Systematic Low Leakage Coding for Physical Unclonable Functions

Matthias Hiller
Institute for Security in Information Technology, Technische Universität München
matthias.hiller@tum.de

Meng-Day (Mandel) Yu
Verayo, Inc.
COSIC, KU Leuven
and CSAIL, MIT
myu@verayo.com

Michael Pehl
Institute for Security in Information Technology, Technische Universität München
m.pehl@tum.de

ABSTRACT

Physical Unclonable Functions (PUFs) derive unique secrets from internal manufacturing variations in integrated circuits. This work shows that key generation with PUFs is a practical application of the generic information theoretic problem of secret key agreement with a compound source.

We present an improved secure sketch construction with our new optimal syndrome coding scheme for PUFs, Systematic Low Leakage Coding (SLLC). Our scheme provides inherent information theoretic security without the need of a hash function or strong extractor, and optimal asymptotic performance concerning maximum key size and minimum helper data size. The secrecy leakage is bounded by a small epsilon that goes to zero for sufficiently good PUFs.

The reference implementation for an ASIC application scenario shows that our scheme does not require the 47% hardware overhead for the hash function that is mandatory for the state-of-the-art approaches.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection
; E.4 [**Coding and Information Theory**]: Error Control Codes

Keywords

Physical Unclonable Functions; Information Theory; Fuzzy Extractor; Secure Sketch; Syndrome Coding; Secret Key Generation; Compound Source; ASIC.

1. INTRODUCTION

Secure and reliable cryptographic keys are a prerequisite to apply cryptography in embedded systems, operating in unprotected environments. Physical Unclonable Functions (PUFs) [1, 2, 3, 4, 5, 6, 7] are circuits that measure manufacturing variations to derive a unique secret inside a device to use it as a cryptographic key or to authenticate a device. One main advantage of PUFs is that they provide secure keys for circuits manufactured in standard CMOS technology that do not have access to state-of-the art secure storage solutions such as secure flash memory [8]. This makes PUFs especially promising for lightweight cryptographic scenarios with resource constrained devices [9].

Similar to measuring biometric patterns in nature, silicon PUFs generate unique patterns inside chips during runtime. PUF circuits are designed to amplify the effects of manufacturing variations in the circuit and extract a unique response pattern for each device. These PUF circuits can be modeled mathematically as random variables with unique properties for each chip that return PUF responses as outcomes. PUF responses are sensitive to measurement noise, environmental changes and aging. So, the PUF responses vary slightly between different measurements, and also over time. Almost all existing PUF types need additional algorithmic support to generate keys that fulfill cryptographic requirements and have a sufficiently low error probability over their entire lifetime.

To derive a secure key from a PUF, we have to ensure that the PUF response has a reasonable amount of entropy. The generation procedure creates helper data that is used later to reconstruct the key in the reproduction phase. The helper data is public so that it can be observed by the adversary. Algorithms for secret key generation with PUFs are a vivid research topic in the hardware security community [3, 10, 11, 12, 13, 14, 15, 16, 17, 18].

Improvements on algorithmic level and new implementations pushed the limits in efficiency and reliability. Some key generation algorithms for PUFs can be seen as *secure sketches* [19]. In general, the current approaches typically require at least two algorithmic components for the post-processing of the raw PUF data:

- *Syndrome coding* schemes address the issue to create public helper data that enables error correction, and for some approaches also the first error correction steps are carried out in the syndrome decoder [12, 14, 15]. During *generation*, public helper data is created. Later during *reproduction*, the cryptographic key is computed from a noisy PUF response and the helper data. The challenge in creating helper data lies in the contradicting interests of revealing information about the PUF that permits error correction, while keeping the parts used for the cryptographic key secret.

ASIA CCS'15, April 14–17, 2015, Singapore.

http://dx.doi.org/10.1145/2714576.2714588

- An *error-correcting code* [20] corrects the remaining errors in the output of the syndrome decoder. Here, it is important that the code is designed to reach the target bit error probability of the key for a given application scenario and PUF. Strict hardware constraints on the size of the decoder pose new challenges on algorithmic and implementation level to design specific low area decoders for this purpose.

We focus on syndrome coding in this work and assume that optimal error-correcting codes with specific properties are given. The construction and efficient implementation of error-correcting codes is out of the scope of this paper.

In [12, 14, 15], it has been proven that the discussed pointer based syndrome coding solutions are information theoretically secure for sufficiently good PUFs. Therefore, the outcome of the error-correcting code can be output directly as a cryptographic key. In contrast, the secure sketches in [19] have to be extended to fuzzy extractors with an additional hash function to derive a cryptographically secure key.

Note that all current approaches have a downside that they either discard PUF response bits so that they do not extract all available secret information or that they are only cryptographically and not information theoretically secure. In this work, we will present a method that overcomes these shortcomings and shows optimal performance and information theoretic security at the same time.

Research in the information theory community led to a better understanding of different key agreement scenarios and adversary models on an abstract level [21, 22, 23, 24, 25, 26]. As one typical problem in information theory, a source returns two correlated output sequences. Two legitimate parties observe one sequence each and derive a common secret by exchanging messages over a public communication channel. Analyzing the asymptotic behavior of this abstract problem gave new insights to understand its fundamental properties, especially the limits of how much key information can be extracted and how much public communication is necessary. Our work shows that secret key generation with PUFs can be seen as a practical use case of the abstract scenario of secret key agreement from correlated sequences drawn from a very recent source model, called *compound source* [26].

In the same work ([26]), Boche and Schäfer introduced an optimal key agreement protocol for a compound source using random code constructions. Random codes show optimal theoretical properties, but are in general hardly suitable for implementation in embedded devices since large random codebooks have to be stored. Whereas previous theoretical work such as [23] only considers aspects of the key, [26] also took the size of the transmitted data into account. Based on this work, we will present a deterministic coding scheme with similar properties that can be implemented in practice.

To introduce our new approach on a solid theoretical basis, we will go from the abstract to a more concrete level, and finally into a practical setting, over the sections of this work.

Our contributions

- This work provides the first information theoretic model that fully represents the problem of secure key generation with PUFs. We translate the practical problem of key generation with PUFs directly into information theoretic models and highlight the corresponding parts as well as possible simplifications.

- Our new syndrome coding scheme *Systematic Low Leakage Coding* (SLLC) is the first optimal syndrome coding scheme that fulfills common requirements in information theoretic security. Using error-correcting codes with systematic encoding separates the codeword into an information and a redundancy part. Storing only helper data for the redundancy part minimizes the helper data size and brings the leakage of the key through the helper data close to zero for sufficiently good PUFs. Returning only the information part as key ensures information theoretic security. In addition, reference secure sketch constructions are compared to SLLC on an information theoretic level.

- The implementation part sketching a lightweight ASIC implementation with a small BCH decoder shows that the low secrecy leakage of SLLC has significant benefits in practice. It enables information theoretic security, removes the area-intensive hash function that is necessary for other approaches and minimizes the helper data size.

Outline

The first contribution of the paper is to reveal the correspondence between the practical problem of secret key generation with PUFs and the theoretical compound source model in Section 2. Section 3 introduces the mathematical notation used in the paper. The information theoretic preliminaries and definitions are presented in Section 4. We discuss the existing optimal random coding scheme in Section 5 and present our new deterministic scheme SLLC in Section 6. The evaluation in Section 7 compares our new scheme to popular secure sketch constructions and Section 8 discusses SLLC in relation to the state of the art in a practical setting. Section 9 draws our conclusions over this work.

2. EQUIVALENCE OF SECURE KEY GENERATION WITH PUFS AND KEY AGREEMENT FROM COMPOUND SOURCES

In this section, we show that secure key generation with PUFs is equivalent to key agreement from correlated sources as shown in Figure 1 from a high-level point of view. In contrast to previous work [27, 23], the compound source provides a more detailed model because the effects of various physical parameters can be represented as states of the source.

For secure key generation with PUFs, a PUF behaves like a random variable that is evaluated in the generation procedure to compute the helper data. Later on, the PUF acts like a correlated random variable, depending on the random variable and the environmental conditions. It is evaluated multiple times in the field to generate the cryptographic key whenever it is needed.

In the information theoretic setting, two parties, Alice and Bob, both have access to one output sequence of a joint source of randomness. The sequences are drawn from a joint probability distribution such that they are correlated, but in general not identical. Therefore, error correction is necessary. Alice corresponds to the PUF during the generation

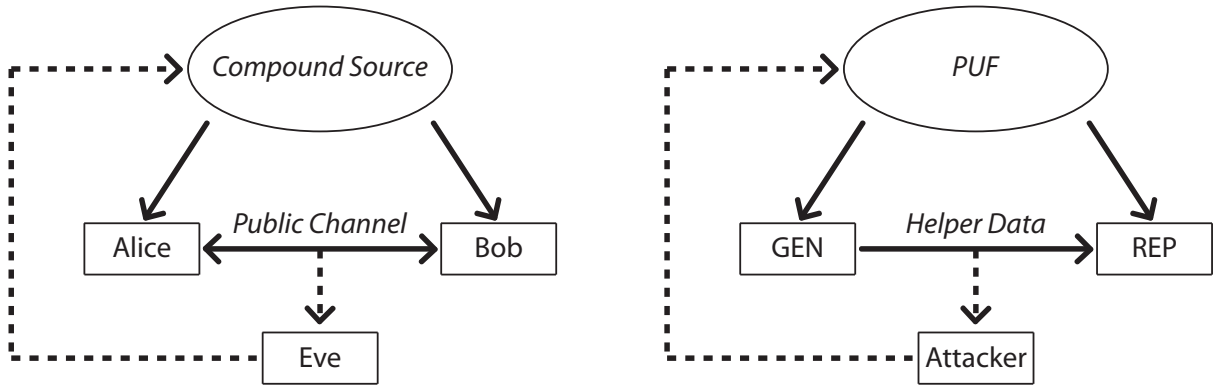

Figure 1: Analogies between the key agreement from a compound source and secret key generation from a PUF

	Information Theoretic Model	Key Generation with PUFs
Source of randomness	Compound Source	PUF
Communication between parties	Public Communication Channel	External Helper Data
Influence of attacker	State of the Compound Source	Operating Conditions of PUF
Distance between parties	Space	Time

Table 1: Comparison of key generation with a PUF and secret key agreement with a compound source

procedure in the manufacturing environment and Bob to the PUF during the reproduction procedure in the field.

The following four criteria show the most prominent equivalents between the two problems:

Source of randomness.

Generating and reproducing one PUF key can be seen as one key agreement between Alice and Bob in the information theoretic scenario, with the difference that Bob can also transmit data to Alice whereas the PUF key reproduction takes place later and we cannot go back in time.

Reading out a PUF twice under different environmental conditions can be interpreted as Alice and Bob evaluating correlated sequences. However, we assume that the helper data of a PUF is generated in a fixed and protected manufacturing environment that is not accessible to the attacker. Therefore, we can remove the state dependency in the sequence of Alice.

The state depends on the environmental conditions and age of the PUF and defines the distribution of the random variable observed by Bob. The random process that determines which actual outcome is drawn from this distribution corresponds to thermal and measurement noise.

Communication between parties.

Storing external helper data permits a communication from helper data generation to key reproduction. In an information theoretic sense, this is equivalent to a noiseless one-way channel from Alice to Bob. Mostly, a bidirectional communication between Alice and Bob is permitted in the theoretical setup. However, the optimal protocol in [26] only requires one-way communication.

Influence of attacker.

We assume that the attacker can determine the operating conditions of the PUF by setting the external voltage, operating the PUF in an environment with a specific temperature, or aging the device. The operating conditions correspond to the state of the compound source. Further, the attacker can read out the helper data, which corresponds to an adversary eavesdropping the public communication.

Distance between parties.

Typically, communication is carried out between two parties that are separated in space using for example a wireless or cable connection. In the PUF scenario, two PUF responses are separated in time. For an adversary with unlimited computational power in the theoretical setting, both scenarios have the same properties.

In practice, a wireless transmission over space can have stricter timing constraints for the computation for the adversary. However, this is more important for other scenarios where an active wiretapper manipulates the public communication between Alice and Bob, which is not considered in this work.

Note that the discussed work does not check, whether the reproduced key is correct or not. See e.g. [28] if the authenticity of the key also has to be ensured.

In a use case, the information theoretic protocol is run several times to establish new session keys between Alice and Bob. Therefore, both parties agree on a new key each time the protocol is carried out. This corresponds to the manufacturing process of PUFs, where multiple different PUFs are manufactured. The source statistics of the PUFs are identical, but the actual sequences and the derived keys vary.

For a single PUF, one set of helper data is generated and then the same key is reproduced multiple times. Alice's

part is only executed once so that Alice's sequence from the source, the derived key, and the public helper data remain constant over the lifetime of the PUF. Only Bob's source sequence and his reproduced key can vary from key generation to key generation.

Taking the previous points into consideration, the PUF scenario can be seen as a typically more resource restricted special case of the generic information theoretic model of secure key agreement from a compound source. See Table 1 for a compact representation of the results.

3. NOTATION

Random variables are denoted in capital italic letters, e.g. X. Scalars, such as outcomes of random variables, are written in small italic letters, e.g. x. Calligraphic letters indicate sets and $|\mathcal{X}|$ is the cardinality of the set $\mathcal{X}$.

Further, a superscript over a letter, e.g. X^n, denotes a vector of n instances of X. X_i^j selects elements i to j of X^n. $P_X(x)$ denotes the probability distribution or, more formally, the probability mass function, of X for $x \in \mathcal{X}$. Matrices are written in bold capital letters. Let $\mathbf{A}^T$ be $\mathbf{A}$ transposed. Concatenations are indicated with two vertical lines ($||$).

For random variables X and Y, $X|Y$ means X under condition Y. Let $H(X)$ stand for the Shannon entropy of X and $H(XY)$ for the joint entropy of X and Y, and let $I(X;Y)$ be the mutual information between X and Y [29].

For a better readability, we will switch between integer representations of numbers and their binary representations in $GF(2^n)$ without marking the binary representation explicitly. In cases that require special emphasis, the binary representation of i in $GF(2^n)$, $n \in \mathbb{Z}^+$, is denoted with $b_n(i)$.

4. INFORMATION THEORETIC PRELIMINARIES AND DEFINITIONS

In this work, we apply the compound source model for the first time to PUFs. Therefore, we present the required models and definitions in this section. Figure 2 shows two parties, Alice and Bob, that want to agree on a mutual secret key while keeping the adversary Eve ignorant. A compound source is a source with multiple correlated outputs and an internal state s selected from set $\mathcal{S}$. Alice and Bob observe different outputs of the same discrete memoryless multiple compound source [26] and use the outputs of the source to generate a common secret. Memoryless means that the n output pairs (x, y) of the source are drawn independently. Empirical entropy evaluations of PUFs, e.g. [30], show that this assumption also holds in practice for popular PUF types like the SRAM PUF so that we can apply this theory.

In the following, we will consider the compound source shown in Figure 2. It has two generic outputs X_s and Y_s with joint probability distribution $P_{XY,s}(x, y)$ depending on the state $s \in \mathcal{S}$. The state s can be set by the adversary Eve. X_s and Y_s with marginal distributions $P_{X,s}(x)$, $P_{Y,s}(y)$ are not accessible to the adversary.

Alice and Bob cannot observe both random variables at once. To establish a joint secret, Bob can observe his marginal distribution $P_{Y,s}(y)$ to make an estimate of s. He can further make a rough estimate of X_s by observing Y_s.

This behavior is represented mathematically as probabilistic memoryless channel $\mathcal{T}_s$ between Alice and Bob with input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$. The effect of the channel is characterized by the conditioned probability distribution $P_{Y|X,s}$, since the output of the channel Y_s depends on the input X_s. For a specific $s \in \mathcal{S}$, the channel $\mathcal{T}_s$ is given by the set of the conditional probabilities for all (x, y) pairs.

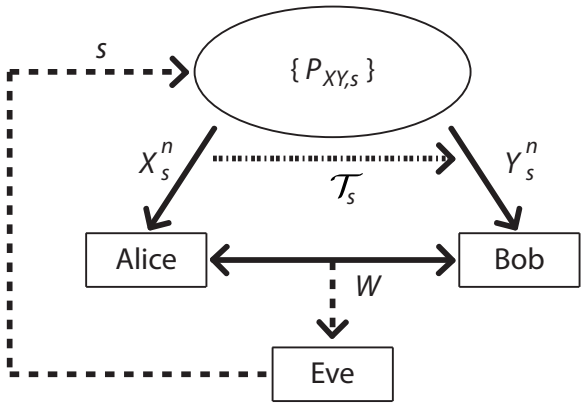

Figure 2: Secret key generation with a compound source

$$\mathcal{T}_s = \left\{ P_{Y|X,s}(x,y) = \frac{P_{XY,s}(x,y)}{P_{X,s}(x)} : x \in \mathcal{X}, y \in \mathcal{Y} \right\} \quad (1)$$

Public communication, or helper data, between Alice and Bob supports this estimation. Alice establishes secret key K and Bob secret key L from the same key space $\mathcal{K}$, with $K, L \in \mathcal{K}$, after observing the two sequences ${X_s}^n$ and ${Y_s}^n$ of length n drawn from the joint probability distribution $P_{XY,s}(x, y)$ and exchanging data over the public channel.

In Section 2, we have shown that this information theoretic model corresponds to the practical behavior of a PUF with an initial response ${X_s}^n$. In contrast to previous models, the compound source can represent that subsequent responses ${Y_s}^n$ depend on environmental factors such as temperature, voltage or aging that can be potentially controlled by the attacker.

The public communication, that is typically carried out over space in a communication scenario, corresponds to storing the helper data W in external non-volatile memory that is accessed again later in time to reproduce a key. A secret key K is established during manufacturing and later on, key L is reproduced in the field. We aim for a reliable key generation protocol, so $k = l$, with $K \mapsto k$ and $L \mapsto l$, should hold with a high probability.

Definition of an achievable rate

A *key rate* R_{key} specifies the amount of secret information that can be derived reliably from each (x, y) pair. This can be interpreted as the ratio between key length and PUF size. Referring to Ahlswede and Csiszar [21], Boche and Schäfer defined in [26] that a key rate R_{key} is an *achievable key rate*, if for any $\epsilon > 0$ and n_ϵ there exists a key generation protocol with source output length $n > n_\epsilon$ where K and L satisfy the

following four conditions for every $s \in \mathcal{S}$:

$$\Pr[k \neq l] < \epsilon \tag{2a}$$
$$I(W;K) < \epsilon \tag{2b}$$
$$\frac{1}{n}H(K) > R_{key} - \epsilon \tag{2c}$$
$$\frac{1}{n}\log_2|\mathcal{K}| < \frac{1}{n}H(K) + \epsilon \tag{2d}$$

Condition 2a ensures the reliability of the generated key K, or more formally that the probability that an incorrect key L is reproduced, is smaller than ϵ. From a security point of view, it is important that no significant amount of key information K leaks through the helper data W, so according to Condition 2b the helper data leaks less than ϵ about the key. The rate R_{key} quantifies the performance of the scheme. The rate is the central criterion that quantifies this performance. For a meaningful measure, the amount of actual key information $H(K)$ has to be close to the specified rate R_{key}, as stated in Condition 2c. Bounded by ϵ, Condition 2d states that the key space $\mathcal{K}$ is only slightly larger than the entropy of the key $H(K)$, such that the generated key has good cryptographic quality.

Capacity definitions

In information theory, a capacity is defined as the best achievable rate. Depending if a capacity C is defined as upper or lower bound of a measure, it is the supremum or infimum over all achievable rates R.

The key capacity C_{key} of a compound source is defined as the supremum over all achievable key rates R_{key}. The authors in [26] show that the key capacity of the compound source is given by the minimum mutual information between both outcomes of the source over all possible states such that

$$C_{key} = \sup_{R_{key} \text{ is achievable rate}} R_{key} \tag{3}$$
$$= \min_{s \in S} I(X_s, Y_s) \tag{4}$$

Reconstructing X_s from Y_s with joint distribution $P_{XY,s}$ requires in average $H(X_s|Y_s)$ bits. The helper data capacity C_{hd} is the minimum helper data size that has to be assigned to be able to restore X_s from Y_s for any state of the source. Therefore, the minimum helper data size is given by

$$C_{hd} = \max_{s \in S} H(X_s|Y_s) \tag{5}$$

See [26] for the proofs.

These theoretical bounds permit to evaluate schemes not only in comparison to previous work. They also give designers quantitative information on how far a scheme can still be improved until the absolute limits are reached.

Code definition

Typically, error-correcting codes are defined by the code length n, the code size (or number of information bits) k and the minimum distance between any two codewords d [20]. Sometimes, also the minimum number of correctable errors $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ is used. Code $\mathcal{C}$ is defined as set containing all codewords C_i, $i = 1, ..., 2^k$.

The Channel Coding Theorem [29] shows that there exist codes for a channel $\mathcal{T}_s$ with capacity $C_{\mathcal{T}}$ where the probability of a decoding error is smaller than a small $\epsilon > 0$ if the code length n is larger than an n_ϵ for every ϵ and rate $R_{Code} = \frac{k}{n} < C_{\mathcal{T}}$.

Similar to [26], we define codes with code length n, code size k and maximum probability of a decoding error for a given channel as (n, k, ϵ) codes for the theoretical part. For the practical part, we will use (n, k, d) codes.

5. AN OPTIMAL SYNDROME CODING SCHEME WITH RANDOM CODES

Boche and Schäfer introduced an optimal combined syndrome coding and error correction construction for a compound source in [26]. In this work, we are the first who apply this approach to PUFs. The scheme creates a large number of random codes such that any ${X_s}^n$ is a codeword of one of the codes with probability close to one. Alice transmits the number of the code as helper data to indicate Bob which code to use for decoding ${Y_s}^n$. The adversary Eve only knows that ${X_s}^n$ is one of the 2^k codewords of the code with the number that is transmitted. However, she received no information which codeword is equal to ${X_s}^n$. Therefore, this information can be used as secret key.

For the sake of simplicity, we will only discuss the use case for PUFs where X^n is evaluated in a controlled manufacturing environment in which the state of the source is fixed. To achieve error probabilities $< \epsilon$ for all possible states and a small $\epsilon > 0$, we use (n, k, ϵ) error-correcting codes with rate

$$R_{Code} = \frac{k}{n} = \max_{s \in S} H(X|Y_s) \tag{6}$$

Before running the actual key agreement process, m random (n, k, ϵ) codes $\mathcal{C}_i$ $(i = 0, ..., m-1)$ with disjoint codewords are created. The codes $\mathcal{C}_i$ are created in sequential order, starting with code $\mathcal{C}_0$. As code generation procedure of codeword C_j $(j \in 0, ..., 2^k - 1)$ of code $\mathcal{C}_i$, we draw an output sequence of length n and with distribution P_X from the source (PUF). If C_j is not already a codeword of the code $\mathcal{C}_i$ $(C_j \notin \mathcal{C}_i = \{C_0, ..., C_{j-1}\})$ or any other previously generated code $\mathcal{C}_0$,...,$\mathcal{C}_{i-1}$ $(C_j \notin \bigcup_{l=0}^{i-1} \mathcal{C}_l)$, the new codeword C_j is added to $\mathcal{C}_i$. Otherwise, the sequence is discarded and a new sequence is drawn. This process is continued until all m codes contain 2^k codewords.

The union over all codes covers the most likely output sequences X^n, so if the number of codes m is sufficiently large

$$\Pr[X^n \in \bigcup_{i=0}^{m-1} \mathcal{C}_i] > 1 - \eta \tag{7}$$

for a small $\eta > 0$. Note that the codebooks are public, so also accessible to the adversary Eve.

Starting with the actual key agreement, Alice draws the sequence X^n from the source. She transmits (or the PUF saves) the index i of the selected code $\mathcal{C}_i$ as helper data W such that

$$W(X^n) = \begin{cases} i, & \text{if } X^n \in \mathcal{C}_i, \text{and } i \in \{0, ..., m-1\} \\ 0, & otherwise \end{cases} \tag{8}$$

According to Eqn 7, the probability of the "'otherwise"' case in Eqn. 8 is bounded by η so that this helper data generation process is sufficiently reliable.

All codes are (n, k, ϵ) codes for the given channel, so Bob (or the PUF in the field) can reconstruct the correct key L from W and ${Y_s}^n$ with a small error probability such that

$$\Pr[k \neq l | X^n \in \mathcal{C}_i] < \epsilon \tag{9}$$

Considering both error events given in Eqns. 7 and 9 leads to an overall error probability of

$$\Pr[k \neq l] < \epsilon + \eta \tag{10}$$

The proofs in [26] show that this approach is capacity achieving and that the security condition, Condition 2b in the definition, is satisfied with $I(W; K) = 0$ for i.i.d. PUF response bits.

Random codes show properties of (n, k, ϵ) codes, so the error correction problem is solved in theory. However, Alice and Bob have to store all random codebooks, which makes this approach infeasible in practice, especially for PUFs when they are used in resource constrained lightweight embedded systems.

6. SYSTEMATIC LOW LEAKAGE CODING FOR PUFS

In this section, we introduce a capacity achieving syndrome coding construction for PUFs, inspired by the random coding construction discussed in the previous section. As main advantage over the approach discussed in the previous section, our new syndrome coding scheme does not have to store any random codebooks. This facilitates implementations in practice.

We replace the random codebook generation with a deterministic procedure where all codes are derived from one parent code. In the following, we also assume that (n, k, ϵ) code $\mathcal{C}$ achieves a rate of R_{Code} for the channel such that decoding errors occur with a probability $< \epsilon$. We assume that there exists a systematic encoding scheme for code $\mathcal{C}$ such that for all codewords $C \in \mathcal{C}$, the first k codeword bits are equal to the information bits $c_1^k = x_1^k$.

This assumption holds for many popular code classes, such as BCH, Reed-Solomon, convolutional, LDPC codes and many other code classes, see e.g. [20].

SLLC code construction

In the following, we use one (n, k, ϵ) block code with systematic encoding and create all other codes as cosets of the basic code. For the i-th coset of the code $\mathcal{C}$, the binary representation of i is XORed on the last $n-k$ bits of each codeword to create code $\mathcal{C}_i$. Iterating over i from 1 to $2^{n-k} - 1$, we are able to assign exactly one code $\mathcal{C}_i$ to each element in $GF(2^n)$.

Let $\mathbf{I}_{n-k}$ be the $((n-k) \times (n-k))$ identity matrix and $\mathbf{A}$ a $((n-k) \times k)$ matrix. For systematic encoding, the code $\mathcal{C}$ has a parity check matrix $\mathbf{H}$ in the form [20]

$$\mathbf{H} = (\mathbf{A} || \mathbf{I}_{n-k}) \tag{11}$$

This leads to a generator matrix $\mathbf{G}$ with

$$\mathbf{G} = (\mathbf{I}_k || \mathbf{A}^T) \tag{12}$$

Let φ_n: $GF(2^k) \mapsto GF(2^n)$ be the encoder of the code $\mathcal{C}$ and $b_k(l)$ the binary representation of l in $GF(2^k)$. For our new approach, we define the code $\mathcal{C}_0$ as

$$\mathcal{C}_0 = \left\{ \varphi_n(b_k(l)) \oplus (0^k || b_{n-k}(0)) \ : \ l = 0, ..., 2^k - 1 \right\} \tag{13}$$

We derive all other codes $\mathcal{C}_i$, $i = 1, ..., 2^k - 1$ as follows:

$$\mathcal{C}_i = \left\{ \varphi_n(b_k(l)) \oplus (0^k || b_{n-k}(i)) \ : \ l = 0, ..., 2^k - 1 \right\} \tag{14}$$

Given the n-bit PUF response X^n, the helper data W^{n-k} is generated by storing code index i.

$$W^{n-k} = b_{n-k}(i) = \left[\varphi_n(X^k) \oplus X^n \right]_{k+1}^n \tag{15}$$

Since for all systematic codes,

$$[\varphi_n(X^k)]_1^k = X_1^k \tag{16}$$

the $n-k$ least significant bits return the binary representation of i. This representation separates the secret key part X_1^k from the redundancy part X_{k+1}^n. The operation $\varphi_n(X^k) \oplus X^n$ can be interpreted as masking the redundancy $[\varphi_n(X^k)]_{k+1}^n$, that leaks key information, with unused PUF bits X_{k+1}^n.

Using indices from 0 to $2^{n-k} - 1$, we are able to cover the entire output space X^n such that there exists a code $\mathcal{C}_i$ for all $X^n \in GF(2^n)$ with $X^n \in \mathcal{C}_i$. We can guarantee that $X^n \in \bigcup_{i=0}^{m-1} \mathcal{C}_i$ because $\bigcup_{i=0}^{m-1} \mathcal{C}_i = GF(2^n)$. Therefore, Eqn. 7 holds with $\eta = 0$.

In contrast to the state-of-the-art secure sketches, where a hash function is mandatory, the first k bits of the corrected PUF response can be used directly as secret key without hashing if $H(X^k) > k - \epsilon$ and $I(X^k; X_{k+1}^n) < \epsilon$. If the entropy of X^k is lower, a standard data compression function g_l, see e.g. [29], can be used to compress the corrected PUF response to a key $K = g_l(X^k)$ such that $H(K) > l - \epsilon$ for a small $\epsilon > 0$.

During reproduction, $\hat{Y}^k$ is reconstructed from ${Y_s}^n$ and $W^{n-k} = b_{n-k}(i)$. Let $\gamma_k = {\varphi_n}^{-1}$: $GF(2^n) \mapsto GF(2^k)$ be the decoder of the code $\mathcal{C}$.

$$\hat{Y}_s^k = \gamma_k({Y_s}^n \oplus (0^k || b_{n-k}(i))) \tag{17}$$

Remark

In a typical communications scenario, cosets are used to characterize errors. All vectors in the j^{th} coset have the same error pattern $b_n(j) \in GF(2^n)$, $j \in \{1, ..., 2^{n-1}\}$, that is added to each codeword. For bounded minimum distance decoding [20], the decoder can correct errors by finding j if the Hamming weight of $b_n(j)$ is bounded by

$$wt(b_n(j)) \leq \lfloor \frac{d-1}{2} \rfloor \tag{18}$$

In contrast, we use $b_{n-k}(i)$, $i \in \{1, ..., 2^{n-k} - 1\}$, to modify the last $n-k$ bits of codewords to generate the i^{th} subcode. Then, $b_{n-k}(i)$ is transmitted as side information to the decoder. We can represent ${Y_s}^n$ as

$$Y_s{}^n = X^n \oplus b_n(j) \tag{19}$$

Using $X^n = \varphi_n(X^k) \oplus (0^k || b_{n-k}(i))$ gives

$$Y_s{}^n = \varphi_n(X^k) \oplus (0^k || b_{n-k}(i)) \oplus b_n(j) \tag{20}$$

Since $b_{n-k}(i)$ is known through the helper data, the decoder can correct any error as long as Eqn. 18 holds. The systematic encoding enables to generate the subcode without changing the first k bits. Therefore, we can obtain the corrected key $\hat{Y}_s^k$ by

$$\hat{Y}_s^k = \gamma_k \left(\varphi_n(X^k) \oplus b_n(j) \right) \tag{21}$$

Note that Eqn. 17 is equivalent to Eqn. 21. Eqn. 21 leads back to the default decoding problem in the standard communications case where the decoder γ_k can be used.

Example

In this toy example, we use our new scheme SLLC together with a $(n = 7, k = 4, d = 3)$ Hamming code with systematic encoding to demonstrate underlying mechanism[1]. 3 bits of helper data are stored. The code has the following generator matrix $\mathbf{G}$

$$\mathbf{G} = \left(\begin{array}{cccc||ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} \right) , \tag{22}$$

and parity check matrix $\mathbf{H}$ [20]

$$\mathbf{H} = \left(\begin{array}{ccc||ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array} \right) \tag{23}$$

Let the PUF outputs return random sequence $x_1^7 = 1001010$. The encoder of the Hamming code encodes x_1^4 to codeword c_1^7

$$c_1^7 = x_1^4 \cdot \mathbf{G} = 1001100 \tag{24}$$

Storing the 3 least significant bits directly would leak information about the key and thus violate Condition 2b in Section 4. In the new scheme, we mask the redundancy part with fresh PUF bits to bring the leakage close to zero or eliminate it completely. The XOR between PUF response and codeword gives the code index as follows (cf. 15)

$$\begin{aligned} w_1^3 &= b_3(i) && (25)\\ &= x_5^7 \oplus c_5^7 && (26)\\ &= 010 \oplus 100 && (27)\\ &= 110 && (28) \end{aligned}$$

Therefore, index 6 is stored as helper data value. The Hamming code can correct one error, which will be labeled in bold notation in the following. Assuming $y_1^7 = 10\mathbf{1}1010$ as PUF response in the field, the syndrome decoder reconstructs

$$\begin{aligned} \tilde{c}_1^7 &= y_1^4 || (y_5^7 \oplus w_1^3) && (29)\\ &= 10\mathbf{1}1 || (010 \oplus 110) && (30)\\ &= 10\mathbf{1}1100 && (31) \end{aligned}$$

The Hamming decoder corrects $\tilde{c}_1^7$ to $\hat{c}_1^7 = 10\mathbf{0}1100$ which gives us $\hat{y}_1^7 = 10\mathbf{0}1010$. This example shows how we can combine SLLC and an error-correcting code to correct errors such that

$$\hat{y}_1^4 = x_1^4, \text{ or in general } k = l \tag{32}$$

Although it is only a toy example, we have shown that SLLC permits error-tolerant secure key generation by using error-correcting codes with systematic encoding.

7. EVALUATION

This section addresses the theoretical properties of SLLC, first to show that it has optimal asymptotic behavior for large block sizes such that the capacities can be achieved. Afterwards, the secrecy leakage of SLLC is compared to two popular secure sketch constructions, namely the code-offset and the syndrome construction. For the convenience of the reader, the reference constructions are briefly discussed in Appendix A.

The overview at the end of this section shows that SLLC is currently the only deterministic scheme that achieves the secret key and the helper data capacity, and also inherently ensures information theoretic security.

In general, good PUFs have a sufficiently high entropy but do not necessarily show perfectly i.i.d. behavior. For the security proof, we therefore loosen the i.i.d. assumption to a wider assumption $H(X^n) = n - \epsilon_A$ and $H(X^k) = k - \epsilon_B$ that can also represent correlations. Further, let $H(W) = n - k - \epsilon_W$ with $\epsilon_A > \epsilon_B + \epsilon_W$.

Achievable rate of SLLC

Lemma 1:

Rate k/n is an achievable rate for SLLC and an (n, k, ϵ_1) code with systematic encoding where Conditions 2a to 2d are bounded by a finite ϵ.

Proof:

This section proves that Conditions 2a to 2d in Section 4 are fulfilled such that rate k/n is an achievable key rate for a compound source with channel $\mathcal{T}_s$ using SLLC and an (n, k, ϵ_1) code $\mathcal{C}$. We will bound ϵ_i for each condition, maximizing finally over all four ϵ_i by one ϵ.

Condition 2a.

ϵ_1 follows immediately from the block error rate of the reproduced key p_B that depends on the code $\mathcal{C}$ and the channel $\mathcal{T}_s$.

In practice, it can be computed by bounding techniques [20] or simulation.

$$\Pr[k \neq l] = p_B < \epsilon_1 \tag{33}$$

[1]We recommend using more complex and powerful codes for a good error correction capability in practice.

Condition 2b.

$$I(K;W) = H(W) + H(K) - H(WK) \tag{34}$$
$$= H(\varphi_n(X_1^k)_{k+1}^n \oplus X_{k+1}^n) + H(X_1^k) - H(X_1^n) \tag{35}$$
$$= (n - k - \epsilon_W) + (k - \epsilon_B) - (n - \epsilon_A) \tag{36}$$
$$\tag{37}$$

Therefore,

$$I(K;W) < \epsilon_2 \tag{38}$$

with $\epsilon_2 > \epsilon_A - \epsilon_B - \epsilon_W$.

Condition 2c.

$$\frac{1}{n} H(K) > R_{key} - \epsilon_3 \tag{39}$$

with Rate $R_{key} = k/n$

$$\frac{1}{n} H(X_1^k) > \frac{k}{n} - \epsilon_3 \tag{40}$$
$$\frac{1}{n}(k - H(X_1^k)) < \epsilon_3 \tag{41}$$

Condition 2d.

$$\frac{1}{n} \log_2 |\mathcal{K}| < \frac{1}{n} H(K) + \epsilon_4 \tag{42}$$
$$\frac{k}{n} < \frac{1}{n} H(X_1^k) + \epsilon_4 \tag{43}$$
$$\frac{1}{n}(k - H(X_1^k)) < \epsilon_4 \tag{44}$$

Therefore, $R_{key} = k/n$ is an achievable key rate for $n_\epsilon = n$ and

$$\epsilon = \max(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4) \tag{45}$$

■

Corollary:

For an ideal PUF with $\Pr[x = 0] = \Pr[x = 1] = 0.5$ and i.i.d outputs, Rate R_{key} is achievable with $\epsilon = \epsilon_1$.

Proof:

Due to i.i.d. PUF outputs, $H(X_{k+1}^n) = n - k$. Further, $H(X_1^k) = k$. Therefore, $\epsilon_2 \to 0, \epsilon_3 \to 0, \epsilon_4 \to 0$ which gives $\epsilon = \epsilon_1$. So, secret key rate k/n is achievable with an (n, k, ϵ_1) code.

■

Lemma 2:

The secret key rate R_{key} and the helper data rate R_{hd} of SLLC achieve the capacities of the source.

Proof:

We have shown that SLLC can achieve a rate $R_{key} = \frac{k}{n}$. In this proof we denote the code size with k_n to highlight that k depends on n for a given ϵ. A capacity achieving (n, k_n, ϵ) code $\mathcal{C}$ for a channel $\mathcal{T}$ has rate R_{key} such that

$$\lim_{n\to\infty} R_{key} = \lim_{n\to\infty} \frac{k_n}{n} = C_{\mathcal{T}} = C_{key} \tag{46}$$

By definition, the capacity achieving code $\mathcal{C}$ has redundancy $n - k_n$ with

$$\lim_{n\to\infty} R_{hd} = \lim_{n\to\infty} \frac{n - k_n}{n} = C_{hd} \tag{47}$$

■

Therefore, we have shown that our systematic coding scheme SLLC fulfills typical information theoretic requirements because the error probability and the security parameters are bounded by ϵ. In addition, our scheme is optimal in a sense that it is capacity achieving such that a maximum key size can be extracted and the required helper data size is brought down to the theoretical limit in an asymptotic setting.

Secrecy leakage in comparison to state-of-the-art secure sketches

The leakage through the helper data is a central security measure since the helper data is accessible to the attacker. Secure sketches using the code-offset construction, the syndrome construction and SLLC all return the corrected PUF response or parts of it as preliminary key. We compare the achievable key rates as well as the key leakage of the three approaches in this section.

The security analysis does not make any assumptions on the PUF. However, it is difficult to compute $H(W)$ in the generic case. To show the ideal case, we add the analysis with a nearly perfect PUF.

Systematic low leakage coding:

For SLLC, the secrecy leakage is given in Equation 38. For nearly optimal i.i.d. PUFs with entropy $H(X^n) = n - \epsilon_A$ and $\epsilon_A \ll 1$, the leakage goes to zero.

Code-offset construction:

The code-offset construction is briefly sketched in A.1. Ignatenko and Willems discussed the leakage in detail in [23] from an information theoretic point of view. A cryptographic upper bound for the leakage is presented in [31] and applied in [32].

In this scenario, the attacker knows that the PUF response X^n was XORed with a random codeword C^n. Therefore, the helper data leaks $H(C^n)$ bits of the restored secret.

$$I(X^n; W^n) = H(W^n) - H(C^n) \tag{48}$$
$$= H(X^n \oplus C^n) - H(C^n) \tag{49}$$

The code has 2^k codewords. Therefore, $H(C^n) \leq k$. For a nearly optimal PUF with entropy $H(X^n) = n - \epsilon$ and $H(C^n) = k$

$$I(X^n; W^n) = H(X^n) - H(C^n) \tag{50}$$
$$= n - k - \epsilon \tag{51}$$

Key generation from internal PUF Response	$\max R_{key}$	$\min R_{HD}$	$I(X;W)$	$I(X;W)$ (perfect PUF)
SLLC	C_{key}	C_{hd}	$H(W^{n-k}) - H(X^n) + H(X^k)$	$< \epsilon$
Random Coding [26]	C_{key}	C_{hd}	0 [26]	0
Code-Offset [19]	C_{key}	1	$H(W^n) - H(C^n)$ [23]	$< n - k + \epsilon$
Syndrome Construction [19]	C_{key}	C_{hd}	$H(W^{n-k})$ [13]	$n - k$

Table 2: Theoretical comparison to related work

Key generation from internal PUF Response	Helper data size	Leakage	Hash function required	Estimated area
SLLC	36 bit	0 bit	no	$\approx 4500\ GE$
Code-Offset [19]	155 bit	36 bit	yes	$\approx 6600\ GE$
Syndrome Construction [19]	36 bit	36 bit	yes	$\approx 6600\ GE$

Table 3: Practical comparison to related work for non-optimized implementations

As a consequence, even for a PUF with nearly optimal properties $n - k + \epsilon$ PUF bits leak, which is significantly higher as for SLLC.

Syndrome construction:

For the syndrome construction presented in [19] and discussed in A.2, two factors affect the leakage through the syndrome $W^{n-k} = X^n \cdot \mathbf{H}^T$. The overall leakage is bounded by the entropy of the PUF $H(X^n)$.

$$H(W^{n-k}) = H(X^n \cdot \mathbf{H}^T) \leq H(X^n) \tag{52}$$

Further, $|W^{n-k}| = n - k$ such that at most $n - k$ bits can be exposed by the helper data. Therefore,

$$I(X^n; W^{n-k}) = \min\{H(X^n), n - k\} \tag{53}$$

Again, for a nearly optimal PUF with $H(X^n) = n - \epsilon$, $n - k$ bits are exposed, whereas SLLC only leaks ϵ.

Remarks:

To design a system with a code-offset or syndrome secure sketch, a hash function is required to provide a cryptographically secure key. For hardware implementations such as [11, 13], lightweight hashes such as Toeplitz hashing [33] or SPONGENT [34] were used.

Maes *et al.* proposed in [13] to use strong extractors [35] to achieve information theoretic security. However, to the best of our knowledge the proposal using a strong extractor was not implemented yet.

In contrast, our new approach SLLC provides a leakage going to zero, according to Eqn. 38, for sufficiently good PUFs as soon as

$$H(X^n) - H(X^k) > n - k - \epsilon \tag{54}$$

holds with a small ϵ. In the random coding approach in [26], a constant leakage of zero can be achieved since the codewords have the same distribution as the PUF response even for low entropy PUFs.

For future research, the secrecy leakage ϵ in SLLC can be reduced with careful code design where $C_{k+1}^n(X_1^k)$ is chosen such that it can be fully masked by X_{k+1}^n. This approach enables nearly zero leakage also for PUFs with lower entropy.

Therefore, we are able to provide a stronger security with a simpler architecture that does not require a cryptographic hash function or even a strong extractor. In the case that $H(X_1^k) < k - \epsilon$, a simple data compression algorithm without cryptographic properties can be used to create a fuzzy extractor since no leakage has to be mitigated.

Theoretical comparison to related work

The performance evaluation in Table 2 shows the achievable rates and the leakage for different secure sketch constructions. All discussed approaches can achieve the key capacity C_{key} and also all except the code-offset construction achieve the minimum helper data size C_{hd}. However, SLLC and the random coding approach are the only ones that inherently provide information theoretic security. This makes, to the best of our knowledge, SLLC to the first capacity achieving and inherently information theoretically secure construction that can be implemented in practice.

8. IMPLEMENTATION

One of the main use cases of PUFs is providing a low or medium security level for standard CMOS circuits that do not have access to sophisticated security technologies that are used in chip cards and other high-security products. In the context of commercial products that are not intended for governmental or otherwise standardized use, reducing the key entropy from 128 bit to 126 or even less than 120 does not affect the security level significantly. If the implementation complexity is reduced therefore, this is a trade-off designers (or their managers) might be willing to take.

From a practical point of view, 100 bits still offer a decent security level, as stated by government agencies e.g. [36, 37]. Entropy estimations of PUF responses such as [30] show that several PUFs already show this desired security level without further privacy amplification.

With the trend towards more reliable and secure PUFs at the circuit level, e.g. [38], PUFs with bit error probabilities during provisioning of 10^{-5} and lower can be manufactured. Therefore, less powerful error correction is necessary to generate reliable keys. The structure has only negligible bias

and correlation so that it offers close to optimum security so that the last column in Table 2 holds in a good approximation.

Calculating the bit error probability according to e.g. [20] shows that in this case, a compact high-rate $(55, 43, 5)$ code already leads to a key regeneration failure rate of $7.87 \cdot 10^{-11}$. This is below the Failure in Time (FIT) specification of most, if not all, popular silicon processes (typical FIT failure rate ranges from $1 \cdot 10^{-9}$ to $2 \cdot 10^{-8}$ [39]).

In our example, we chose a BCH $(55, 43, 5)$ code in a systematic construction, which is a shortened version of a BCH $(63, 51, 5)$ code [20]. Running this three times results in $55 \cdot 3 = 165$ PUF bits consumed to derive $43 \cdot 3 = 129$ data bits for the key, and using $(55-43) \cdot 3 = 36$ helper data bits.

The proof-of-concept BCH decoding core is a modified version of the one used in [40] and requires 4441 NAND2 equivalent gates in an ASIC implementation, synthesized using Synopsys Design Compiler, comprising of 194 flip-flops and the rest conventional standard-cell combinatorial logic. It uses a serialized input and output interface. The latency is 372 clock cycles per block, and three blocks (1116 cycles) are required to generate a 128-bit key. The decoder operates in $GF(2^6)$ with field elements constructed using the primitive polynomial $p(x) = 1 + X + X^6$. The generator polynomial used to generate the codewords for the (55,43,5) BCH code is $g(x) = \big((1 + X + X^6) \cdot (1 + X + X^2 + X^4 + X^6)\big)$. Since the code is shortened, the first 6 data bits of each 63 bit block are regarded as fixed to 0.

Note that the SLLC decoder only has a negligible impact on the overall implementation size. It can be implemented as a 6 bit counter and a comparator that decides if an input bit is within the information part of the codeword and fed directly into the BCH decoder, or if it is XORed with a helper data bit.

In addition to the BCH decoder, the code-offset and syndrome methods require a hash function. Compact implementations of popular lightweight hash functions like SPONGENT (256/256/16) [34] or PHOTON (256/32/32) [41] require around 2300 GE and 2150 GE, respectively.

Table 3 compares the helper data size, the secret key leakage and an estimated gate count for SLLC, the code-offset and the syndrome method. It can be seen that SLLC is the only approach that combines minimal helper data size and zero leakage through the helper data. For both other approaches $n-k = 36$ bit of the PUF response are revealed.

Due to SLLC, we can use 128 of the 129 data bits directly as keying bits, without processing them through a hash function. The overhead for helper data is only 36 bits.

The area is estimated by adding the size of the BCH decoder and the hash function, if necessary. In a modular implementation where the BCH module and the hash module are distinct, SLLC method requires only the BCH module of an estimated 4400 GE. With code-offset or syndrome construction, an additional an estimated 2150 GE is required for the hash, for a total of an estimated 6500 GE. This is an extra 47% overhead that SLLC avoids. This result is shown in Table 3.

There is the option to decrease the size of the implementation by designing a more advanced joint BCH decoder and SPONGENT module to share resources, e.g. registers. The savings are limited by the fact that the state registers of the hash function cannot be shared with the BCH decoder after the first codeword is hashed into the state. However, there are still possibilities to share other registers or logic that do not always contain valid data.

With an overhead of less than $4,500$ gates, 36 extra helper data bits, and the use of SLLC which eliminates the use of a key hashing function, a 47% overhead in GE is avoided compared to the code-offset and syndrome method. In addition, the required helper data is cut to 24% of the code-offset method. We also achieve information theoretic security, and do not have to make additional assumptions on the security of the hash function. From a practical standpoint, there is an additional module less to secure, e.g. against the side-channel attack in [42].

We drive the reliability of high reliability PUF circuits such as [38] into a realm where the PUF key generation reliability is below the failure rate of the underlying silicon device, and also help to account for aging and other environmental effects not yet characterized in [38] and related publications.

9. CONCLUSIONS

Secure key generation with PUFs lies at the intersection of cryptology and information theoretic security on the theoretical side and secure hardware implementations as application. This work transfers recent results from information theory to design a new scheme that is suitable for implementation. As first step, we show the correspondence between the practical problem and a recent theoretical model of using a compound source.

Moreover, this work presents Systematic Low Leakage Coding (SLLC), the first capacity achieving syndrome coding scheme for PUFs that achieves maximum key size and minimum helper data size under the constraint that the leakage of the key through the helper data can be brought close to zero for sufficiently good PUFs. We discuss information theoretic preliminaries and the immediate theoretical predecessor to motivate our work and to set in a larger context.

Our approach uses error-correcting codes with systematic encoding to split the codeword in an information part and a redundancy part. Storing masked helper data only for the redundancy part is the lowest possible amount and reduces the leakage close to zero. A theoretical comparison to related work shows that our scheme is currently the only capacity achieving scheme with information theoretic security while other secure sketch constructions only achieve cryptographic security and require an additional security assumption on the hash function.

The case study demonstrated the advantages of SLLC in a practical setting, namely minimal helper data and that no hash function is mandatory to generate secure keys.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. This work was partly funded by the German Federal Ministry of Education and Research (BMBF) in the project SIBASE through grant number 01IS13020A and the Bavaria California Technology Center (BaCaTeC) through grant number 2014-1/9.

APPENDIX

A. REVIEW OF STATE-OF-THE-ART SECURE SKETCHES

The code-offset and syndrome construction are two popular secure sketches introduced by Dodis *et al.* in [19].

A.1 Code-Offset

The code-offset construction is based on the fuzzy commitment scheme by Juels and Wattenberg [43]. For helper data generation, a random number Z^k is selected. The encoder φ_n of the code creates a random codeword $C^n = \varphi_n(Z^k)$ that is XORed with X^n.

$$W^n = X^n \oplus \varphi_n(Z^k) \tag{55}$$
$$= X^n \oplus C^n \tag{56}$$

During reproduction, the decoder γ_k computes the corrected PUF response $\hat{Y}^n$ from Y^n and W^n. First, $\hat{Z}^k$ is computed by

$$\hat{Z}^k = \gamma_k \left(Y^n \oplus W^n \right) \tag{57}$$

Then, $\hat{Y}^n$ is given by the corrected codeword $\hat{C}^n = \varphi_n(\hat{Z}^k)$ and W^n such that,

$$\hat{Y}^n = \hat{C}^n \oplus W^n \tag{58}$$

A.2 Syndrome Construction

The presented method uses the parity-check matrix $\mathbf{H}$ of the code to generate the helper data. Using the steps described [13], the helper data is generated according to

$$W^{n-k} = X^n \cdot \mathbf{H}^T \tag{59}$$

During reproduction Y^n is interpreted as $Y^n = X^n + E^n$ with an error vector E^n. The syndrome S^{n-k} is computed by

$$S^{n-k} = Y^n \cdot \mathbf{H}^T \oplus W^{n-k} \tag{60}$$
$$= (X^n \cdot \mathbf{H}^T) \oplus (E^n \cdot \mathbf{H}^T) \oplus W^{n-k} \tag{61}$$
$$= E^n \cdot \mathbf{H}^T \tag{62}$$

For an (n, k, d) code and $HW(E^n) \leq \left\lfloor \frac{d-1}{2} \right\rfloor$, Y^n can be decoded to $\hat{Y}^n$.

For algebraic codes, there exists an analogous definition for a syndrome polynomial such that the discussed method can also be applied BCH codes.

Both approaches were already implemented in hardware, e.g. [11] and [13]. A comprehensive theoretical analysis can be found in [23, 24].

B. REFERENCES

[1] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[2] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *ACM/IEEE Design Automation Conference (DAC)*, 2007, pp. 9–14.

[3] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Springer Berlin / Heidelberg, 2007, pp. 63–80.

[4] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 670–673.

[5] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann, "A formal foundation for the security features of physical functions," in *IEEE Symposium on Security and Privacy (S&P)*, 2011, pp. 397–412.

[6] A. Maiti and P. Schaumont, "Improved ring oscillator PUF: An FPGA-friendly secure primitive," *Journal of Cryptology*, vol. 24, no. 2, pp. 375–397, 2011.

[7] R. Maes, "Physically unclonable functions: Constructions, properties and applications," Dissertation, Katholieke Universiteit Leuven, 2012.

[8] H. Handschuh and E. Trichina, "Securing flash technology," in *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. IEEE, 2007, pp. 3–20.

[9] A. Poschmann, "Lightweight cryptography - cryptographic engineering for a pervasive world," Dissertation, Ruhr-University Bochum, 2009.

[10] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGAs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, E. Oswald and P. Rohatgi, Eds., vol. 5154. Springer Berlin / Heidelberg, 2008, pp. 181–197.

[11] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, C. Clavier and K. Gaj, Eds. Springer Berlin / Heidelberg, 2009, pp. 332–347.

[12] M. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[13] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, E. Prouff and P. Schaumont, Eds., vol. 7428. Springer Berlin / Heidelberg, 2012, pp. 302–319.

[14] M. Hiller, D. Merli, F. Stumpf, and G. Sigl, "Complementary IBS: Application specific error correction for PUFs," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2012, pp. 1–6.

[15] 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 *International Workshop on Trustworthy Embedded Devices (TrustED)*. ACM, 2013, pp. 43–54.

[16] M. Hiller and G. Sigl, "Increasing the efficiency of syndrome coding for PUFs with helper data compression," in *Design, Automation & Test in Europe (DATE)*. ACM/IEEE, 2014.

[17] X. Kan, M. T. Rahman, D. Forte, H. Yu, S. Mei, and M. Tehranipoor, "Bit selection algorithm suitable for high-volume production of SRAM-PUF," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 101–106.

[18] S. Müelich, S. Puchinger, M. Bossert, M. Hiller, and G. Sigl, "Error correction for physical unclonable functions using generalized concatenated codes," in *International Workshop on Algebraic and Combinatorial Coding Theory (ACCT)*, 2014.

[19] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology (EUROCRYPT)*, ser. LNCS, C. Cachin and J. L. Camenisch, Eds., vol. 3027. Springer Berlin / Heidelberg, 2004, pp. 523–540.

[20] M. Bossert, *Channel Coding for Telecommunications*. New York: John Wiley & Sons, 1999.

[21] R. Ahlswede and I. Csiszar, "Common randomness in information theory and cryptography - Part I: Secret sharing," *IEEE Transactions on Information Theory*, vol. 39, no. 4, pp. 1121–1132, 1993.

[22] U. Maurer, "Secret key agreement by public discussion from common information," *IEEE Transactions on Information Theory*, vol. 39, pp. 733–742, 1993.

[23] T. Ignatenko and F. M. J. Willems, "Information leakage in fuzzy commitment schemes," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 337–348, 2010.

[24] ——, "Biometric security from an information-theoretical perspective," *Foundations and Trends in Communications and Information Theory*, vol. 7, no. 2-3, pp. 135–316, 2012.

[25] A. Khisti, S. N. Diggavi, and G. W. Wornell, "Secret-key generation using correlated sources and channels," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 652–670, 2012.

[26] H. Boche and R. F. Wyrembelski, "Secret key generation using compound sources - optimal key-rates and communication costs," in *International ITG Conference on Systems, Communications and Coding (SCC)*. IEEE, 2013.

[27] P. Tuyls and J. Goseling, "Capacity and examples of template-protecting biometric authentication systems," in *Biometric Authentication International Workshop (BioAW)*, ser. LNCS, D. Maltoni and A. Jain, Eds., vol. 3087. Springer Berlin / Heidelberg, 2004, pp. 158–170.

[28] Y. Dodis, B. Kanukurthi, J. Katz, L. Reyzin, and A. Smith, "Robust fuzzy extractors and authenticated key agreement from close secrets," *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 6207–6222, 2012.

[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley, 2006.

[30] S. Katzenbeisser, U. Kocabas, V. Rozic, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (PUFs) cast in silicon," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, E. Prouff and P. Schaumont, Eds., vol. 7428. Springer Berlin / Heidelberg, 2012, pp. 283–301.

[31] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.

[32] P. Koeberl, L. Jiangtao, A. Rajan, and W. Wei, "Entropy loss in PUF-based key generation schemes: The repetition code pitfall," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, 2014, pp. 44–49.

[33] H. Krawczyk, "LFSR-based hashing and authentication," in *Advances in Cryptology (CRYPTO)*, ser. LNCS, Y. Desmedt, Ed., vol. 839. Springer Berlin / Heidelberg, 1994, pp. 129–139.

[34] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "SPONGENT: The design space of lightweight cryptographic hashing," *IEEE Transactions on Computers*, vol. 62, no. 10, pp. 2041–2053, 2013.

[35] N. Nisan and D. Zuckerman, "Randomness is linear in space," *Journal of Computer and System Sciences*, vol. 52, no. 1, pp. 43–52, 1996.

[36] European Union Agency for Network and Information Security, "Algorithms, key sizes and parameters report - 2013 recommendations," Tech. Rep., 2013.

[37] Bundesamt für Sicherheit in der Informationstechnik, "Kryptographische Verfahren: Empfehlungen und Schlüssellängen (BSI TR-02102-1)," Tech. Rep., 2014.

[38] C. Böhm and M. Hofer, *Physical Unclonable Functions in Theory and Practice*. Springer, 2013.

[39] Xilinx, Inc., "Device reliability report UG116 (v10.1)," Tech. Rep., 2014.

[40] M. Yu, R. Sowell, A. Singh, D. M'Raihi, and S. Devadas, "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.

[41] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON family of lightweight hash functions," in *Advances in Cryptology (CRYPTO)*, ser. LNCS, P. Rogaway, Ed., vol. 6841. Springer Berlin / Heidelberg, 2011, pp. 222–239.

[42] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-channel analysis of PUFs and fuzzy extractors," in *International Conference on Trust and Trustworthy Computing (TRUST)*, ser. LNCS, J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres, Eds., vol. 6740. Springer Berlin / Heidelberg, 2011, pp. 33–47.

[43] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *ACM Conference on Computer and Communications Security (CCS)*. ACM, 1999, pp. 28–36.