit this proof here, as it is straightforward.

Further, in the case of the Gaussian distribution (cf. Section 5), we may set $\epsilon_2 = \mathsf{neg}(k)$. For example, set $\epsilon_2 = 2^{-\Theta(\log^2(m))} = \mathsf{neg}(k)$. From Section 5.4, this implies $\sigma_r = \Theta(1/\log(n))$, and therefore the number of correctable errors is $O(m/\log(m)) = \widetilde{O}(m)$. Further, from Section 5.4, the number of stable bits (i.e., $|T_1|$) is $m' = \theta(m)$. This meets Definition 8.3.

Now, it may at first seem as though this approach requires the same assumption of i.i.d. POK bits as in Section 8! This is technically true if one desires a mathematical proof of correctness. However, the key difference is that both of the above requirements are *empirically verifiable for unknown distributions*.

To see the importance of this distinction, we first point out that all POKs that the authors are aware of have unknown distributions, as they are derived from physical systems with complex internal behavior, which is subsequently affected by environmental parameters and noise. There has been significant effort (especially in the silicon POK case [27], [57], [42], [40], [49]) to make the POK distribution as close to i.i.d. as possible. However, it is not possible to *prove* through empirical measurements that a POK distribution is i.i.d.

Any mathematical proof of correctness must assume a property of the POK distribution, and therefore will ultimately have to be empirically verified. For example, information theoretic approaches typically have a min-entropy requirement on the distribution of the POK bits. This requirement is empirically verified through measurement of large samples of POKs [48].

In the case of the above stateless PUF construction, we require first that $|T_1| = \theta(m)$, $\epsilon_2 = \mathsf{neg}(k)$. This can be verified by studying the stability of bits. For example, in Section 9, we provide evidence that the Gaussian distribution of bits is correct, and therefore that this is indeed the case. Further, in Section 9, we observe that 76 percent of the ring oscillator bits *do not flip* across all measurement parameters. Therefore, we do not believe that the above requirement is unreasonable.

Second, we require that $\hat{T_0} \approx T_0$, so that $\mathsf{Ver\_Noisy}$ accepts with high probability. Again, data in Section 9 show that the independent Gaussian model reasonably approximates the behavior of the ring oscillator POK, so $\mathsf{Ver\_Noisy}$ will accept with high probability. However, we note that even if the POK distribution differs slightly from Gaussian, such a difference only affects extraction efficiency, not the security of the construction. Ultimately, in the practical setting, the true extraction efficiency will be measured and optimized empirically for a given POK architecture.

8.3 Stateless PUF Theorem

We are now ready to state the security theorem for the stateless PUF constructions. For i.i.d. POK outputs (Theorem 8.6), the theorem holds under Conjecture 2.1. For more complex distributions (Theorem 8.7), we use Conjecture 8.4.

Theorem 8.6. *Let k be a security parameter, and $n = \mathsf{poly}(k)$. There exists a choice of n, $m \geq n$, χ such that Construction 7.3 is a (m, χ) stateless PUF that is ϵ-secure with error δ, with $\epsilon = \mathsf{neg}(k)$, $\delta = \mathsf{neg}(k)$ under Conjecture 2.1.*

Proof. First, recognize that Construction 7.3 is efficient. Clearly, $\mathsf{Gen}_{\mathrm{POK}}$ runs in polynomial time. Section 5.2 shows $\mathsf{Ver}_{\mathrm{POK}}$ runs in polynomial time. Second, under Conjecture 2.1, there does not exist any PPT A that gains advantage $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) > \epsilon$, where $\epsilon = \mathsf{neg}(k)$. □

Theorem 8.7. *If POK makes $\mathsf{Ver\_Noisy}$ accept with probability δ where $\delta = \mathsf{neg}(k)$ and has distribution χ obeying Definition 8.3, then there exists a choice of n, $m \geq n$, such that $\{\mathsf{Gen\_Noisy}_{\mathrm{POK}}, \mathsf{Ver\_Noisy}_{\mathrm{POK}}\}$ is a (m, χ) stateless PUF that is ϵ-secure with error δ, with $\epsilon = \mathsf{neg}(k)$, $\delta = \mathsf{neg}(k)$ under Conjecture 8.4.*

Proof. First recognize that the construction is efficient. Second, Lemma 8.5 shows that if Conjecture 8.4 is true, then there does not exist a PPT A that gains advantage $\mathsf{Adv}^{\mathrm{s-uprd}}_{\mathbf{PUF}}(\mathsf{A}) > \epsilon$, where $\epsilon = \mathsf{neg}(k)$. □

9 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. Fig. 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 2.1.

We have provided a theory explaining the resilience of the LPN construction to noise and environmental parameters using this confidence information in Section 5. 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) range of temperature and voltage. Three interesting points are $-40\,^\circ$C@0.95V, $25\,^\circ$C@1.00V, and $105\,^\circ$C@1.05V. Other ranges that we will use are the differential count values at commercial (0 to 70°C) and extended industrial (-40 to 85°C). The $\sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}}$ ratios improve as the temperature range is reduced.

TABLE 2
(Left) Measured Bias of 320 RO Pairs at Varying Temperatures. (Right) Measured σ_{INTRA} for Varying Temperatures

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

We note that 24 percent 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.

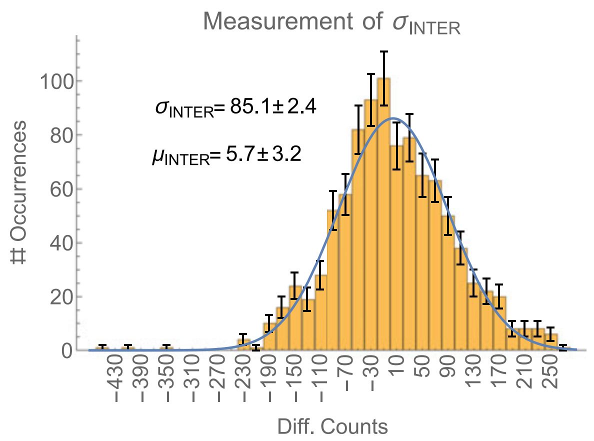
We first measured the bias of the RO counts across temperature as shown in Table 2. Therefore, our pessimistic estimate of bias ignoring correlation effects as 45 percent (or 55 percent equivalently) is correct.

These differential count values are distributed according to the distribution discussed in Section 5 with variance $\sigma^2_{\mathrm{INTER}}$. We verified for each of these temperatures that the distribution of differential counts was Gaussian, as we assumed in Section 5. Each of the fits from which parameters are derived has a reduced $\chi^2 \approx 1$, indicating that the Gaussian model is a good fit to the data within experimental error. 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 Fig. 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}.

To accomplish this, we used room temperature as a baseline (this would be the condition 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 5).

The distribution at $105\,^\circ$C is shown in Fig. 5. The measurements at various temperatures are shown in Table 2. It is



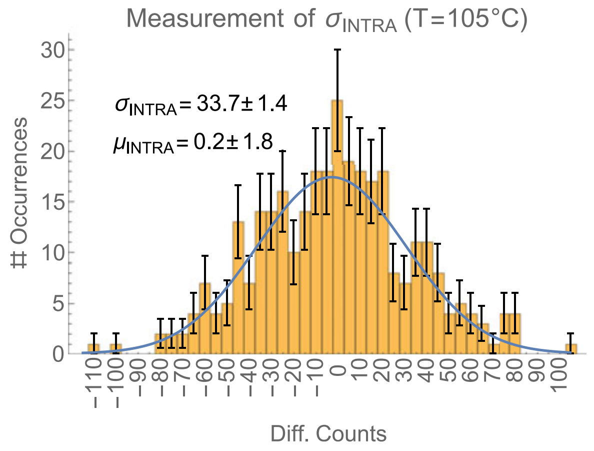


Fig. 5. (Top) Measurement of σ_{INTER} through the estimation of the distribution of differential counts across 320 RO pairs across room temperature and the fast and slow voltage/temperature corners. (Bottom) Measurement of σ_{INTRA} by subtracting differential counts at $25\,^{\circ}\text{C}@1\text{V}$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

important to note that although in Section 5 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 Fig. 5. Using these measurements, we calculate the ratio $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ for commercial (0 to 70°C) as 0.20, extended industrial (-40 to 85°C) as 0.29, and the maximum temperature range our experiment could support (-40 to 105°C) as 0.40. This is summarized in Table 3.

We now present an analysis of the resource requirements (number of RO pairs) of our LPN fuzzy extractor scheme with a security parameter of 128, and probability of error 10^{-6} over the above temperature ranges.

First, we remark that our theoretical construction in Section 4 is far too conservative for practical purposes. In practice, we simply choose the most stable $m' = n$ bits, and most likely there are at most $t' \leq 1$ error bits in them. For example, if $\epsilon_2 = 3 \times 10^{-6}$, a simple binomial distribution analysis shows that $\Pr(t' > 1) < 10^{-6}$. Therefore, an exhaustive search over the error bit with a Gaussian elimination operations for each will suffice.

Plugging $\epsilon_1 = 10^{-6}$, $\epsilon_2 = 3 \times 10^{-6}$, $m' = n = 128$ (giving a security parameter of 128) and $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ values into Equations (3), (4), we compute m (the total number of RO pairs) for various temperature ranges, also shown in Table 3.

TABLE 3
Summary of $\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$ and Resources Required for an LPN Fuzzy Extractor Over the Specified Temperature Range

Temp.	Erroneous Bits	$\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$	# Ring Osc. Pairs ($= m$)
$0 - 70^{\circ}\text{C}$	9%	0.20	450
$-40 - 85^{\circ}\text{C}$	21%	0.29	770
$-40 - 105^{\circ}\text{C}$	24%	0.40	1870

The percentage of erroneous bits over environmental conditions and associated ratio is displayed. Extraction succeeds with error probability $< 10^{-6}$ and a security parameter of 128.

Note that the bits of the extracted bitstring are *not all simultaneously pseudorandom* with security parameter 128. In order to obtain a pseudorandom bitstring for use as a key, one must use a hash function that approximates a random oracle. To avoid the use of such a function, one may also double the LPN secret size to $n = 256$ and then select an arbitrary subset of 128 bits. These 128 bits would be pseudorandom by the result from [1].

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 very well with the works described in Section 3. For example, PUFKY [48] requires 2052 helper data bits for a 10—80°C temperature range, compared to our 450 helper data bits for a comparable 0—70°C temperature range, and 770 helper data bits for a much wider temperature range -40—85°C. Moreover, unlike most prior work on information theoretic extractors, our LPN fuzzy extractor can be scaled to higher noise settings simply by increasing m without affecting the security argument.

10 Conclusion

We have presented a computationally secure construction of a stateless Physical Unclonable Function in this paper based on precise hardness assumptions. This has been an open problem for over thirteen years since silicon PUFs were introduced in 2002 [26].

Our construction is secure in the random oracle model under the difficulty of standard Learning Parity with Noise and a variant LPN problem. 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 [25] are enabled by our construction.

In order to construct a noise-free PUF, we presented the first construction of a computational fuzzy extractor with a trapdoor in this paper, using the standard LPN problem as the hard problem. The trapdoor allows our construction to correct $\Theta(m)$ errors in polynomial time. We relaxed the i.i.d. assumptions on the POK outputs showing that if correlation can be estimated, the only change to the fuzzy extractor construction is in the selection of parameters.

We show how error profiles obtained from a Field Programmable Gate Array implementation of PUFs subject to wide environmental variation can be efficiently corrected using helper data sizes that are substantially smaller than prior art.

Lastly, we remark that certain human biometrics may also produce confidence information. Ongoing work includes testing our constructions on biometrics.

ACKNOWLEDGEMENTS

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] A. Akavia, S. Goldwasser, and V. Vaikuntanathan, “Simultaneous hardcore bits and cryptography against memory attacks,” in *Proc. 6th Int. Conf. Theory Cryptography*, 2009, pp. 474–495.

[2] B. Applebaum, B. Barak, and A. Wigderson, “Public-key cryptography from different assumptions,” in *Proc. 42nd Symp. Theory Comput.*, 2010, pp. 171–180.

[3] B. Applebaum, D. Cash, C. Peikert, and A. Sahai, “Fast cryptographic primitives and circular-secure encryption based on hard learning problems,” in *Proc. 29th Annu. Int. Cryptol. Conf.*, 2009, pp. 595–618.

[4] F. Armknecht, R. Maes, A. Sadeghi, O.-X. Standaert, and C. Wachsmann, “A formalization of the security features of physical functions,” in *Proc. IEEE Symp. Security Privacy*, 2011, pp. 397–412.

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

[6] G. T. Becker, “The gap between promise and reality: On the insecurity of XOR arbiter PUFs,” in *Proc. 17th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2015, pp. 535-555

[7] G. T. Becker, A. Wild, and T. Güneysu, “Security analysis of index-based syndrome coding for PUF-based key generation,” in *Proc. IEEE Int. Symp. Hardware Oriented Security Trust*, May 2015, pp. 20–25.

[8] D. J. Bernstein and T. Lange, “Never trust a bunny,” in *Radio Frequency Identification Security and Privacy Issues*, New York, NY, USA: Springer, 2013, pp. 137–148.

[9] A. Blum, M. Furst, Kearns, M., and R. Lipton, “Cryptographic primitives based on hard learning problems,” in *Proc. 13th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 1994, pp. 278–291.

[10] A. Blum, A. Kalai, and H. Wasserman, “Noise-tolerant learning, the parity problem, and the statistical query model,” *J. ACM* , vol. 50, no. 4, pp. 506–519, 2003.

[11] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, “SPONGENT: A lightweight hash function,” in *Proc. 13th Int. Workshop Cryptographic Hardware Embedded Syst.,*, 2011, pp. 312–325.

[12] S. Bogos, F. Tramer, and S. Vaudenay, “On solving LPN using BKW and variants,” International Association for Cryptologic Research, Tech. Rep., Report 2015/049, 2015.

[13] K. W. Bowyer, K. Hollingsworth, and P. J. Flynn, “Image understanding for iris biometrics: A survey,” *Comput. Vis. Image Understanding*, vol. 110, no. 2, pp. 281–307, 2008.

[14] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, “Secure remote authentication using biometric data,” in *Proc. 24th Annu. Int. Conf. Adv. Cryptology*, 2005, pp. 147–163.

[15] J. Bringer, H. Chabanne, G. Cohen, B. Kindarji, and G. Zémor, “Optimal iris fuzzy sketches,” in *Proc. 1st IEEE Int. Conf. Biometrics: Theory, Appl. Syst.*, 2007, pp. 1–6.

[16] M. Chiani and D. Dardari, “Improved exponential bounds and approximation for the q-function with application to average error probability computation,” in *Proc. Global Telecommun. Conf.*, 2002, vol. 2, pp. 1399–1402.

[17] I. Damgård and S. Park, “Is public-key encryption based on LPN practical?,” in *Proc. IACR Cryptology ePrint Archive*, 2012, p. 699.

[18] J. Delvaux and I. Verbauwhede, “Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise,” in *Proc. 6th IEEE Int. Symp. Hardware-Oriented Security Trust*, 2013, pp. 137–142.

[19] J. Delvaux and I. Verbauwhede, “Attacking PUF-based pattern matching key generators via helper data manipulation,” in *Proc. Int. Conf. Topics Cryptol.*, 2014, pp. 106–131.

[20] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith, “Robust fuzzy extractors and authenticated key agreement from close secrets,” *IEEE Trans. Inform. Theory*, vol. 58, no. 9, pp. 6207–6222, Sep. 2012.

[21] Y. Dodis, L. Reyzin, and A. Smith, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data,” In *Proc. Int. Conf. Adv. Cryptology*, 2004, pp. 523–540.

[22] B. Fuller, X. Meng, and L. Reyzin, “Computational fuzzy extractors,” In *Proc. 19th Int. Conf. Theory Appl. Cryptology Inform. Security*, 2013, pp. 174–193.

[23] M. Gao, K. Lai, and G. Qu, “A highly flexible ring oscillator PUF,” in *Proc. 51st Annu. Int. Conf. Des. Autom.*. 2014, pp. 89:1–89:6.

[24] B. Gassend, “Physical random functions,” M.S. thesis, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Jan. 2003.

[25] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Controlled physical random functions,” presented at the 18th Annu. Computer Security Applications Conf., Silver Spring, MD, USA, Dec. 2002.

[26] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in *Proc. 9th ACM Int. Conf. Comput. Commun. Security*, 2002, pp. 148–160.

[27] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, “Delay-based circuit authentication and applications,” in *Proc. ACM Symp. Appl. Comput.*, Mar. 2003, pp. 294–301..

[28] S. Graybeal, and P. McFate, “Getting out of the STARTing block,” *Scientific American*, vol. 261, no. 6, pp. 64–65, 1989.

[29] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, “Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain,” *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[30] Q. Guo, T. Johansson, and C. Löndahl, “Solving LPN using covering codes,” in *Proc. 20th Int. Conf. Theory Appl. Cryptology Inform. Security.*, 2014, pp. 1–20.

[31] M. Hiller, D. Merli, F. Stumpf, and G. Sigl, “Complementary IBS: Application specific error correction for PUFs,” in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2012, pp. 1–6.

[32] M. Hiller, M. Weiner, L. Rodrigues Lima, M. Birkner, and G. Sigl, “Breaking through fixed PUF block limitations with differential sequence coding and convolutional codes,” in *Proc. 3rd Int. Workshop Trustworthy Embedded Devices*, 2013, pp. 43–54.

[33] D. Holcomb, W. Burleson, and K. Fu, “Power-up SRAM state as an identifying fingerprint and source of true random numbers,” *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[34] N. J. Hopper and M. Blum, “Secure human identification protocols,” in *Proc. 7th Int. Conf. Adv. Cryptol.*, 2001, pp. 52–66.

[35] G. Hospodar, R. Maes, and I. Verbauwhede, “Machine learning attacks on 65 nm arbiter PUFs: Accurate modeling poses strict bounds on usability,” in *Proc. 4th IEEE Int. Workshop Inform. Forensics Security*, 2012, pp. 37–42.

[36] D. Karakoyunlu, and B. Sunar, “Differential template attacks on PUF enabled cryptographic devices,” in *Proc. IEEE Int. Workshop Inform. Forensics Security*, Dec. 2010, pp. 1–6.

[37] D. Kirovski, “Anti-counterfeiting: Mixing the physical and the digital world,” in *Towards Hardware-Intrinsic Security*, A.-R. Sadeghi and D. Naccache, Eds. New York, NY, USA: Springer, 2010, pp. 223–233.

[38] P. Koeberl, J. Li, A. Rajan, and W. Wu, “Entropy loss in PUF-based key generation schemes: The repetition code pitfall,” in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, May 2014, pp. 44–49.

[39] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-Hashing for message authentication,” United States, 1997.

[40] R. Kumar, and W. Burleson, “On design of a highly secure PUF based on non-linear current mirrors,” in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2014, pp. 38–43.

[41] É. Levieil, and P.-A. Fouque, “An improved LPN algorithm,” in *Proc. 5th Int. Conf. Security Cryptography Networks*, 2006, pp. 348–359.

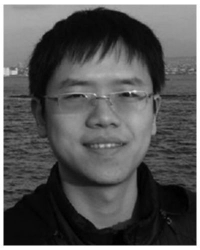
[42] D. Lim, “Extracting secret keys from integrated circuits,” M. S. thesis, Dept. Electrical Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, May 2004.

[43] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “Extracting secret keys from integrated circuits,” *IEEE Trans. VLSI Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[44] K. Lofstrom, W. R. Daasch, and D. Taylor, “IC identification circuit using device mismatch,” in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2000, pp. 372–373.

[45] V. Lyubashevsky, "The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem," in *Proc. 8th Int. Workshop Approximation, Randomization Combinatorial Optimization Algorithms Techn.*, 2005, pp. 378–389.
[46] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs," in *Proc. 11th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2009, pp. 332–347.
[47] R. Maes, P. Tuyls, and I. Verbauwhede, "Soft decision helper data algorithm for SRAM PUFs," in *Proc. IEEE Int. Conf. Symp. Inform. Theory*, 2009, pp. 2101–2105.
[48] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *Proc. 14th Int. Conf. Cryptographic Hardware Embedded Syst.*, 2012, pp. 302–319.
[49] M. Orshansky, "Physically unclonable functions based on non-linearity of sub-threshold operation," US Patent 8,938,069, 2015.
[50] R. Pappu, "Physical one-way functions," Ph.D. thesis, Massachusetts Inst. Technol., Cambridge, MA, USA, 2001.
[51] Z. Paral, and S. Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2011, pp. 128–133.
[52] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, p. 34, 2009.
[53] U. Rührmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder," in *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. Berlin, Germany: Springer, 2012, ch. 4, pp. 65–102.
[54] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 237–249.
[55] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. Inform. Forensics Security*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013.
[56] G. E. Suh, "AEGIS: A single-chip secure processor," Ph.D. thesis, Dept. Electrical Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Aug. 2005.
[57] G. E. Suh, and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 4th ACM/IEEE Design Autom. Conf.*, 2007, pp. 9–14.
[58] J. Tobisch, and G. T. Becker, "On the scaling of machine learning attacks on PUFs with application to noise bifurcation," in *Proc. RFIDSec*, 2015, pp. 17–31.
[59] M.-D. M. Yu, and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 48–65, Jan./Feb. 2010.
[60] M.-D. M. Yu, D. M"Raïhi, R. Sowell, and S. Devadas, "Lightweight and secure PUF key storage using limits of machine learning," in *Proc. 13th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2011, pp. 358–373.
[61] M. M. Yu, M. Hiller, and S. Devadas, "Maximum-likelihood decoding of device-specific multi-bit symbols for reliable key generation," in *Proc. IEEE Int. Symp. Hardware Oriented Security Trust,*, 2015, pp. 38–43.

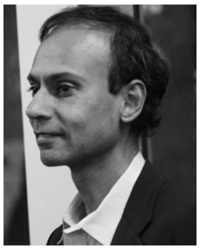
Charles H. Herder III recieved the master's degree in electrical engineering and computer science, the BS degree in EECS, and the BS degree in physics all from the Massachusetts Institute of Technology. His research interests include the physics of computation, cryptography, computer architecture, and computer security. His prior experience at Texas Instruments includes developing embedded systems authentication technology and serving as a technical lead for the development of proprietary power management and authentication systems.



Ling Ren recieved the bachelor's degree in electrical engineering from Tsinghua University, China, and the master's degree in electrical engineering and computer science from the Massachusetts Institute of Technology. His research interests include the computer security, applied cryptography and computer architecture. While at MIT, he has worked on secure processors and oblivious random access memory.

Marten van Dijk received the MS degree in computer science, the MS and PhD degrees in mathematics from the Eindhoven University of Technology. He is an associate professor at the University of Connecticut. He joined UConn in 2013. Prior to joining UConn, he worked at MIT CSAIL, RSA, and Philips Research. His research interests include computer security and cryptography.

Mandel Yu currently working toward the PhD degree based on a research career with COSIC/ KU Leuven. He received the BSEE/MSEE degrees from Stanford. He is the chief scientist at Verayo, Research Affiliate for CSAIL/MIT. He was the Manager of R&D Engineering at TSI, and developed a secure digital baseband radio. where he was a Mayfield Fellow. His research interests include coding and security, he served on ACM and IACR program committees.



Srinivas Devadas received the MS and PhD degrees from the University of California, Berkeley, in 1986 and 1988, respectively. He is the webster professor of electrical engineering and computer science at the Massachusetts Institute of Technology (MIT), where he has been since 1988. He served as associate head of EECS from 2005 to 2011. His research interests include computer-aided design, computer architecture and computer security. He is a Fellow of the IEEE and ACM.