Trapdoor Computational Fuzzy Extractors

Charles Herder†, Ling Ren†, Marten van Dijk‡, Meng-Day (Mandel) Yu°, Srinivas Devadas†
cherder@mit.edu, renling@mit.edu, vandijk@engr.uconn.edu, myu@verayo.com, devadas@mit.edu

† MIT CSAIL ‡ UConn ° Verayo

November 15, 2014

***Abstract*—We describe a method of cryptographically-secure key extraction from a noisy biometric source. The computational security of our method can be clearly argued through hardness of Learning Parity With Noise (LPN).**

We use a fuzzy commitment scheme so the extracted key is chosen by definition to have uniformly random bits. The biometric source is used as the noise term in the LPN problem. A key idea in our construction is to use additional 'confidence' information produced by the source for polynomial-time key recovery even under high-noise settings, i.e., $\Theta(m)$ errors, where m is the number of biometric bits. The confidence information is never exposed and is used as a noise-avoiding trapdoor to exponentially reduce key recovery complexity. Previous computational fuzzy extractors were unable to correct $\Theta(m)$ errors or would run in exponential time in m.

A second key result is that we relax the requirement on the noise in the LPN problem, which relaxes the requirement on the biometric source. Through a reduction argument, we show that in the LPN problem, correlation in the bits generated by the biometric source can be viewed as a bias on the bits, which affects the security parameter, but not the security of the overall construction.

Using a silicon Physical Unclonable Function (PUF) as a concrete example, we show how keys can be extracted securely and efficiently even under extreme environmental variation.

***Index Terms*—Physical Unclonable Function, PUF, ring oscillator, learning parity with noise, LPN, learning with errors, LWE.**

I. Introduction

Fuzzy extractors convert biometric data into reproducible uniform random strings, and make it possible to apply cryptographic techniques for biometric security; the biometric can be human or silicon in nature. They are used to encrypt and authenticate user data with keys derived from biometric inputs [15]. Fuzzy extractors are designed to achieve information-theoretic security. Secure sketches, a part of the the fuzzy extractor model, are also proposed in [15] and allow precise reconstruction of a noisy input without addressing nonuniformity. On input w, a procedure outputs a sketch h. Then, given h (also called "helper data") and a value w' close to w, it is possible to recover w. The sketch is secure in the sense that 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.

Fuzzy extractors typically require two stages during key recovery: the secure sketch (error correction) phase and a privacy amplification (hashing) phase. The recovery of w represents the first of these phases. However, in typical biometric applications, w does not have full entropy, so it must be compressed prior to its use as a cryptographic key.

Based on the fuzzy extractor framework, it is possible to extract near-full-entropy keys from a biometric source while maintaining information-theoretic security. The information-theoretic security, however, comes at a high cost in terms of the raw entropy required. For example, consider generating a 128-bit key by using a hash function for privacy amplication. The entropy loss associated with the use of the hash alone is $\geq$ 128 bits [27] due to the left-over hash lemma. To satisfy this information-theoretic requirement would more than double the size of the entropy source required to generate a key. Moreover, there is additional leakage associated with the secure sketch (error correction) phase of the fuzzy extractor, where the helper data leakage has to be accounted for. The higher the noise in the biometric source, the larger the size of the helper data, and the more difficult it is to argue security. Moreover, growing the source entropy is expensive and grows the noise. For some noise settings, the resulting min-entropy may be negative. For these reasons there has been recent work looking into computational fuzzy extractors (e.g., [16]).

We present a computational fuzzy extractor with a trapdoor in this paper. The fuzzy extractor is based on a well-established computational hardness assumption corresponding to the Learning Parity With Noise (LPN) problem and is extensible to the Learning With Errors (LWE) problem. Broadly speaking, we differ from previous work in that:

- Constructions that are information-theoretically secure can only correct a limited number of errors in w [17], [40] or make strong assumptions about the statistics of the biometric source (e.g., [30], [43]).
- Constructions (e.g., [16]) that are computationally secure under well-established assumptions (e.g., LWE) require exponential time to correct $\Theta(m)$ errors, where m is the number of bits in the biometric source.
- Other constructions either assume that the helper data does not leak information (e.g., [33]) or argue computational security based on hardness assumptions of less well established problems [34], [44], [32].[1]

We summarize our contributions below:

1) We present a novel Project function that instead of correcting a small number of random errors, projects w in the fuzzy extractor discussion above onto a subspace

[1]For example, that *any* learning algorithm requires at least z vectors to learn a model with non-negligible advantage.

of w that has a lower probability of error (i.e., picks bits of w that are more stable). If it is unknown as to which bits are stable, this approach is only practical for low-noise settings ($O(\log m)$) and impractical for $\Theta(m)$ high-noise settings. However, we recognize that many physical systems that provide keying material also provide 'confidence information' that can be used to identify which bits of w are stable.[2] Incorporating this information into the key recovery process results in an exponentially faster algorithm that is polynomial-time even for $O(m)$ noise. This confidence information is *never* revealed, and as such serves as a *noise-avoiding trapdoor* during the key recovery process. In our construction the biometric source is itself the trapdoor.

2) We use m, the number of equations in the LPN problem (and the number of biometric bits) in a novel way. In LPN/LWE cryptosystems $m > n$ is determined based on requirements of encryption/decryption procedures. In our construction, the choice of m is crucial to reliable key recovery. We derive the value of m in relation to n such that a negligible failure probability can be obtained for $\Theta(m)$ noise and show that $m = \Theta(n^3)$ in this situation.
3) We define the assumptions that are made on the distributions of the biometric data as well as their correlations. We recognize that the proposed approach remains secure with weaker assumptions about these data than previous schemes which assumed uniform distributions in the LWE case. In particular, we do not require the biometric data to be i.i.d. or uniformly distributed. Through a reduction argument, we show that correlation in the biometric source turns into a heavier bias in the associated LPN problem, and only affects the security parameter and not the security of the construction.

We provide a simple extension of our construction that deals with helper data that can be maliciously manipulated without impacting security, and briefly discuss an extension to the more celebrated Learning With Errors (LWE) problem.

Finally, we provide a concrete instantiation of our scheme in the context of key generation using a silicon Ring Oscillator (RO) Physical Unclonable Function (PUF) that naturally provide confidence information [18], [41]. We analyze the relationship between the length of the secret key and the security parameter for LPN and provide concrete numbers for all aspects of the design. Using experiments on Field Programmable Gate Arrays (FPGAs) we show that our scheme has comparable helper data to previously-implemented schemes under higher environmental variation, and most important, easily-argued computational security.

Organization: We give background on LPN and PUFs in section II. We discuss related work in section III. Section IV describes the key provisioning and key extraction schemes whose security is based on LPN. The reliability of key extraction is the subject of section V, where we show that the confidence information serves as a trapdoor. We discuss our security assumptions in Section VI. Extensions to our construction are the subject of Section VII. We show in section VIII that an efficient, computationally secure key extraction scheme can be instantiated using ring oscillator PUFs. We conclude the paper in section IX.

[2] As an example, a bit in w' may be (re)generated using a majority vote on repeated evaluations, and the confidence information may simply be the strength of the vote.

II. Background

We provide background on the Learning Parity with Noise problem and ring oscillator Physical Unclonable Functions (PUFs) that we will use in the concrete instantiation of our key extraction scheme.

A. Learning Parity with Noise

The Learning Parity with Noise (LPN) problem is a famous open problem that is widely conjectured to be hard, as the best known algorithm is slightly subexponential ($2^{\Omega(n/\log n)}$) [7], [4]. As a result, this problem has since been used as the foundation of several cryptographic primitives [21], [3], [2], [6]. The LPN problem can be thought of as a special case of the Learning With Errors (LWE) problems discussed by Regev [35]. However, Regev's reduction to the independent shortest vector problem (ISVP) does not apply to the LPN case. Therefore, the difficulty of solving LPN is a separate conjecture from the difficulty of solving LWE.

The extension of our key extractor to the LWE case can be easily and briefly described; the majority of this paper will focus on a key extractor whose helper information reveals no bits of the secret to an adversary assuming that LPN is hard. With this in mind, we present the canonical LPN cryptographic construction. One can easily extend this construction to LWE by allowing the bits in the following representation to instead be integers modulo a prime number P.

The problem is posed as follows. Let $\overrightarrow{s} \in \{0,1\}^n$ be chosen uniformly at random. Let $a_i \in \{0,1\}^n$ be uniformly random, and i from 1 to $m > n$. Let $e_i \in \{0,1\}$ be chosen according to a distribution χ. Finally, define b_i as:

$$\begin{aligned} b_1 &= \langle \overrightarrow{a_1}, \overrightarrow{s} \rangle + e_1 \\ b_2 &= \langle \overrightarrow{a_2}, \overrightarrow{s} \rangle + e_2 \\ \vdots &= \quad \vdots \\ b_m &= \langle \overrightarrow{a_m}, \overrightarrow{s} \rangle + e_m \end{aligned}$$

The problem is to learn $\overrightarrow{s}$ given only the values of b_i and $\overrightarrow{a}_i$. When each e_i has probability τ of being 1 and probability $1-\tau$ of being 0 with non-negligible τ and $1-\tau$, there is no known polynomial-time algorithm that solves this problem.

B. Physical Unclonable Functions (PUFs)

Physical Unclonable Functions (PUFs) are primitives that are used for authentication and secret key storage without the requirement of secure EEPROMs or tamper-resistant hardware [18], [36]. This is possible, because instead of storing secrets in digital memory, PUFs derive a secret from the physical characteristics of the integrated circuit (IC). This approach is

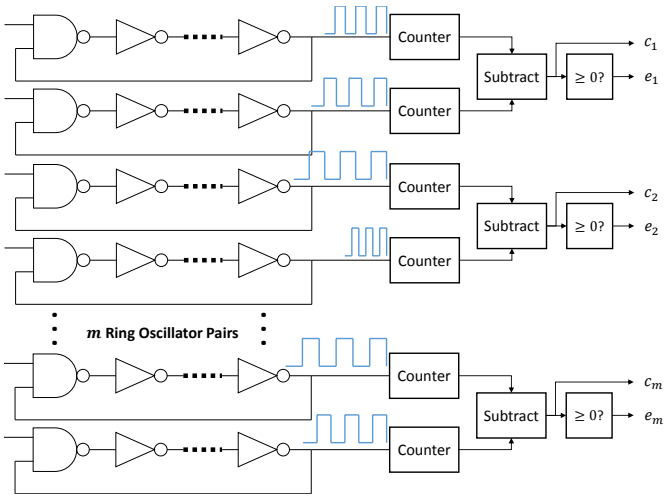

Fig. 1. A basic Ring Oscillator PUF with m differential pairs. Note that in addition to the output bits e_i, confidence values c_i may be made available to the surrounding logic. These confidence values are in the form of the actual differential count between the two ring oscillators, while the PUF output bits e_i correspond to whether the differential count is greater/less than 0.

advantageous over standard secure digital storage for several reasons including enhanced physical security of volatile keys (as opposed to keys stored in non-volatile memory).

A PUF is based on the idea 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'). In addition, due to the manufacturing variability that defines the secret, one cannot manufacture two identical chips, even with full knowledge of the chip's design. PUF architectures exploit manufacturing variability in multiple ways. In addition to gate delay, architectures also use the power-on state of SRAM, threshold voltages, and many other physical characteristics to derive the secret.

To date, there has been little or no cryptographic backing to many practical PUF implementations. When PUFs are used for authentication, many challenge response schemes are vulnerable to machine learning attacks (e.g., [37]). Our focus in this paper is key generation using PUFs. These schemes typically use public helper data to correct for PUF errors. While information-theoretic constructions exist, in practical implementations, this helper data either does not correct enough PUF errors or there is no clear theory proving that the helper data bits do not reveal bits of the secret key.

C. Ring Oscillator PUFs

The PUF construction in this paper will be based on the Ring Oscillator (RO) PUF, shown in Figure 1. Many PUF architectures have been proposed that leverage this particular construction [43], [45], [44], [32]. In general, RO PUFs generate bits by comparing the frequencies of two ring oscillators that are identical by design. Manufacturing variation, however, leads to one of the oscillators being faster than the other.

The frequencies are compared by counting oscillations from each oscillator for a certain amount of time and subtracting one from the other. The PUF bit is simply determined by which oscillator is faster. The design and layout of each ring oscillator is identical, and fabrication variation results in different frequency of oscillation. Each e_i bit could be a 0 or a 1. Moreover, because the measurement is differential, many environmental effects are significantly reduced. However, noise and environmental factors can and do cause errors in the regeneration of bits by ring oscillator PUFs. Therefore, key extraction needs to correct for these errors.

The variation in the manufacturing process typically has both random and systematic components. Assumptions need to be made about the independence of ring oscillator frequencies and/or the entropy and bias of bits generated from ring oscillator comparisons to make claims about the extracted keys.

Additional bits of information can be used in a fuzzy extractor, in particular, the difference in counts between the two oscillators. It has been observed that if the difference in counts between the two ring oscillators (c_i) is large, then one can have a higher degree of confidence that noise and changes in environmental parameters will not cause the output bit e_i to flip erroneously when read from the PUF at a later time [41].

This confidence information has been used to generate helper data in some schemes [41] [43]. It is not used as a part of the secret key. In our construction, we will *not* use the confidence information to even generate helper data[3], but we will use it to aid the extractor design process and key recovery process, never exposing it to the outside. This will be described in greater detail in sections IV and V.

Each PUF bit (denoted e_i) corresponds to a pair of ring oscillators. Therefore, to generate unique e_i for $1 \leq i \leq m$, one needs a bank of m pairs of ring oscillators.

III. Related Work

A. PUF Error Correction

Silicon Physical Unclonable Function (PUF) key generation was first introduced using Hamming codes in [17] and more details were presented in [40]. A limited number of errors could be corrected since the security argument was information theoretic. Specifically, if one requires a k-bit secret from n bits generated by the PUF, then at most $n - k$ bits could be exposed.

The fuzzy extractor described in [15] consists of a secure sketch (error correction) and privacy amplication stage (hashing), and [15] described the requirements for information-theoretic security. There have been many attempts to instantiate a version of a fuzzy extractor in the context of PUF key generation. These works can be divided based on the scope of the security argument used to justify the security of the PUF-generated keying bits.

Information-Theoretic Security of Error Correction Helper Data. There were several works that created helper data that is information-theoretically secure. These works, however, were confined to address the first part of the fuzzy extractor (secure sketch) and did not explicitly address the leakage associated

[3]If we did we would have to argue about the security of this use.

with the second part (privacy amplification). We describe these below.

[43] uses PUF error correction helper data called Index-Based Syndrome (IBS), as an alternative to Dodis' code-offset helper data. IBS is information theoretically secure, under the assumption that PUF output bits are independent and identically distributed (i.i.d.). Given this i.i.d. assumption, IBS can expose more helper data bits than a standard code-offset fuzzy extractor construction. We note that [43] uses confidence information to generate helper data and therefore requires the i.i.d. assumption to argue security.

Efficiency improvements to IBS, which maintained information-theoretic security of the error correction helper data are described in [19] and [20]. A soft-decision PUF error correction decoder was described in [30], [31] where the confidence information part of the helper data was proven to be information-theoretically secure under an i.i.d. assumption (the remaining redundancy part followed the standard code-offset construct).

We note that while these works are laudable, and practical implementations were created based on a provably secure information-theoretic foundation, these works did not explicitly address the full key generation processing (secure sketch + privacy amplification), and addressed only the error correction (secure sketch) phase, and are in that sense incomplete. Further, they made a strong assumption on PUF output bits. Finally, while they use confidence information (to generate helper data), our construction uses confidence information in a very different way.

Computational Security Based on Machine Learning Heuristics. There were several works [34] [44] that created helper data that were heuristically secure based on results of state-of-the-art machine learning attacks on PUFs (e.g., [38]). These designs used a candidate strong PUF based on XORs [41] but leak only a limited number of PUF response bits as helper data to generate a key. While over 3+ years of attacks by several groups around the world have established the heuristic security of leaking a limited number of bits from a candidate strong PUF [12] [22] [37] [38] there is not yet a proof to reduce this difficulty into a computational hardness assumption accepted by the cryptography community. These works are also limited in scope in that they do not explicitly address the full key generation processing, only the error correction phase.

Secure Sketch + Privacy Amplification. To the best of our knowledge, there is one paper that attempted to implement and address the security associated with both stages of a PUF fuzzy extractor [32]. The paper accounted for the information-theoretic loss of the error correction helper data, using code-offset syndrome [15], but did not implement the extra overhead required for an information-theoretically secure privacy amplification stage and instead opted for a more efficient implementation using a lightweight hash called SPONGENT [8] as an entropy accumulator. The error rate corrected in [32] corresponds to limited environmental variation of ring oscillator PUFs, and the overheads for a 128-bit secret (i.e., 2000+ bits of helper data) would increase if environmental noise increased.

Others. There are several works that cite the need for a fuzzy extractor [15] but did not fully implement it in a manner that fully accounts for the information-theoretic entropy loss. This includes [9] [42]. There are some works that describe a PUF key generation scheme but without specific accounting for entropy loss associated with helper data and key generation [33].

In aggregate, these works show that while the fuzzy extractor framework [15] is theoretically sound, it is difficult to implement practically and achieve an information-theoretic level of security in real-world settings. As such, a more efficient key extraction framework, based on an established computational hardness problem is compelling.

B. Computational Fuzzy Extractors and Fuzzy Encryption

Fuller et al. [16] describe a computational fuzzy extractor based on LWE. Their construction is similar to our LPN construction in that the biometric source is used as the noise term in the LWE problem and in their use of a fuzzy commitment scheme. However, they do not have a Project function and therefore can only correct $O(\log m)$ errors efficiently. They state that there is no place to store a trapdoor in a fuzzy extractor because there is no place for secure storage – in our construction the biometric source is itself the trapdoor.

Our work is related to but distinct from fuzzy identity-based encryption (IBE) [39] [1]. Fundamentally, in fuzzy IBE, the noisy ID string is public and is used to derive the private key that is noise-free. In our case, the noisy ID string (e.g., PUF bits) are private and are used to derive the 'public key' (our helper information) that is noiseless. Correspondingly, the noisy ID bits in our construction affect the implementation of the decryption algorithm, while in fuzzy IBE constructions, they affect the encryption algorithm.

C. Helper Data Manipulation

The issue of helper data manipulation has been addressed in the biometric realm with robust fuzzy extractors [10] [14]. Their use of a helper data hash do not address recent helper data manipulation attacks in [25] [13]. If helper data is manipulated in our schemes, key recovery may fail if the corrupted bits are used to solve for the key. The adversary can conceivably learn information about what bits are stable based on helper data manipulation and key recovery success or failure. Our scheme can be made secure against helper data manipulation (see section VII).

IV. Fuzzy Extractor Using LPN

We will construct a computationally secure extractor from the LPN problem. Recall the description of the LPN problem in section II-A. The key intuition behind our construction is that one can use biometric data as the e_i values. Therefore, while an adversary may have to learn the equations with probability of error being τ, where τ relates to the entropy of the biometric data, having access to the biometric data allows one to regenerate estimates e_i' of the original e_i values. Note

that these are estimates due to intrinsic noise of the biometric data as well as environmental changes. Therefore, the error rate τ' of the LPN problem with access to the biometric data is dramatically reduced ($\tau' \ll \tau$). However, the LPN problem remains hard even for small τ' implying that key recovery will run in exponential time for $\Theta(m)$ number of errors.

We will show in the next two sections that using this intuition in combination with 'confidence information' techniques allows the one to *efficiently* regenerate the secret key given just the public helper information $(\overrightarrow{a_i}, b_i)$, and the biometric source, even for $O(m)$ errors.

In the following description, we will refer to the bits of the biometric data (measured at provisioning) as $\overrightarrow{e} = \{e_1, e_2, \dots e_m\}$, ($e_i \in \{0,1\}$) and their noisy counterparts (measured during key extraction) as $\overrightarrow{e}' = \{e'_1, e'_2, \dots e'_m\}$, (once again, $e'_i \in \{0,1\}$). Moreover, the confidence information associated with these noisy measurements (where applicable) will be denoted as $\overrightarrow{c}' = \{c'_1, c'_2 \dots c'_m\}$, where $c'_i \in \mathbb{Z}$.

We describe the algorithms associated with the key extraction below. Typically, a fuzzy extractor, information-theoretic or computational, has the functions Gen and Extract, where Gen produces the public helper information, and Extract takes the biometric data and public helper information and returns the error-corrected key.

This paper expands this construction to include four functions in the extraction process:

- $\mathsf{Fab}(1^k, \epsilon_1, \epsilon_2, \mathsf{BIAS})$: Represents the fabrication step: takes the security parameter k, the desired probabilities of failure ϵ_1, ϵ_2 (see section V for further description). It performs the following:
  1) Set $n = f(k, \mathsf{BIAS})$ (see section VI for discussion). The BIAS term describes the bias of individual ring oscillators towards 0 or 1. n will be the size of the secret vector. n will increase with as the ring oscillators become more biassed (BIAS goes to 0 or 1). This is reflected in the analysis in VI.
  2) Compute m such that with probability greater than $1-\epsilon_1$, at least n of the m bits of biometric data are 'stable' over relevant noise/environmental parameters. We define 'stable' to be $\mathsf{Pr}(e'_i \neq e_i) \leq \epsilon_2$. (See section V for how this is done). Note that although we will use confidence information to identify these stable bits during key reconstruction, no confidence information is used in Fab or Gen.
  3) Manufacture a device that produces m biometric bits.
- $\mathsf{Gen}(n) \rightarrow (\overrightarrow{a}_i, b_i)$: Represents the provisioning step of the device manufactured in Fab. Gen takes the size of the secret vector n (calculated in Fab) and returns public helper data $(\overrightarrow{a}_i, b_i)$ to be used in the Recovery function. Recall that there are m bits of biometric data. Gen has the following steps:
  1) Measure m bits of biometric data as $\overrightarrow{e} = \{e_1, e_2, \dots e_m\}$.
  2) Generate m unique uniformly random $\overrightarrow{a_i} \in \{0,1\}^n$.
  3) Generate uniformly random $\overrightarrow{s} \in \{0,1\}^n$. For example, using a hardware random number generator. (We differ from a conventional fuzzy extractor in that we are using a fuzzy commitment scheme [24]).
  4) Compute $b_i = \langle \overrightarrow{a_i}, \overrightarrow{s} \rangle + e_i$ for i between 1 and m.
  5) Store in insecure nonvolatile memory and/or publish $(\overrightarrow{a}_i, b_i)$. Note that the $\overrightarrow{a}_i$ can be the same for every system and can be hardcoded into the system. Therefore, we will only have to store system-specific b_i's and count these as the helper data, not the $\overrightarrow{a}_i$'s, which are public system parameters.
  6) Discard $\overrightarrow{s}$ and e_i.
- $\mathsf{Project}(\overrightarrow{c}', \epsilon_2) \rightarrow S$: Represents the algorithm that determines the projections that Recovery will use during its execution. In effect, an algorithm that determines the 'stable bits' of biometric data. We describe Project as taking a set of confidence information $\overrightarrow{c}' \in \mathbb{Z}^n$ that is correlated to the probability that a given bit of biometric information is stable.
  1) Use confidence information c'_i to find $n \leq m' < m$ stable bits of the biometric data. For each of the m' stable bits, $\mathsf{Pr}(e'_i \neq e_i | c'_i) < \epsilon_2$ for some small ϵ_2. (See section V for how this is done).
  2) Construct S, a polynomial-sized sample of the $\binom{m'}{n}$ possible subsets of size n of the m' stable bits. Different rules may be used to select these subsets depending on application.
  3) Return S.
- $\mathsf{Recovery}(\overrightarrow{e}', S) \rightarrow \overrightarrow{s}$: Represents the augmented key recovery algorithm.[4] Note that in addition to receiving the noisy biometric data $\overrightarrow{e}'$, this function also takes as an argument S, the polynomially-sized set of projections that describe how to choose n stable bits from the m noisy bits in $\overrightarrow{e}'$. The steps of the algorithm are as follows:
  1) Select $S_i \in S$, and select the n equations denoted by S_i. (See figure 2).
  2) Use Gaussian elimination on the n equations to solve for $\overrightarrow{s}$.
  3) Repeat (1) and (2) for each $S_i \in S$. We now have $|S|$ different estimates for $\overrightarrow{s}$.
  4) Return the $\overrightarrow{s}$ that occurs most frequently.

We note that the Fab algorithm can be viewed as system design steps, while the Gen algorithm will be executed for each instantiation of the design. One exception to this is the selection of $\overrightarrow{a}_i$. These values can be set to constant, random values across all designs of this type.

Before looking into the effects of noise on the above algorithm, there are several notes that should be made.

First, if the LPN problem is hard, then an adversary in possession of $(\overrightarrow{a_i}, b_i)$ cannot compute $\overrightarrow{s}$. In fact, the public helper information $(\overrightarrow{a_i}, b_i)$ is *precisely* the public key of the LPN cryptosystem. Since knowing $\overrightarrow{e}$ makes it trivial for an adversary to solve $\overrightarrow{s}$, the LPN hardness assumption also guarantees an adversary cannot figure out $\overrightarrow{e}$. Therefore, if the LPN conjecture is true, this key extraction algorithm is secure.

[4]Together, Project and Recovery correspond to Extract in a fuzzy extractor.

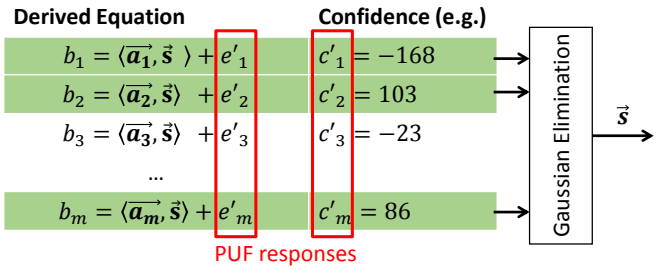

Fig. 2. Overview of LPN Key extraction algorithm. The e'_i values are regenerated and the c'_i values identify the e'_i values that have high absolute values of confidence. Gaussian elimination is then used on these selected equations to extract the secret key.

Also recognize that although the algorithm uses the confidence information c'_i for each bit of biometric data e'_i, confidence information is not used to determine the secret key nor the public key, never published, and is discarded after use. Therefore, no adversary will have access to this information.

There are two critical intuitions as to why the LPN key extractor works. One is that the LPN (and LWE) algorithm allows *arbitrary redundancy* in the number of equations supplied. In other words, m can be much greater than n. Therefore, we can supply enough helper information to where with high probability (see section V for precise definitions) we will have greater than n stable bits of biometric data, even for large error rates. Furthermore, the confidence information c'_i acts as a trapdoor for identifying 'stable' bits in key recovery. Therefore, the key recovery algorithm is faced with a much lower error rate and hence much easier problem than an adversary without the trapdoor.

This intuition suggests that difficult learning problems in general should be applicable to key extraction algorithms since the difficulty such problems is insensitive to the number of training data supplied.

Without the confidence information, recovery degenerates to the computational fuzzy extractor by Fuller et al. [16]. We show below that such a computational fuzzy extractor can only correct a logarithmic number of errors in the noisy biometric data. Its recovery algorithm works as follows:

1) Upon receiving a noisy version of an input biometric data $\vec{e}' = \{e'_1, e'_2, \ldots e'_m\}$, randomly select n out of the m equations $b_i = \langle \vec{a_i}, \vec{s} \rangle + e'_i$.
2) Solve $\vec{s}$ (using Gaussian Elimination for instance) and compute the residue $\Sigma_{i=1}^{m} | \langle \vec{a_i}, \vec{s} \rangle + e'_i - b_i|$.
3) If the residue is $\leq t$, output the recovered string $e_i = \langle \vec{a_i}, \vec{s} \rangle - b_i$ for i between 1 and m. Else, repeat the above steps.

If the input $\vec{e}'$ satisfies $|\vec{e} - \vec{e}'| \leq t$, the recovery algorithm will eventually terminate with the correct $\vec{e}$ once it selects n noise-free equations. From those equations, the algorithm would find the correct $\vec{s}$ and recover $\vec{e}$. If $|\vec{e} - \vec{e}'| > t$, the recovery algorithm is likely to not converge.

Now we analyze the expected runtime of this recovery algorithm. Assuming $|\vec{e} - \vec{e}'| \leq t$, in each iteration, the algorithm has a probability of at least $\binom{m-t}{n}/\binom{m}{n}$ to select n noise-free equations. So the expected number of trials is $\binom{m}{n}/\binom{m-t}{n} = \frac{m(m-1)\cdots(m-n+1)}{(m-t)(m-t-1)\cdots(m-t-n+1)} < (\frac{m}{m-t})^n = (1 + \frac{t}{m-t})^n < e^{\frac{nt}{m-t}}$.

For some parameter settings ($m = 2n$, $t \ll m$), the number of expected iterations is less than e^t, and the recovery algorithm finishes in $e^t \mathsf{Poly}(n)$ time. Therefore, $t = O(\log m)$ allows this recovery algorithm to be polynomial time. For high-noise situations, t could, for example, be $\frac{m}{4}$ or $\Theta(m)$, and the above recovery algorithm requires an exponential number of iterations. In fact, the LPN conjecture precludes the existence of any polynomial recovery algorithm at such a high noise rate.

We will show in sections V and VIII how using confidence information can reduce the expected number of iterations to 1.

V. Noise-Avoiding Trapdoors

In section IV, the LPN based key extraction construction requires two functions (Fab and Project) that required knowledge of the error rate of bits of biometric data. This section will explore how these functions are implemented in the case where the biometric data is provided by ring oscillator PUFs.

Fab uses a priori knowledge of the statistical distribution of PUF internal properties. The bound on bias given to Fab is public information. Project leverages 'confidence' information that the PUF bit is correct ($e'_i = e_i$) . Confidence data are extracted upon measurement of a PUF in the manner described in section II-C, and are considered private information.

As discussed in section II-C, ring oscillator (RO) PUFs are a standard PUF topology. Moreover, one can easily measure not only the output bit from a differential ring oscillator pair, but also some 'confidence' information in the form of the magnitude of the difference in counts between the two oscillators.

In general, if the RO pair differs by a large number of counts, then one can be more confident that the bit has not (and will not) flip when one considers noise and/or changes the temperature, voltage, or other environmental parameters.

We define our notation as c_i to be the differential counts measured at the time of PUF provisioning, and $e_i = \mathsf{Sign}(c_i)$. We define c'_i to be the differential counts measured at the time of key extraction and $e'_i = \mathsf{Sign}(c'_i)$.

To provide concrete analysis, we consider the probability distribution of the c_i and c'_i values. Among different pairs of ring oscillators (i.e., the distribution of c_i with no prior information), this PDF can be taken to be a zero-mean Gaussian with variance σ_{INTER}. Note that this directly implies that the τ parameter to the LPN problem is $\frac{1}{2}$. In other words, e_i is not biased towards 1 or 0.

In actual physical systems, there will be a bias towards 1 or 0. However, we will see that assuming a 50% bias represents a 'worst case' *from the standpoint of error correction*. (Note that we will use a different worst case bias of 40% as an input to Fab to determine n given the security parameter in section VI.)

Now, given that c_i and c'_i represent the distribution of measuring the *same* RO pair at different times and environmental parameters, the conditional distribution $\Pr(c_i|c'_i)$ is different.

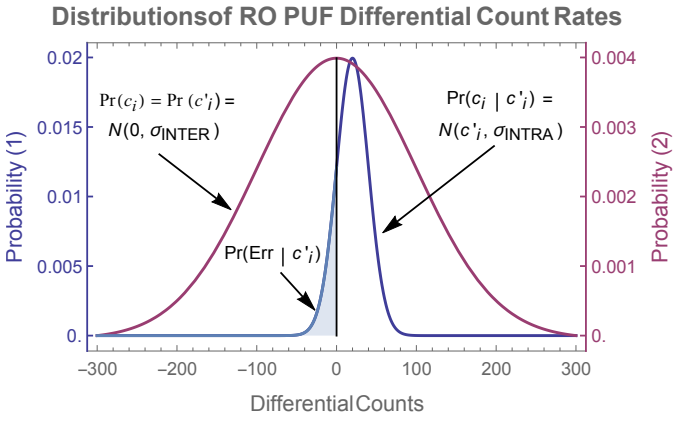

Fig. 3. Distribution of differential counts for RO pairs among different RO pairs and with the same i^{th} pair measured repeatedly over time/environmental parameters. We show the probability of error given a confidence measurement c_i' as the integral of the shaded region.

This distribution is described as a Gaussian distribution with mean c_i' and variance σ_{INTRA}. Both distributions are shown in figure 3.

Note that since c_i and c_i' are extracted from the same ring oscillator pair, they have the same distribution (with no prior knowledge). Therefore, one can recognize that $\mathsf{Pr}(c_i|c_i') = \mathsf{Pr}(c_i'|c_i)$.

This is critical, because it allows us to use the confidence information collected during the fabrication step to reason about the probability of error at key extraction. We may now define the probability of error given confidence information. Since $e_i = \mathsf{Sign}(c_i)$, we recognize that the error probability given a measurement of the confidence information is the integral of the shaded region in figure 3. In particular, the CDF up to 0:

$$\mathsf{Pr}(e_i' \neq e_i|c_i') = \frac{1}{2}\left(1 + \mathsf{Erf}(-|c_i'|/(\sqrt{2}\sigma_{\text{INTRA}}))\right) \quad (1)$$

First note that the probability of error in this system $\mathsf{Pr}(e_i' \neq e_i)$ is the integration of equation 1 over the distribution of c_i', which is the distribution in figure 3. This probability depends *only* on σ_{INTRA} and σ_{INTER} (not on n, m, etc.). Therefore the error rate for m ring oscillator pairs is clearly $\Theta(m)$.

Once again, due to the symmetry of the distributions of c_i and c_i', we know that $\mathsf{Pr}(e_i' \neq e_i|c_i') = \mathsf{Pr}(e_i' \neq e_i|c_i)$.

With this in mind, we may now describe how Fab and Project work.

A. Fabrication/Provisioning

In addition to the bound on bias, the Fab algorithm is given a security parameter k, and two error rates ϵ_1, $\epsilon_2 > 0$ that are defined as follows. Fab must compute m such that with probability greater than $1 - \epsilon_1$, at least n of the m ring oscillator pairs will produce stable bits. A bit e_i is defined to be stable if $\mathsf{Pr}(e_i' \neq e_i) < \epsilon_2$ over relevant environmental parameters.

We recognize that, given c_i, we can compute the probability of error of a future measurement c_i' according to equation 1 and the symmetry of c_i and c_i'.

We use this in combination with the distribution on c_i to compute how many ring oscillator pairs we need before at least n of them will produce stable bits with probability $1 - \epsilon_1$.

To do this, we recognize that requiring $\mathsf{Pr}(e_i' \neq e_i|c_i) < \epsilon_2$ sets a threshold c_T. If a ring oscillator has $|c_i| > c_T$, then the probability of an error in this bit is less than ϵ_2 (the bit is stable). We 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 ring oscillator pair is stable (has a c_i value above the threshold) can be computed by integrating the PDF of $\mathsf{Pr}(c_i)$ shown in figure 3 at in the region $|c_i| > c_T$. This is equal to:

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

The inequality is because the probability of a bit being stable is *smallest* when the bit bias is 50% (the Gaussian is centered at 0). One can see that as the center of the Gaussian shifts, more probability density falls in the region of $|c_i| > c_T$. Therefore, we will complete the calculation assuming bias is 50% with the knowledge that the probability of a stable bit can only be higher than our calculation expects. Plugging in c_T from equation 2 gives the completed formula:

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

The final step is to compute m such that at least m' PUF bits will be stable ($c_i > c_T$), with probability $1-\epsilon_1$. This is a binomial distribution and is subject to a Chernoff bound. We identify the canonical Chernoff bound for observing less than m' stable bits out of a total of m, given P_{STABLE}:

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

Using this bound and our requirement of ϵ_1, we find that to ensure the probability that n of the m PUF bits are stable with probability $1-\epsilon_1$, we need the following relationship between n and m:

$$m \geq \frac{1}{P_{\text{STABLE}}}\left(m' - \log(\epsilon_1) + \sqrt{\log(\epsilon_1)\left(\log(\epsilon_1) - 2m'\right)}\right) \quad (4)$$

Where P_{STABLE} is a function of ϵ_2. Therefore, given m', ϵ_1, and ϵ_2, one can compute m such that at least m' of the PUF bits are stable, as is required in the Fab algorithm.

B. Projection/Extraction

The extension of the above analysis to the Project algorithm is comparatively simple. Recall that in Project, we will use the confidence information c_i' to generate a set S of a polynomial number of sets $S_i \in [m]$ such that $\forall j \in S_i, \mathsf{Pr}(e_j' \neq e_j|c_j') < \epsilon_2$ and $|S_i| = n$.

Given an ϵ_2, we use equation 2 to compute a threshold c_T value for which we can be confident that if $|c_i'| > c_T$, $\mathsf{Pr}(e_i' \neq e_i | c_i') < \epsilon_2$.

We then use the PUF to generate each of the m values of c_i. We select only those such that $|c_i'| > c_T$. Because of the properties of the Fab algorithm, we can be confident that we will find at least n of them. Assume that the algorithm finds $m' > n$ stable bits. There are a total of $\binom{m'}{n}$ possible sets that can be returned.

Project then selects an arbitrary non-empty subset of these $\binom{m'}{n}$ sets and returns it as S.

C. Showing the 'Trapdoor'

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

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

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

Plug them into equation 4, and one obtains:

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

This shows that even in pessimistic scenarios, $m \in \mathsf{Poly}(k)$. In reality, $\sigma_{\mathrm{INTRA}}/\sigma_{\mathrm{INTER}} < 1$, so the growth is much smaller, as will be seen in section VIII.

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

D. Concluding Remarks on Noise Analysis

The noise analysis above depends on only two properties of the PUF: σ_{INTER} and σ_{INTRA}. Both of these parameters are identical for every ring oscillator pair and every physical chip.

Moreover, the critical property of PUFs is that $\sigma_{\mathrm{INTRA}} < \sigma_{\mathrm{INTER}}$. However, $\sigma_{\mathrm{INTRA}} > 0$, so the key extraction technique must correct for possible errors.

Note also that we do not justify the assumption that the distribution of measuring the i'th ring oscillator pair repeatedly over time and environmental parameters should be Gaussian. Indeed, this distribution depends on the distribution of environmental conditions that the PUF is operating in, which is far more complex. However, we will use this distribution as a pessimistic estimation of the potential variations in a single ring-oscillator pair by setting σ_{INTRA} to be large enough. This will be discussed further with associated empirical data in section VIII.

VI. Security Analysis and Assumptions

A. Security Parameter and Computational Hardness Assumption

The security of our construction clearly depends on the conjecture that the LPN problem is hard. There is significant evidence to justify this conjecture [35], [7], [5]. Moreover, this construction is simply extensible to depend on the LWE problem, which has been reduced to the ISVP problem [35].

In the above algorithm, we derive the size n of the secret vector $\overrightarrow{s}$ as a function of the security parameter k. In typical LPN constructions, the adversary gains significant advantage can be gained from the fact that distribution of the error term $\Pr(e_i = 1)$ must be much less than 50% for the decryption algorithm to execute correctly. For a security parameter of 128, the associated noise probability $\tau = \Pr(e_i = 1) = 0.0024$ [11]. This in turn requires n to be very large – on the order of 29000.

However, although our algorithm is based on the LPN problem, we *do not* use the LPN encryption/decryption algorithm. *Therefore, we do not have the same restriction on* τ.

In our construction we set τ to be the bias of the biometric data. We pessimistically assume this bias to be 40% ($\tau = 0.4$). We emphasize that the key extractor will work for any non-zero bias for a corresponding n. As such, we use the analysis presented in [5] to discover an estimate for n based on k in order to obtain a security parameter of 128. We obtain for this set of parameters, $n \approx 540$. Details of this analysis are presented in the Appendix.

B. Assumptions on Biometric Data

We do not assume that the bits of biometric data have perfect 50% bias. Therefore, we must discuss our assumptions about the bounds of this bias. We must also discuss our assumptions about whether one bit is correlated to a different bit. This correlation may exist within a single extractor (i.e., if e_i is correlated to e_j), as well as across different extractors (i.e., if e_1 is correlated across two or more different extractors).

We do assume for the purposes of experimentation that ring oscillator comparison bits will have a bias between 40% and 60%. This is a loose bound, as we will see in section VIII.

We *do not* assume that different biometric bits are uncorrelated (on the same extractor, or different extractors). This correlation is plausible in a physical instantiation of such a construction. For example, in the case of biometric data coming from a Ring Oscillator (RO) PUF, the i^{th} ring oscillator pair (resulting in e_i) may always have significant bias towards 1 among different chips due to some effect resulting from its location on the die/wafer. We show in the next section that correlation in biometric data does not break LPN hardness.

C. Dealing with Correlation in Biometric Data

We will show that potential correlation between biometric data (in one or more devices) does not impact the security of the overall construction. Basically, correlation turns into a increased bias for ring oscillator bits that, in turn, will result in increased n to keep the security parameter the same.

We show in Lemma VI.2 that attacking a system of LPN problems (or one problem) that has correlated bits is still hard if you don't know which bits they are. This is the case if only the public helper information $(\overrightarrow{a}_i, b_i)$ are revealed.

Definition VI.1. *Define the 'Correlated LPN' problem* CORLPN *with arguments η, τ' as the following problem:*

Present the public keys $(\overrightarrow{a}_{i,u}, b_{i,u})$, $(\overrightarrow{a}_{i,v}, b_{i,v})$ of 2 LPN systems to an adversary, where:

$$b_{i,u} = \langle \overrightarrow{a}_{i,u}, \overrightarrow{s}_u \rangle + e_{i,u}$$
$$b_{i,v} = \langle \overrightarrow{a}_{i,v}, \overrightarrow{s}_v \rangle + e_{i,v}$$

$$\eta = \mathsf{Pr}_i(e_{i,u} = e_{i,v})$$
$$\tau' = \mathsf{Pr}_i(e_{i,u} = 1) = \mathsf{Pr}_i(e_{i,v} = 1)$$

Ask the adversary to solve for $\overleftarrow{s_u}$, $\overleftarrow{s_v}$.

Without loss of generality, define $\eta > 1/2$, and $\tau' < 1/2$.

We present the following Lemma: (Once again, we let $\eta > 1/2$, τ, $\tau' < 1/2$ without loss of generality)

Lemma VI.2. *An algorithm $\mathcal{A}$ capable of solving* CORLPN *with correlation η and bias τ' to non-negligible advantage in polynomial time can solve a system of two uncorrelated* LPN *with non-negligible advantage in polynomial time if its bias is*

$$\tau = \frac{1}{2}\left(1 - \sqrt{2\eta - 1}\right)$$

as long as $1/2 > \tau' > \tau$.

Proof. At a high level, we will prove this lemma by constructing a correlated LPN problem from an uncorrelated LPN problem while maintaining the solution of the uncorrelated problem. We do this by adding correlated noise to the public key of the uncorrelated LPN problem.

In particular, we define the following problem that is equivalent to the problem in Definition VI.1.

System A: Define a system of three LPN problems:

$$b_{i,u} = \langle \overrightarrow{a}_{i,u}, \overrightarrow{s}_u \rangle + e_{i,u} \tag{5}$$
$$b_{i,v} = \langle \overrightarrow{a}_{i,v}, \overrightarrow{s}_v \rangle + e_{i,v} \tag{6}$$
$$b_{i,u} + b_{i,v} = \langle \overrightarrow{a}_{i,u} | \overrightarrow{a}_{i,v}, \overrightarrow{s}_u | \overrightarrow{s}_v \rangle + e_{i,w} \tag{7}$$

Let $\mathsf{Pr}_i(e_{i,u} = 1) = \mathsf{Pr}_i(e_{i,v} = 1) = \tau'$, and $\mathsf{Pr}_i(e_{i,w} = 1) = 1 - \eta$. (This can be seen because $e_{i,w} = e_{i,u} + e_{i,v}$.)

Give the public keys $(\overrightarrow{a}_{i,u}, b_{i,u})$, and $(\overrightarrow{a}_{i,v}, b_{i,v})$ to an adversary and ask it to solve for $\overrightarrow{s}_u$ and $\overrightarrow{s}_v$.

If $\mathcal{A}$ can solve CORLPN, clearly it can solve for $\overleftarrow{s_u}$, $\overleftarrow{s_v}$ given the above system of equations.

Now, consider a system of LPN problems using $\mathsf{Pr}_i(e_{i,x} = 1) = \mathsf{Pr}_i(e_{i,y} = 1) = \tau$, and $e_{i,x}$, $e_{i,y}$ uncorrelated. We would like to use $\mathcal{A}$ to break this system:

System B:

$$b_{i,x} = \langle \overrightarrow{a}_{i,x}, \overrightarrow{s}_x \rangle + e_{i,x} \tag{8}$$
$$b_{i,y} = \langle \overrightarrow{a}_{i,y}, \overrightarrow{s}_y \rangle + e_{i,y} \tag{9}$$
$$b_{i,x} + b_{i,y} = \langle \overrightarrow{a}_{i,x} | \overrightarrow{a}_{i,y}, \overrightarrow{s}_x | \overrightarrow{s}_y \rangle + e_{i,x} + e_{i,y} \tag{10}$$

Note that since $e_{i,x}$, $e_{i,y}$ are uncorrelated with bias τ, this implies $\mathsf{Pr}_i(e_{i,x} + e_{i,y} = 1) = 2\tau(1 - \tau)$.

We construct System B such that $\mathsf{Pr}_i(e_{i,x} + e_{i,y} = 1) = \mathsf{Pr}_i(e_{i,w} = 1)$. Specifically, we define $\tau = \frac{1}{2}\left(1 - \sqrt{2\eta - 1}\right)$. Now, the LPN problem described by equation 7 in System A and the LPN problem in equation 10 in Problem B have the same error probability.

Given an uncorrelated system with these error probabilities (System B), we can now modify this system (System B) such that all three LPN problems have the same error statistics as the correlated system (System A) by probabilistically flipping bits in the public keys $((a_{i,x}, b_{i,x}), (a_{i,y}, b_{i,y}))$. If the error rates are identical, then the systems are identical, and $\mathcal{A}$ can break it.

We modify the public keys as follows: Given an initial error rate τ of System B, and a desired error rate $\tau' > \tau$ of System A, we randomly flip bits of $b_{i,x}$ with probability $\frac{\tau' - \tau}{1 - 2\tau}$. This will increase the error rate of the LPN problem in equation 8 to τ'.

Now, we flip the *same* bits of $b_{i,y}$. This way, we increase the error of the LPN problem in equation 9 to τ', but the error rate of the LPN problem in equation 10 remains the same – flipping the same bits of $b_{i,1}$ and $b_{i,2}$ have no effect on the left hand side.

Therefore, by probabilistically flipping bits of the public key of System B, we have created a system identical in its statistics to System A.

Therefore, we can now use $\mathcal{A}$ to extract $\overrightarrow{s_x}$, $\overrightarrow{s_y}$. This is a contradiction if the LPN problem with error rate τ is hard. □

In effect, the proof shows that one can *introduce correlation* to a system of LPN problems by modifying only the public keys.

Note that the above proof trivially extends to the case where > 2 systems of LPN problems are considered. One can simply flip the *same bits on all three* b_i vectors to add correlation between the three systems.

It also extends to a single LPN problem that has correlation within several of its internal bits. For example, say one wanted to construct an LPN problem with e_1, e_2 correlated using an uncorrelated LPN problem by only modifying the public parameters. One starts with an uncorrelated LPN problem with significant bias (same as in the reduction). One then probabilistically flips b_i bits, *excluding* b_1, b_2. The new LPN problem has e_1, e_2 correlated, but clearly the difficulty is reducible to the original problem, since only bits of the public key are flipped.

Therefore, if we believe that bits of the biometric data are correlated with probability η, one must increase n to maintain the security parameter (τ has decreased), but the security of the problem is maintained.

We note that ring oscillators are a primary means of discovering and measuring correlations in silicon fabrication processes. Therefore, a fuzzy extractor designer can take correlation into account to set a bound on the bias. If analysis of fabricated ring oscillators in a particular fabrication process shows correlation due to the manufacturing process, and this is discovered *after* LPN key extractor chips have been built, these chips need not be thrown away provided n can be configured appropriate to the correlation. Assuming a loose bias for the design (e.g., 40%) is thus a conservative approach.

D. Advantages of LPN-based Key Extractor

The key extractor described in this paper is novel in its use of LPN as a foundation for its security. It turns out that taking this approach results in several advantages over previous approaches that required information-theoretic security.

First, there is a clear reduction of the security of this construction to the difficulty of the LPN problem. For many fuzzy extractor based systems, the computational hardness assumption is not clear.

Second, the extracted key is immediately usable as a cryptographic key since it has full entropy.

Third, we can correct a much higher error rate than information-theoretically secure fuzzy extractors, or prior computational fuzzy extractors for that matter. To observe this, first recognize that the released helper data $(\overrightarrow{a}_i, b_i)$ has *many* more bits than $\overrightarrow{s}$.[5] This would typically spell disaster in the information-theoretic case, but clearly this construction is secure assuming LPN is hard. This provides intuition for why we are able to correct for a much higher probability of error than is possible using traditional fuzzy extractors.

Recall that instead of requiring a small number of errors, we only require n 'stable' bits of biometric data. This is a much looser requirement on the noise of the biometric bits. It is possible to do this because the LPN problem is necessarily overdetermined, so the equations ($b_i = \langle \overrightarrow{a}_i, b_i \rangle + e_i$) have redundancy. The Project and Recovery algorithms leverage this redundancy to pick only the most stable bits (as indicated by the confidence information) for use in Gaussian elimination to dramatically improve recovery efficiency.

[5]As mentioned earlier, the $\overrightarrow{a}_i$ values can be stored in compact Read-Only Memory (ROM) or otherwise be part of the design.

As a result, this intuitively demonstrates that this construction will be much more resilient to higher error rates in the biometric data. This analysis is confirmed by physical data for RO PUFs that will be presented in sections V, VIII.

Fourth and finally, we recognize that our assumptions regarding the distributions on the biometric bits are much looser than is the case for many fuzzy extractor constructions (e.g., those that assume i.i.d.). In particular, as long as the bias on individual bits is bounded, LPN remains hard, and correlations between these bits cannot be discovered as long as $\overrightarrow{s}$ is never revealed. This property is critical for silicon biometric data such as PUFs, where there may be (and probably are) correlations between ring oscillator behavior depending on the location of the ring oscillator on the die/wafer/lot.

VII. Extensions

A. Malicious Helper Data

We observe that although one can store the helper information $(\overrightarrow{a_i}, b_i)$ in insecure memory with the PUF, this does not have to be the case. If the PUF is designed to receive the public helper information from an external source, we must consider the possibility of receiving malicious helper information. The adversary can repeatedly tweak the helper information and attempt to discover the stable bits by observing if key recovery fails or succeeds for each tweak. Key recovery will succeed if the b_i's that are modified are not used to recover the secret key.

An easy way of preventing this attack is to tweak the Gen algorithm to produce a hash of the public key keyed with $\overrightarrow{s}$ and store it along with the public key. The Recover algorithm attempts to regenerate $\overrightarrow{s}$, and is augmented to check that the keyed hash matches. If $\overrightarrow{s}$ is recovered correctly (because the subsystem of equations corresponding to 'stable' bits were not changed by the adversary), any modification of the public key will be detected with overwhelming probability. If a different $\overrightarrow{s}'$ is recovered (because the adversary changed one of the system of equations corresponding to a 'stable' bit), due to public key modification, the keyed hash will also not match with overwhelming probability.

B. Combination with Public Key Cryptosystems

One can easily use the key from the above construction as the input to a public key cryptosystem. For example the ECDLP problem of size n can be solved in $\mathcal{O}(2^{\sqrt{n}})$, so one may use the output $\overrightarrow{s}$ to generate the private key for ECDSA [23]. In particular, for 128-bit security, we require a uniformly randomly generated secret of size $n = 540$ bits for LPN. Therefore, one can simply use 256 of the bits for the 256-bit private key for ECDSA.

One could instead consider using the LPN-based public key cryptosystems as the signing/encryption algorithm. This would have the advantages of unifying the entire cryptosystem under a single computational hardness assumption as well as potentially allowing for re-use of hardware to compute the dot products and thresholds.

However, as noted in section VI, LPN-based public key cryptosystems suffer from requiring very small biases in the

e_i parameters [2], [6]. This, in turn, requires comparatively enormous n to achieve a security parameter of 128 [5], [11]. This immediately implies that one would not want to architect the above extractor to also function as a public key system (i.e., $\overrightarrow{s}$ is also the private key for the LPN-based public key cryptosystem). This is because using $\overrightarrow{s}$ as a private key would require n to be very large (on the order of 29000 [11] for a security parameter of 128), and correspondingly requires $m > n$ bits of biometric data, which is expensive.

Instead one could consider using the extracted $\overrightarrow{s}$ to decrypt (using AES or similar) the private key of a separate LPN cryptosystem. Unfortunately, the LPN cryptosystem will still require very high n for comparably small security parameter. Also, one is adding the assumption that AES is secure to the foundation of the security of the cryptosystem.

In conclusion, although our extractor is using the LPN hardness assumption as well as hardware implementing certain aspects of the LPN problem, it is likely more efficient (from a practical perspective) to couple the extracted key to a well-established public key cryptosystem such as EC or RSA, rather than try to leverage the existing LPN-related hardware resources of the construction as part of the cryptosystem.

C. Extension to LWE

The learning parity with noise problem is deeply related to the learning with errors problem. Therefore, it would make intuitive sense that our construction is extensible to leverage LWE instead of LPN.

This is indeed the case, but there are some aspects of the construction that become more complicated, so it is simpler to describe the LPN construction above and then remark on the changes to support LWE.

First, recognize that the majority of the hardware remains very similar. The $\overrightarrow{a}_i$ and $\overrightarrow{s}$ values would still be uniformly randomly chosen, only now each would have $\log(Q)$ bits (where Q is the modulus of the LWE cryptosystem). Indeed, the only challenge is to produce the e_i values correctly given the noise of the biometric data such that the extraction strategy discussed above still applies.

One simple extension of the above algorithm would be to generate e_i by concatenating order $\log_2 Q$ unique biometric bits. Recall that in LWE, Q is of order n^2, and n is on the order of the security parameter (remember that for a given security parameter k, the associated n for LWE is smaller than for LPN) [35].

For example, we choose $n = 128$, $Q = 2053$ based on the analysis in [29]. Note that in their example, the security parameter for these choices of n and Q is low, but we are considering the case where the e_i parameters have a much wider distribution. We find that increasing the variance of e_i to levels representative of our construction brings the security parameter to a reasonable level (just as in the LPN case).

Note that there are $\log_2 2053 = 11$ biometric bits per e_i. The primary complication is that we now must consider the probability of a stable e_i value given the stability of a biometric bit (see section V for details on the definition of 'stability'). In order for an e_i to be stable we need all 11 bits to be stable. This means that we need $n = 128$ groups of 11 bits to be stable. This is much more stringent than requiring 128×11 bits to be stable out of m total bits. If one follows the analysis of section V, we observe that such a system would require roughly $100\times$ more bits of biometric data than in the LPN case to recover the secret key with high confidence.

This is clearly not feasible. One area for future research would be into more efficient usage of biometric data in the LWE construction.

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

TABLE I
MEASURED BIAS OF 320 RO PUFS AT VARYING TEMPERATURES.

VIII. RESULTS USING A RING OSCILLATOR PUF

Although the LPN construction has applications that are much more broad than PUFs, we have noted that the LPN construction is well suited to address the problem of PUF key extraction because of the availability of 'confidence information' as described in section II-C.

We have provided a theory explaining the resilience of the LPN construction to noise and environmental parameters using this 'confidence information' in section V. Now, we use this theory and collected data from a set of 320 pairs of ring oscillator PUFs 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 $-40°\text{C}@0.95V$, $25°\text{C}@1.00V$, and $110°\text{C}@1.05V$. Other ranges that we will use are the differential count values at commercial (0°C to 70°C) and extended industrial ($-40°$C to 85°C). The $\sigma_{\text{INTRA}}/\sigma_{\text{INTER}}$ ratios improve as the temperature range is reduced.

We note that approximately 25% of the ring oscillator pairs produce different responses in the environmental range; this is the typical $O(m)$ error case for such circuits.

We first measured the bias of the RO counts across temperature as shown in table I. Therefore, our pessimistic estimate of bias as 40% (or 60% equivalently) is correct.

These differential count values are distributed according to the distribution discussed in section V with variance σ^2_{INTER}. We verified for each of these temperatures that the distribution of differential counts was Gaussian, as we assumed in section V. Each of the fits from which we derived parameters have a χ^2 value corresponding to $> 95\%$ confidence. Moreover, neither the mean nor variance of the distribution changed significantly over temperature or voltage. Therefore, we describe the distribution in terms of a single mean, variance (μ_{INTER}, σ_{INTER}) shown in figure 4.

To measure μ_{INTRA} and σ_{INTRA}, one must measure the distribution of how these differential counts change regardless

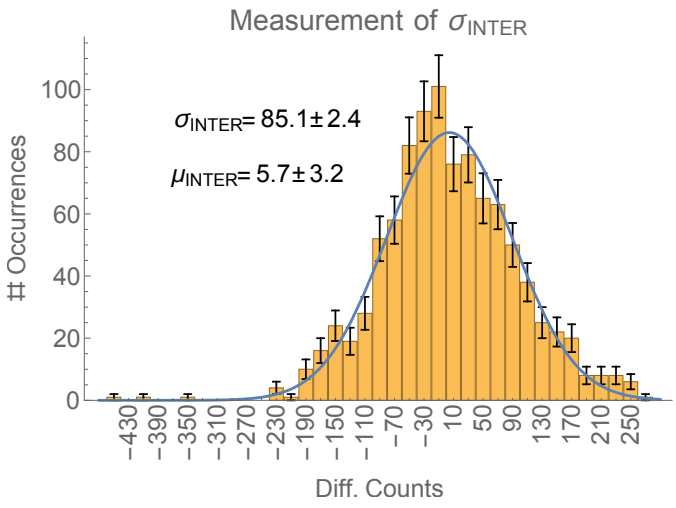

Fig. 4. Measurement of σ_{INTER} through the estimation of the distribution of differential counts across 320 RO pairs across room temperature and the fast and slow process corners.

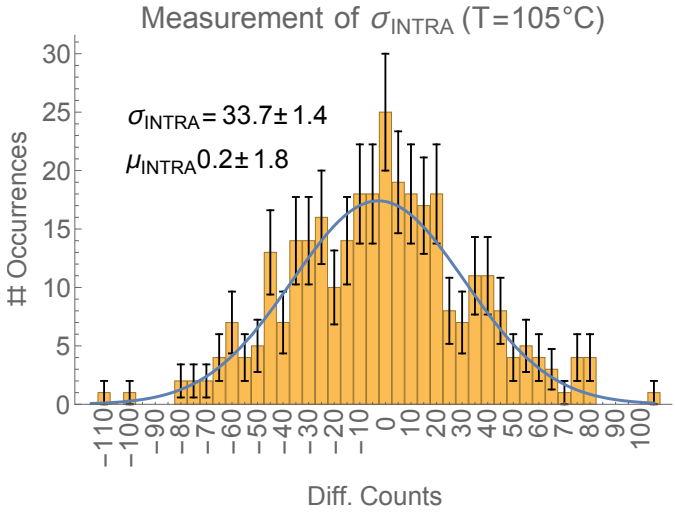

Fig. 5. Measurement of σ_{INTRA} by subtracting differential counts at $25°\text{C}@1V$ from $105°\text{C}@1.05V$.

of the differential count measured at provisioning. This distribution is $\text{Pr}(c_i' - c_i)$. 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 conditions in which the PUF would be provisioned), 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 V).

The distribution at $105°\text{C}$ is shown in figure 5. The measurements at various temperatures are shown in table II.

It is important to note that although we mentioned in section V that we do not present any theoretical justification for the reason why the distribution of counts of a single ring

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

TABLE II
MEASURED σ_{INTRA} FOR VARYING TEMPERATURES.

oscillator pair over relevant environmental conditions would be Gaussian, this does turn out to be the case. This is found to be true to within experimental error as demonstrated in figure 5.

Using these measurements, we now can calculate the ratio ($\frac{\sigma_{\text{INTRA}}}{\sigma_{\text{INTER}}}$) required in section V for commercial ($0°\text{C}$ to $70°\text{C}$) as 0.20, extended industrial ($-40°\text{C}$ to $85°\text{C}$) as 0.29, and the maximum temperature range our experiment could support ($-40°\text{C}$ to $105°\text{C}$) as 0.40.

Finally, we must choose ϵ_1, ϵ_2. We choose an error rate of $\epsilon_1 = \epsilon_2 \approx 10^{-6}$.

Using $n = 540$ (a security parameter of 128) and these values for ϵ_1 and $\sigma_{\text{INTRA}}/\sigma_{\text{INTER}}$ with equations 3, 4 we may compute m for commercial temperature ranges as ≈ 1850, extended industrial as ≈ 3700, and our max range as ≈ 11000.

Note that our analysis is pessimistic (e.g., the choice of the bias of 40% for the choice of n in section IV, choosing the bias to be 50% for the noise analysis in section V) and our construction is unoptimized. Even with an unoptimized implementation, these results compare favorably, e.g., 1850 helper data bits for commercial range of operation, with the works described in Section III. Given the clear security argument for the construction, we believe that our construction will be attractive to theoreticians and practitioners alike.

Some optimizations left to future work are:

1) In the results above, the number of Gaussian eliminations required is exactly 1, providing an exponential reduction in key recovery algorithm complexity through the use of confidence information. Choosing these values of m guarantee that if one chooses n ring oscillator pairs whose confidence/stability is over the chosen threshold, they will be stable to confidence $1 - \epsilon_1$. However, by choosing, for example, two disjoint or overlapping sets of n oscillators (at different thresholds) and attempting key recovery sequentially, one can reduce the failure probability for a given m or reduce m for a given failure probability.
2) Rather than choosing a threshold, one can sort the ring oscillators in decreasing confidence and pick the top n. This complicates the analysis, but will reduce failure probability, since the most stable bits will be selected.

IX. Conclusion

Fuzzy extractors have served as a useful construct to generate secret keys from noisy biometric sources. However, practical issues such as the level of noise, and the size of the helper data, have held back fully information-theoretically secure constructions, and security compromises are typically made. Computational fuzzy extractors are thus

attractive, provided they can address efficiency problems under established computational hardness assumptions. Prior work in computational fuzzy extractors has been unable to address the noise level issue, running in exponential time for $\Theta(m)$ errors. Especially in the silicon biometric source realm, e.g., Physical Unclonable Functions (PUFs), low level of noise is far from achievable.

We presented the first construction of a computational fuzzy extractor with a trapdoor in this paper. We use the Learning Parity With Noise (LPN) problem as the hard problem in our construction. While a construction under Learning With Errors is also feasible, LPN is particularly well-suited to PUFs. The trapdoor is unusual in that it is part of the biometric source and is used to avoid noise. Due to the exponential reduction in key recovery complexity enabled by the trapdoor, our construction is able to correct $\Theta(m)$ errors in polynomial time.

We relax the assumptions on the biometric source with respect to correlation showing that if correlation can be estimated, the only change to the construction is in the selection of parameters. Finally, we show how error profiles obtained from a Field Programmable Gate Array implementation of PUFs subject to wide environmental variation can be efficiently corrected.

REFERENCES

[1] AGRAWAL, S., BOYEN, X., VAIKUNTANATHAN, V., VOULGARIS, P., AND WEE, H. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *Proceedings of the 15th International Conference on Practice and Theory in Public Key Cryptography* (2012), PKC'12, pp. 280–297.

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

[3] APPLEBAUM, B., CASH, D., PEIKERT, C., AND SAHAI, A. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology - CRYPTO 2009*, S. Halevi, Ed., vol. 5677 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 595–618.

[4] ARORA, S., AND GE, R. New algorithms for learning in presence of errors. In *Automata, Languages and Programming*. Springer, 2011, pp. 403–415.

[5] BERNSTEIN, D. J., AND LANGE, T. Never trust a bunny. In *Radio Frequency Identification. Security and Privacy Issues*. Springer, 2013, pp. 137–148.

[6] BLUM, A., FURST, M., KEARNS, M., AND LIPTON, R. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology - CRYPTO 93*, D. Stinson, Ed., vol. 773 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1994, pp. 278–291.

[7] BLUM, A., KALAI, A., AND WASSERMAN, H. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM) 50*, 4 (2003), 506–519.

[8] BOGDANOV, A., KNEEVI, M., LEANDER, G., TOZ, D., VARC, K., AND VERBAUWHEDE, I. spongent: A lightweight hash function. In *Cryptographic Hardware and Embedded Systems CHES 2011*, vol. 6917 of *Lecture Notes in Computer Science*. 2011, pp. 312–325.

[9] BÖSCH, C., GUAJARDO, J., SADEGHI, A.-R., SHOKROLLAHI, J., AND TUYLS, P. Efficient helper data key extractor on fpgas. In *Proceeding Sof the 10th International Workshop on Cryptographic Hardware and Embedded Systems* (2008), CHES '08, pp. 181–197.

[10] BOYEN, X., DODIS, Y., KATZ, J., OSTROVSKY, R., AND SMITH, A. Secure remote authentication using biometric data. In *EUROCRYPT'05* (2005), pp. 147–163.

[11] DAMGÅRD, I., AND PARK, S. Is public-key encryption based on lpn practical? *IACR Cryptology ePrint Archive 2012* (2012), 699.

[12] DELVAUX, J., AND VERBAUWHEDE, I. Side channel modeling attacks on 65nm arbiter pufs exploiting cmos device noise. In *6th IEEE International Symposium on Hardware-Oriented Security and Trust - HOST 2013* (2013), pp. 137 – 142.

[13] DELVAUX, J., AND VERBAUWHEDE, I. Attacking puf-based pattern matching key generators via helper data manipulation. In *Topics in Cryptology CT-RSA 2014*, vol. 8366 of *Lecture Notes in Computer Science*. 2014, pp. 106–131.

[14] DODIS, Y., KANUKURTHI, B., KATZ, J., REYZIN, L., AND SMITH, A. Robust fuzzy extractors and authenticated key agreement from close secrets. *Information Theory, IEEE Transactions on 58*, 9 (Sept 2012), 6207–6222.

[15] DODIS, Y., REYZIN, L., AND SMITH, A. Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - Eurocrypt 2004* (2004).

[16] FULLER, B., MENG, X., AND REYZIN, L. Computational fuzzy extractors. In *Advances in Cryptology-ASIACRYPT 2013*. Springer, 2013, pp. 174–193.

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

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

[19] HILLER, M., MERLI, D., STUMPF, F., AND SIGL, G. Complementary ibs: Application specific error correction for pufs. In *IEEE Int. Symposium on Hardware-Oriented Security and Trust* (2012), IEEE.

[20] HILLER, M., WEINER, M., RODRIGUES LIMA, L., BIRKNER, M., AND SIGL, G. Breaking through fixed puf block limitations with differential sequence coding and convolutional codes. In *Proceedings of the 3rd International Workshop on Trustworthy Embedded Devices* (2013), TrustED '13, pp. 43–54.

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

[22] HOSPODAR, G., MAES, R., AND VERBAUWHEDE, I. Machine learning attacks on 65nm arbiter pufs: Accurate modeling poses strict bounds on usability. In *4th IEEE International Workshop on Information Forensics and Security (WIFS 2012)* (2012), pp. 37 – 42.

[23] JOHNSON, D., MENEZES, A., AND VANSTONE, S. The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security 1*, 1 (2001), 36–63.

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

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

[26] KIRCHNER, P. Improved generalized birthday attack. *IACR Cryptology ePrint Archive 2011* (2011), 377.

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

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

[29] LINDNER, R., AND PEIKERT, C. Better key sizes (and attacks) for lwe-based encryption. In *Topics in Cryptology–CT-RSA 2011*. Springer, 2011, pp. 319–339.

[30] MAES, R., TUYLS, P., AND VERBAUWHEDE, I. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems (CHES)* (2009), pp. 332–347.

[31] MAES, R., TUYLS, P., AND VERBAUWHEDE, I. Soft decision helper data algorithm for sram pufs. In *Proceedings of the 2009 IEEE International Conference on Symposium on Information Theory - Volume 3* (2009), ISIT'09, pp. 2101–2105.

[32] MAES, R., VAN HERREWEGE, A., AND VERBAUWHEDE, I. Pufky: A fully functional puf-based cryptographic key generator. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems* (2012), CHES'12, pp. 302–319.

[33] MATHEW, S. K., SATPATHY, S. K., ANDERS, M. A., KAUL, H., HSU, S. K., AGARWAL, A., CHEN, G. K., PARKER, R. J., KRISHNAMURTHY, R. K., AND DE, V. 16.2 a 0.19 pj/b pvt-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm cmos. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International* (2014), IEEE, pp. 278–279.

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

[35] REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM) 56*, 6 (2009), 34.
[36] RÜHRMAIR, U., DEVADAS, S., AND KOUSHANFAR, F. Security based on physical unclonability and disorder. In *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. Springer, 2012, ch. 4, pp. 65–102.
[37] RÜHRMAIR, U., SEHNKE, F., SÖLTER, J., DROR, G., DEVADAS, S., AND SCHMIDHUBER, J. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security (CCS)* (2010), ACM, pp. 237–249.
[38] RUHRMAIR, U., SOLTER, J., SEHNKE, F., XU, X., MAHMOUD, A., STOYANOVA, V., DROR, G., SCHMIDHUBER, J., BURLESON, W., AND DEVADAS, S. Puf modeling attacks on simulated and silicon data. *Information Forensics and Security, IEEE Transactions on 8*, 11 (Nov 2013), 1876–1891.
[39] SAHAI, A., AND WATERS, B. Fuzzy identity-based encryption. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques* (2005), EUROCRYPT'05, pp. 457–473.
[40] SUH, G. E. *AEGIS: A Single-Chip Secure Processor*. PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science., Aug. 2005.
[41] SUH, G. E., AND DEVADAS, S. Physical unclonable functions for device authentication and secret key generation. In *ACM/IEEE Design Automation Conference (DAC)* (2007).
[42] VAN DER LEEST, V., PRENEEL, B., AND VAN DER SLUIS, E. Soft decision error correction for compact memory-based pufs using a single enrollment. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings* (2012), pp. 268–282.
[43] YU, M.-D. M., AND DEVADAS, S. Secure and robust error correction for physical unclonable functions. *IEEE Design and Test of Computers 27* (2010), 48–65.
[44] YU, M.-D. M., M'RAIHI, D., SOWELL, R., AND DEVADAS, S. Lightweight and secure PUF key storage using limits of machine learning. In *Cryptographic Hardware and Embedded Systems (CHES)*. 2011, pp. 358–373.
[45] YU, M.-D. M., SOWELL, R., SINGH, A., M'RAIHI, D., AND DEVADAS, S. Performance metrics and empirical results of a PUF cryptographic key generation ASIC. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* (2012), pp. 108–115.

APPENDIX

SECURITY PARAMETER DERIVATION

In section IV, we derived the n (the length of $\overrightarrow{s}$) based on the security parameter k using known cryptanalysis of the LPN problem [35], [7], [5].

The purpose of this appendix is to provide more detail on the derivation of n from k.

We follow the concrete attack approach put forth in [5], which combines ideas from canonical LPN attacks [7], [26], [28].

We restrict the memory of the attack to 2^{23}, as this corresponds to roughly 10MB of memory which must be accessed at a single-cycle latency.

The algorithm in [5] requires parameters of a, b, l, W, and q. These are new parameters that are not related to any other variables in this paper.

In particular, we optimize over $1 \leq a \leq 100$, $1 \leq b \leq 100$, $1 \leq l \leq 10$, and $1 \leq W \leq 4$, and $1 \leq q \leq 500$.

We find that for $n = 540$, $\tau = 0.40$, the expected number of bit operations is on the order of 2^{131} for parameters in table III.

Using the breakdown from [5], we recognize that an individual iteration is required of several steps: The extraction of the required additional queries requires $2^{23.7}$ bit operations. The filtering/clearing step requires roughly $2^{26.3}$ bit operations. The Walsh transform requires roughly $2^{25.7}$ bit operations. Finally, the dot product computation dominates with roughly $2^{33.6}$ bit operations.

Therefore, a single iteration of the algorithm requires roughly $2^{33.7}$ bit operations. However, one iteration of the algorithm has probability $2^{-97.7}$ of success, and so it must be repeated an expected number of $2^{97.7}$ times.

This yields an overall expected number of bit operations equal to 2^{131}.

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

TABLE III
TABLE OF PARAMETERS FOR [5] THAT MINIMIZE COMPUTATION TIME FOR 10MB MEMORY LIMIT ACHIEVING A SECURITY PARAMETER OF 128. SINCE WE ARE NOT USING LPN IN A CRYPTOSYSTEM WE CAN HAVE A LARGE VALUE FOR τ.