Cherry-Picking Reliable PUF Bits with Differential Sequence Coding

Matthias Hiller, Meng-Day (Mandel) Yu, *Member, IEEE,* and Georg Sigl

***Abstract*—Silicon Physical Unclonable Functions (PUFs) produce a sequence of response bits from chip-unique manufacturing variations. Since the response bits are physically derived, there is noise present. To generate bit-exact cryptographic keys, error correction algorithms are used. The error correction is typically split into small processing blocks to reduce implementation complexity. The reliability of PUF responses varies from bit to bit but there has been very little work so far that mathematically analyzes the effect of the block size on the reliability of PUF response sequences.**

We use the information theoretical concept of typicality to show that the probability of drawing an unreliable sequence decreases exponentially with the block size. We present Differential Sequence Coding (DSC) that scales efficiently across larger block sizes without having the super-linear increase in decoding complexity of prior approaches. It scans the entire PUF response sequentially and then only operates on one single, maximally reliable, block to generate the cryptographic key.

Our sample FPGA implementation with a convolutional code is designed for a popular SRAM PUF scenario. It generates a 128 bit key for an average input bit error probability of 15% with an output bit error probability of $6.14 \cdot 10^{-9}$ and only uses 974 PUF bits and $1,108$ helper data bits. There are 36% less PUF bits and 71% less helper data bits than the best previous individual results in both criteria without increasing the implementation size of the key generation module noticeably.

***Index Terms*—Physical Unclonable Function (PUF), Syndrome Coding, Error Correction, Differential Sequence Coding (DSC), Typicality, Convolutional Code, FPGA.**

I. Introduction

Silicon Physical Unclonable Functions (PUFs) provide security for standard CMOS circuits by measuring the sub-micron variations that are inevitable in the manufacturing process [3]. Typically, circuits are designed to show a predictable behavior over all manufactured devices. PUFs aim for the opposite: the circuit should be sensitive to this variation to produce a unique behavior for each chip. Examples of popular PUF types include the Arbiter PUF [4], the Ring-Oscillator PUF [5], and the SRAM PUF [6]. A direct comparison of PUF ASICs manufactured in the same technology can be found in [7].

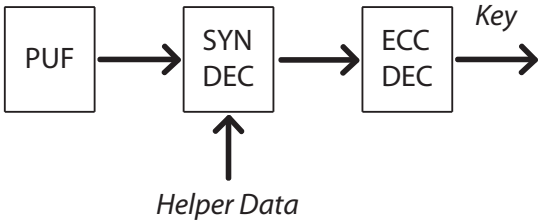


Fig. 1: Concatenation of a syndrome and ECC decoder for secret key reproduction with PUFs [1]

There are two main applications for PUFs: PUFs with a challenge-response interface can be used for lightweight authentication applications to provide a practical security improvement under strict cost constraints. As second application, which is considered in this work, cryptographic keys are derived from PUF responses as root of trust for cryptographic algorithms and for an enhanced security in standard digital circuits.

State-of-the-art PUFs generate unique and unpredictable responses which make them suitable for security applications. However, PUF responses change over temperature, voltage and time, and are also affected by random noise. Therefore, error correction is necessary to mitigate these effects and generate reliable keys.

In order to apply error correction to PUF response sequences, it is necessary to map a sequence into a form that can be corrected by a decoder of an Error-Correcting Code (ECC), for example by mapping it into a codeword of the ECC. The syndrome encoder performs this mapping, and outputs helper data that is later used by the syndrome decoder to remap the regenerated PUF response sequence into a format where an ECC decoder can detect and remove errors.

A good error correction has a small implementation size of the decoding logic, requires only a low number of PUF bits and the size of the stored helper data is small.

Figure 1 shows a system with the basic components to reproduce a key from a PUF. The PUF response and the helper data are combined in the syndrome decoder and then the errors are removed in the decoder of the ECC that outputs the cryptographic key.

Manuscript received 20.11.2015; revised 13.04.2016; accepted 16.05.2016.

Matthias Hiller and Georg Sigl are with the Chair of Security in Information Technology, Technical University of Munich, Munich, Germany. Email: {matthias.hiller, sigl}@tum.de

Meng-Day (Mandel) Yu is with Verayo, Inc., San Jose, CA, USA; COSIC, Katholieke Universiteit Leuven, Leuven, Belgium; and CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA. Email: myu@verayo.com

Preliminary versions of this work were presented at the International Workshop on Trustworthy Embedded Devices (TrustED) 2013 [1] and the Design, Automation & Test in Europe Conference & Exhibition (DATE) 2014 [2]. The previous work is extended by the theoretical foundations using typicality, new simulation results, and an improved implementation.

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.

The reliability of individual PUF response bits varies significantly over the PUF response sequence [8], [9], [10]. The required number of measurements depends on the desired precision of the reliability estimation, see e.g. [11]. In practice, ten to one hundred measurements often already provide a sufficient quality [12]. Assessing the reliability over different operating conditions improves the reliability of the estimation [13]. In [14], the implications of taking multiple measurements are analyzed from an information theoretical point of view. If the PUF provides fine-grained outputs, such as counter differences for the RO PUF, more precise reliability information can be derived [10].

In the following, we will apply the reliability distribution in [8], which was obtained from real world data, for discussion and simulation. This makes our results reproducible and also comparable to referenced work referring to the same setup.

Aiming for a low implementation size of the error correction, it is crucial to minimize the computational complexity. Complexity can be reduced for example by selecting the most reliable PUF bits and ignoring unreliable ones [15], [16], [17].

Typically, the decoding effort increases more than linearly with the block size, e.g. quadratic in certain implementations [16]. State-of-the-art implementations like [18], [12], [16], [17] decrease complexity by splitting large blocks into smaller ones, for example through code concatenation [19] or even fully independent blocks. This enables compact hardware implementations but comes with an increase in error probability compared to single-block solutions. A comparison of the schemes on algorithmic level can be found in [20].

So far, there has been very little work describing the influence of the block size on the reliability in the PUF context. We use the information theoretical concept of typicality [21], which is similar to the Law of Large Numbers, to make more precise estimations on the reliability of a PUF sequence with an increasing length. We use this concept to predict the number of reliable PUF bits within one processed block. Applying letter typical sequences [22] shows that the probability of a non-typical sequence decreases exponentially with the block size.

We introduce Differential Sequence Coding (DSC), a new syndrome coding scheme that searches the entire PUF response sequence for PUF response bits fulfilling a given reliability criterion. This makes DSC a lossy algorithm (in an information theoretical sense) that outputs a failure if not enough reliable PUF bits are available. As a consequence, we are able to give a lower bound for the reliability of each deployed device whereas previous approaches can only make statements on the average reliability over all devices. We note that the prior schemes do not scale well with larger block sizes, e.g., processing the entire key with a length of 128-bits as a single block causes a significant increase in decoding implementation complexity.

Comparing the performance of different approaches in a practical scenario supports our theoretical results so that we are able to provide the most efficient FPGA hardware implementation for the popular SRAM PUF scenario presented by Guajardo *et al.* [6] and extended by Maes *et al.* [8], [12].

Our contributions:

- This work is the first to theoretically analyze the relationship between the block size and the reliability of PUF response blocks. In particular, we analyze the effect of the block length on the distribution of reliable PUF bits within each block using the information theoretical concept of typicality.
- We introduce DSC, a new syndrome coding scheme that operates only on one single, maximally reliable, block for key generation, and are also the first to use convolution codes, which do not operate on block boundaries, for PUFs.
- New DSC simulation results are compared against the state of the art. The bit error probabilities for different PUF bit / key bit ratios is between $20\times$ and $3,800\times$ lower than the best prior results.
- The comparison of our implementation against prior results shows that with a similar implementation complexity we require 36% less PUF bits and 71% less helper data bits.

Outline:

Section II presents related work on syndrome coding. The concept of typicality is applied to syndrome coding in Section III. We discuss DSC in Section IV and analyze the performance of the full PUF key generation in Section V. Our FPGA implementation is presented in Section VI. Afterwards, Section VII concludes the paper.

II. Related Work

There exists a wide body of previous work that will later serve as reference to evaluate our new approach in a direct comparison. We distinguish between two major families of syndrome coding schemes: the first family performs linear operations between the PUF response and codewords of the ECC while the second computes pointers to specific codeword bits or entire codewords.

All of the referenced approaches have in common that they split the PUF response into blocks such that they cannot correct errors from large numbers of unreliable PUF bits efficiently in practice.

A. Linear Operations

The code-offset approach by Dodis *et al.* [23] XORs a sequence of PUF response bits and a random codeword of an ECC to create the helper data. The implementations by Bösch *et al.* [18] and van der Leest *et al.* [24] use Bose-Chaudhuri-Hocquenghem (BCH) and Golay codes [19] with relatively short block sizes. In addition, Maes *et al.* [8], [12] discussed and implemented a soft-decision approach with Reed–Muller codes [19] and soft-decision decoding as generalized multiple concatenated codes [19]. In [25], Müelich *et al.* introduced a construction based on generalized code concatenation and Reed–Muller codes that was implemented in [26]. The extension by Puchinger *et al.* [27] also contains constructions with Reed–Solomon codes [19].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2573766, IEEE Transactions on Information Forensics and Security

In the syndrome-based approach, the PUF response is multiplied with the parity check matrix of the code [23] and the syndrome is stored as helper data. A stand-alone implementation of a PUF key generator with Ring-Oscillator PUFs and a BCH code on an FPGA can be found in [28]. Recent work by Herder *et al.* [29] uses a computational security argument.

Systematic Low Leakage Coding (SLLC) [30] is a recent approach that uses codes with systematic encoding and only stores helper data for the redundancy part.

B. Pointer-Based Approaches

Index-Based Syndrome Coding (IBS) by Yu and Devadas [16] creates helper data by storing pointers to the most stable PUF outputs within each block. Hardware implementations on ASIC and FPGA were presented in [31], [32]. Complementary IBS (C-IBS) by Hiller *et al.* [17] is an extension of IBS that increases the reliability by searching the same set of PUF outputs multiple times to store several pointers.

The implementations in [16] and [17] use small block sizes of 8 and 9 and it was shown in [33] that the reliability differs significantly among such small blocks. In [31], the block size of 32 leads to less variation but only one out of 32 PUF bits is used. The approaches have the advantage that helper data and key are statistically uncorrelated under the assumption that the PUF bits are independent and identically distributed (i.i.d), which makes these approaches information theoretically secure. For an SRAM PUF, the bits are derived from physically distinct circuit components so that an i.i.d. assumption is quite reasonable for good implementations with low bias and correlation as discussed for example in [7], [9]. However, note that this does not hold in general for all SRAM PUFs, see e.g. [34], so that this prerequisite should be checked in advance for a specific circuit and technology.

The symbol recovery by Yu *et al.* [35] can also be seen as pointer-based approach in a wider sense. The maximum likelihood decoding performance is optimal and the decoder is easily implementable to embed up to 9 key bits into blocks of 64 to 256 PUF bits. The approach is very efficient for PUF responses with bit error probabilities over 20%. However, the decoding effort increases exponentially with the number of key bits per block so that the approach is not suitable to embed large numbers of key bits per block.

Our DSC approach bridges the gap by enabling a maximum block size for higher key bit / PUF bit ratios which makes DSC very efficient for average PUF response bit error probabilities of 15% and lower. A fair quantitative comparison between implementations requires identical assumptions on the PUF and desired key error probability, and a similar FPGA platform. Later, we will therefore compare our results to [18], [8], [12], [17] which are all designed for an SRAM PUF with average bit error probability of 15%, a target key error probability of 10^{-6} on Xilinx Spartan 3 FPGAs.

III. Typicality in Syndrome Coding

In this section we will show with the information theoretical concept of typicality how the efficiency of syndrome coding schemes improves with an increasing block size. Averaging effects become stronger and the impact of statistical outliers is reduced. Therefore, the longer the block, the more precise the prediction on symbols. In our case this refers to the number of reliable PUF bits in the block.

Assuming sequences of length 100 that are drawn from a binary source with uniform distribution, it is intuitive that most sequences have a roughly balanced number of zeros and ones. To be precise, for example a sequence with less than 25 or more than 75 ones is only drawn with a probability around 10^{-7}. So, we can make predictions about how many zeros and ones will be in the sequence without knowing their actual positions, and the longer the sequence is, the more precise the prediction.

A. Definitions and Notation

Random variables are marked with capital italic letters, e.g. X and scalars such as outcomes of random variables in small italic letter, e.g. x. Calligraphic letters $\mathcal{X}$ indicate sets. X^n denotes a vector of n random variables with the same output alphabet. Note that the random variables in X^n can have different probability distributions such that each PUF response bit can have an individual bit error probability. $\Pr[A]$ is the probability of event A. $P_X(\cdot)$ denotes the probability distribution corresponding to random variable X. $cdf(\cdot)$ is the corresponding cumulative distribution function. Further, let $\mu(\cdot)$ be the mean operator and $\sigma(\cdot)$ the standard deviation.

For random variables X and Y, $X|Y$ denotes X under the 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 [21].

B. Typical Sequences in Syndrome Coding

In the following, we will use the concept of typicality to analyze the effect of the block size on the distribution of reliable inputs for syndrome coding. Let X^n be a part of the overall response sequence drawn from the PUF.

Let set $\mathcal{P}$ contain the probability distributions P_{X_i} over PUF bits X_i, $i = 1, ..., n$. Each $X_i \in \{0, 1\}$ has i.i.d. Bernoulli probability distributions P_{X_i}. The distribution of the probability distributions in $\mathcal{P}$ depends on the reliability of the PUF, e.g. [8].

For a PUF response bit X_i with expectation $\mu(X_i) \geq 0.5$, we define an error as drawing a 0 which occurs with $\Pr[X_i = 0] = 1 - \mu(X_i)$, and for $\mu(X_i) < 0.5$ for drawing a 1 analogously. The first question is, what is the probability p that a given PUF output is reliable?

We define a fixed threshold p_{max} as the maximum tolerable error probability of a PUF response bit to be considered reliable. This gives

$$p = \Pr[\mu(X) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}] \qquad (1)$$

Therefore, for each of the n PUF responses X_i in X^n we get a Bernoulli distributed reliability indicator R_i with parameter p that indicates if Eqn. 1 holds for a specific PUF bit X_i or not. The sequence r^n drawn from $\{0, 1\}^n$ for a specific X^n

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2573766, IEEE Transactions on Information Forensics and Security

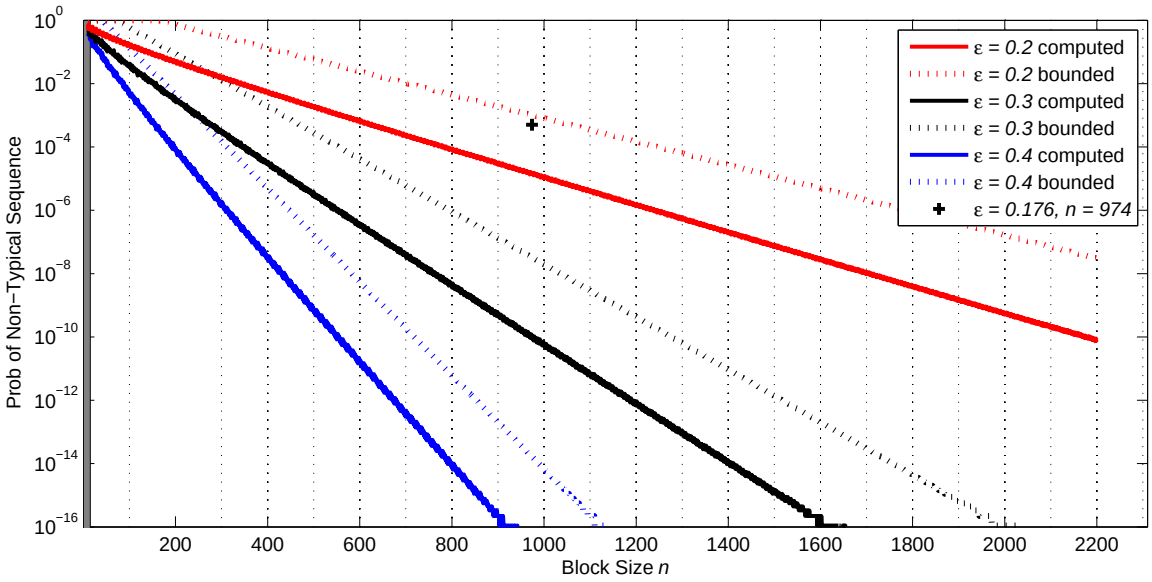


Fig. 2: Probability of drawing non-ϵ-letter-typical sequences, computed and upper bounded values for $p = 0.326$ and different ϵ parameters. The parameters $\epsilon = 0.176$ and $n = 974$ are used later in the implementation. ECCs are designed to correct errors in typical sequences; non-typical sequences can lead to increased error probabilities.

indicates for each PUF bit X_i if it is reliable $(r_i = 1)$ or not $(r_i = 0)$, $i = 1, ..., n$.

For the quantitative evaluation of typicality, we require the following notation:

- The parameter $\epsilon > 0$ quantifies the maximum allowed deviation of a sequence to still be part of the typical set.
- The letters $a \in \{0, 1\}$ indicate whether a PUF response bit is reliable or not.
- $N(a|r^n)$ is the number of occurrences of letter a in sequence r^n and quantifies the empirical distribution, i.e. the number of reliable and unreliable PUF bits in X^n.
- $P_R(\cdot)$ gives the precise theoretical distribution of letters occurring in sequences R^n.

According to [22], a reliability indicator sequence r^n is an ϵ-letter typical sequence if

$$\left|\frac{1}{n}N(a|r^n) - P_R(a)\right| \leq \epsilon \cdot P_R(a) \text{ for all } a \in \{0, 1\} \quad (2)$$

and the letter typical set $\mathcal{T}_\epsilon^n(P_R)$ is defined as set containing all sequences in $\{0, 1\}^n$ that fulfill Eqn. 2.

Applying Hoeffding's inequality in [22], the probability of an ϵ-letter typical sequence is lower bounded by

$$\Pr[R^n \in \mathcal{T}_\epsilon^n(P_R)] \geq 1 - 4 \cdot e^{-n\epsilon^2 \min[p,1-p]} \quad (3)$$

and as complementary event

$$\Pr[R^n \notin \mathcal{T}_\epsilon^n(P_R)] < 4 \cdot e^{-n\epsilon^2 \min[p,1-p]} \quad (4)$$

Note that the bound is a very tight bound where the probability of a non-typical sequence decreases exponentially with n. In contrast, applying the more widely used concept of entropy typical sequences [22] only gives a linear decrease of the probability over n.

C. Analysis

An efficient error correction is designed to correct the errors that occur in typical sequences which have a controlled number of reliable bits. We cannot make precise statements on the other sequences so that errors can occur more likely if we draw a sequence that is not element of the typical set. Reducing the probability of drawing a non-typical sequence is a first step to reduce the overall error probability.

Figure 2 plots the probability of drawing a non letter-typical sequence over the block size n for $p = 0.326$ and different parameters ϵ. This p parameter will be used later in the implementation to reduce the average bit error probability of the PUF from 15% to 2.7%. An epsilon value of 0.4 corresponds to a ratio of at least $(1 - \epsilon) \cdot p = 19.5\%$ reliable PUF bits in a typical sequence, whereas for $\epsilon = 0.2$ already 26% reliable bits are guaranteed.

The solid lines show the precise computed values while dotted lines give bounded values according to Eqn. 4. Note that the straight lines on the logarithmic scale correspond to an exponential behavior.

Figure 2 shows that the block size has a large impact on the probability of drawing non-typical sequences. Smaller blocks will lead to an increased key error probability. As a consequence, maximizing the block size is a first requirement for an efficient usage of the reliable PUF bits. In addition, it is important to find a good trade-off between the ϵ parameter and the probability of a non-typical sequence.

If ϵ is selected too strict, we can make very specific predictions on the PUF sequence but only have a reduced probability of drawing such a sequence. Otherwise, if ϵ is too large, we have a high probability of drawing a typical sequence but only know less precise information about the number of reliable bits.

The reference implementations for our scenario use block sizes between 3 and 11 [36], [8], [37]. This region is highlighted in gray in the far left of the plot. Several design space explorations have shown that this is the most favorable region for the conventional approaches. However, the other curves show that the probability of drawing non-typical sequences in this area is $> 10^{-1}$ so that errors through non-typical sequences have to be corrected on a regular basis.

The point that will be used later in our implementation is marked with the black cross. We will operate on a single block of size 974 and probability of a non-typical sequence of $5 \cdot 10^{-4}$. ECCs are designed to correct a specific number of errors and if the errors in all typical sequences can be corrected with a high probability, we can ensure successful error correction with a high probability. By using one maximum-sized block, we bring the probability of drawing a non-typical sequence down by a factor of $200\times$.

So far, the largest block sizes of up to 256 can be found in the work of Yu *et al.* [35], which is designed for higher PUF noise, resulting in significantly higher PUF bit / key bit ratios.

IV. Differential Sequence Coding

The previous section has shown that controlling the number of reliable PUF bits within each block is a prerequisite for efficient key generation with PUFs. Larger block sizes are favorable to control the number of unreliable bits per block. However, ECCs with large block sizes typically create a heavy resource overhead. Our Differential Sequence Coding (DSC) approach operates on one maximally reliable block with low overhead. In particular, we ensure beforehand that a PUF response sequence with low bit error probabilities is fed into the ECC decoder. To minimize the decoder complexity to achieve a low hardware overhead, only the reliable PUF bits from the maximally-sized single block are processed while the rest is discarded.

A. DSC Encoding

During the generation step, the PUF provides a sequence of PUF bits X together with a reliability indicator $\mu(X)$ for each PUF response bit. Note that the reliability indicators $\mu(X)$ are unique for each chip so that they have to be obtained for each device separately. DSC reads the entire PUF response sequence X^n and marks the PUF bits that have a reliability above a predefined threshold. They point to a secret sequence C^k within the PUF response sequence. The notation is adapted from block codes where n describes the block size and k the number of embedded bits.

The PUF sequence X^n is scanned sequentially for PUF bits that are more reliable than a given error probability threshold p_{max}. When such a PUF bit is found, the distance to the last reliable PUF bit is stored as differential distance pointer U. If the expected value $\mu(X)$ of the PUF bit is closer to the corresponding code sequence bit C, the inversion bit V is set to zero. Otherwise, it is set to one. A version without inversion bits is also possible where only PUF bits that are close to a code sequence bits C_i are indexed.

Figure 3 shows an example for DSC encoding. Code sequence c^4 is provided by an ECC and DSC stores one pointer for each code sequence bit. In the example, zeros are represented by white boxes and ones by black boxes. For the PUF response X^{16} and a given maximum error probability p_{max}, a white box denotes $\mu(X) < p_{max}$. A black box stands for $\mu(X) \geq 1 - p_{max}$ and gray boxes show the unreliable PUF outputs with $p_{max} \leq \mu(X) < 1 - p_{max}$.

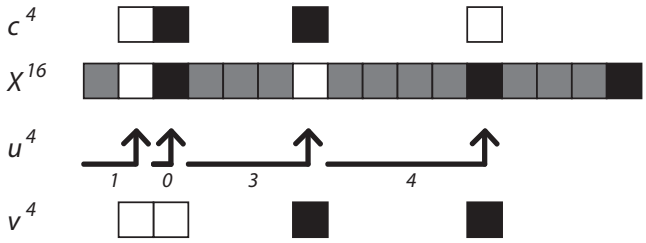


Fig. 3: Example for DSC encoding

The code sequence $c^4 = (0, 1, 1, 0)$ and the PUF response X^{16} are encoded to the helper data tuple $w^4 = (u, v)^4$. X_2 is the first reliable PUF response bit. We count the unreliable PUF response bits between two reliable ones as distance pointer, so $u_1 = 1$. For the first inversion bit, $v_1 = 0$ since both boxes have the same color. X_3 is the next reliable PUF bit, so $u_2 = 0$ and again $v_2 = 0$. After skipping three unreliable PUF bits, X_7 is indexed by $u_3 = 3$. Since a white box is indexed for a black code bit, $v_3 = 1$. u_4 and v_4 are computed accordingly, such that $u^4 = (1, 0, 3, 4)$ and $v^4 = (0, 0, 1, 1)$.

An algorithmic description of DSC encoding can be found in [1], [2]. If the length of the code sequence exceeds the number of reliable PUF bits, or if helper data cannot be stored, an error is thrown and the algorithm fails to generate a valid set of DSC helper data.

B. Security

The security of DSC was analyzed in [1] in detail. The final equation of the security analysis was:

$$I(C^k; W^k) \leq k - H(\overline{X}^k) \tag{5}$$

The secrecy leakage through the helper data, given by the mutual information between code sequence and indexed PUF response bits $\overline{X}^k$, can be brought to zero for PUF bits that are close to independent and uniformly distributed such that they have a sufficiently high entropy.

C. Yield

Previous work such as IBS and C-IBS, or the code-offset approach takes a fixed number of PUF bits to encode a fixed number of secret bits for each block. The average bit error probability for a failure in the field can be determined and is well-controlled. However, it is not possible to determine the reliability of a specific device. As a consequence, we cannot guarantee that all devices of a batch fulfill a given minimum reliability.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2573766, IEEE Transactions on Information Forensics and Security

In contrast, DSC already detects unreliable devices when no valid set of helper data is generated. This occurs when not enough stable PUF bits are found during manufacturing. This additional measure gives a-priori reliability information about individual devices. Therefore, we can assess the average error probability in the field over all devices and also provide bound that the reliability of no individual device will exceed a given maximum error probability.

If the error occurs during manufacturing, the device can still be used and programmed with a different parameter set, for example, for a lower target reliability or smaller key size. In the worst case, it has to be discarded during the manufacturing process. However, the consequences are well-controlled and do not affect devices during operation out in the field. If an error would occur during operation in the field instead, the device is not able to generate the correct key and subsequent tasks cannot be performed successfully. In our DSC approach, the probability of any individual device failing in the field is bounded. In the conventional approach, there is only a guarantee of the average failure probability.

Recall that we defined p as the probability that the error probability of a PUF response bit X is smaller than p_{max} for a PUF with a given cumulative distribution function $cdf(\cdot)$.

$$\begin{aligned} p &= \Pr[\mu(X) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}] \\ &= cdf(p_{max}) + (1 - cdf(1 - p_{max})) \end{aligned} \tag{6}$$

Note that some of the previous work such as [8], [16], [17] requires a precise estimation of $\mu(X)$ whereas our DSC approach only operates on a binary decision whether $\mu(X_i) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}$, $i \in \{1, ..., n\}$, holds, or not. The reliability information can be obtained for example by performing multiple measurements [8] or evaluating analog PUF output values [10].

As mentioned before, there are two events where helper data generation fails.

Error Event 1: Lack of reliable PUF Bits

Successful DSC encoding requires that X^n contains k reliable PUF bits with $\mu(X) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}$. If less than k are found, error event $error\_1$ is triggered. This occurs with a probability of

$$e_1 = \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} \tag{7}$$

The probability e_1 that $error\_1$ occurs is the sum over all probabilities for outcomes in which X^n contains less than k PUF reliable bits with expectation $\mu(X) \in \{[0, p_{max}] \cup [1 - p_{max}, 1]\}$. Note that according to the typicality discussion in Section III, e_1 decreases exponentially in n for a constant ratio $\frac{n}{k}$ (see Section IV-E for more intuition about this ratio).

Error Event 2: Helper Data Overflow

The second error event $error\_2$ occurs if the variably sized helper data does not fit in the allocated space. Later in Section V-D we will obtain the distribution of the helper data size of the selected parameter set through simulation.

The yield y is computed by the probability that neither $error\_1$ nor $error\_2$ occur. $error\_1$ and $error\_2$ are not disjoint, so

$$y > 1 - (e_1 + e_2) \tag{8}$$

Note that e_1 depends on the worst case reliability and the size of the PUF, whereas e_2 is only affected by the size of the helper data.

The events $error\_1$ and $error\_2$ define hard break conditions and affect the yield directly. In the following, we will aim for a yield $y > 99.9\%$, and thus set $e_1 \leq 5 \cdot 10^{-4}$ and $e_2 \leq 5 \cdot 10^{-4}$.

D. Helper Data Representation

The straight-forward approach to represent a distance pointer u_i as helper data is to allocate l bits and store the binary representation of u_i. The pointers u are geometrically distributed with parameter p, so the probability distribution P_U is

$$P_U(u) = (1-p)^u \, p \tag{9}$$

The basic representation contains a significant amount of redundancy. Run-Length Encoding (RLE) [38] is an optimal lossless compression algorithm for geometric distributions and has a reasonably low implementation complexity. RLE encodes any integer number u by a series of ones followed by a zero as delimiter and a small number of a finite alphabet $\mathcal{L}$ with elements $l_j \in \mathcal{L}$, $j = 0, ..., m-1$ and $|\mathcal{L}| = m$. For the run-length part, m determines how many unsuccessful trials are represented by every 1 and l gives the number in the remaining $u \mod m$ trials. Therefore, the compressed version $q(u)$ of u is given by

$$q(u) = \underbrace{1 \ \ldots \ 1}_{\lfloor \frac{u}{m} \rfloor \text{ times}} \ 0 \ l_{(u \bmod m)} \tag{10}$$

The input distribution $P_U(u)$ and parameter m define the distribution and value of $q(u)$. In Figure 4, p is plotted on a logarithmic x-axis and the average length $\mu(q(u))$ is shown for different m on the linear y-axis. Note that the entropy (solid cyan line) is the lower bound for every lossless compression. Therefore, we try to approach this limit as closely as possible.

Figure 4 shows that varying m gives very low overheads for different values of p such that RLE can enable optimal compression for DSC. See [2] for more detailed reasoning and results on helper data compression.

E. DSC Bit Error Probability

For a given PUF response distribution P_X, the bit error probability of DSC p_{syn} can be computed analytically for a given maximum output error probability p_{max} through the integral over the error probabilities of all PUF response bits within the specification, weighted by their probability of occurrence p given by Eqn. 6, so

$$p_{syn} = \frac{1}{p} \left(\int_0^{p_{max}} P_X(x) \cdot x \ dx + \int_{1-p_{max}}^{1} P_X(x) \cdot (1-x) \ dx \right) \tag{11}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2573766, IEEE Transactions on Information Forensics and Security

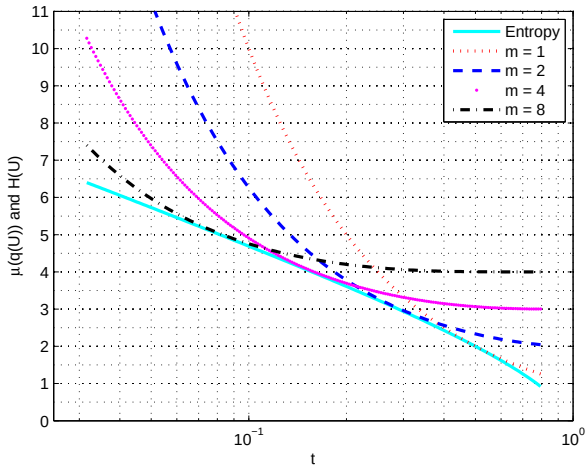


Fig. 4: Average RLE encoded pointer sizes $\mu(q(U))$ and entropy $H(U)$ for geometrically distributed random variables U with parameter p

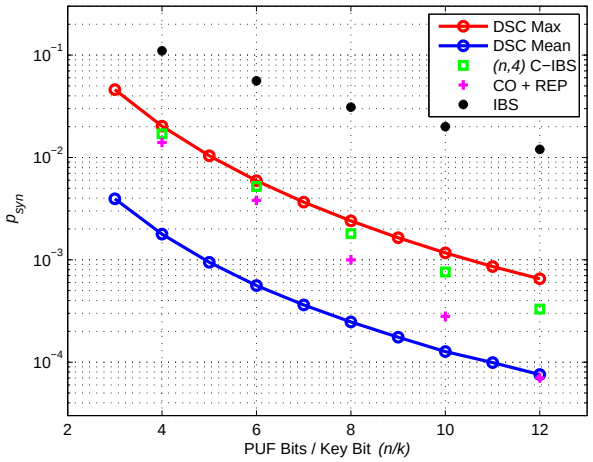


Fig. 5: Max and mean bit error probabilities of syndrome coding schemes without second stage ECCs for an SRAM PUF with 15% average bit error probability.

In the following, we will also use the SRAM PUF distribution presented in [8] to compare DSC to state-of-the-art approaches. Distributions for other PUFs can be generated for example with the help of [39], [10].

In Figure 5, the bit error probabilities p_{syn} for different syndrome coding schemes are shown over the number of SRAM PUF bits n per embedded bit k with the distribution given in [8] and mean error probability 15%. For DSC and key size[1] $s = k = 128$, n is chosen such that $e_1 = 5 \cdot 10^{-4}$ resulting in block size n for DSC. Note that DSC's maximum error probabilities $p_{syn-max} = p_{max}$ are in the same range as previous mean error probabilities for low n to k ratios.

[1]In this basic case, we do not consider an additional ECC such that $s = k$. Later, in Section V, a convolutional code is added after the DSC decoder, so $s < k$.

Comparing the mean error probabilities, DSC is considerably more efficient than previous work for a PUF bit / key bit ratio of 4. This is caused by the fact that other approaches operate on very small independent blocks with varying reliability. It takes an $\frac{n}{k}$ ratio between 9 and 10 for the code-offset method and a repetition code with SDML decoding to approach DSC performance. This shows that a careful selection of the 10% most reliable PUF bits still has a lower error probability as computing repetition code blocks of size 10. In addition, adding 10 bits and performing the decision if the sum is larger than 5 is significantly more complex than simply forwarding one single bit.

V. Design of a Complete Key Generation Module

It is obvious that the error probabilities after syndrome coding are not sufficient to consider the reproduced secret as reliable cryptographic key. Therefore, PUFs need a second stage error correction.

Convolutional codes are a popular class of ECCs and our work is the first to use convolutional codes in the PUF context. This section analyzes the performance of DSC concatenated with a convolutional code to derive a parameter set suitable for implementation.

A. Effect of the Block Size on the Typical Set

In virtually all PUF key generation schemes published to date, block-based error correction is used. This can be in the form of a BCH or repetition code, or index based coding (in which case block size corresponds to 2 to the power of the index size).

The reliability of these schemes is influenced by the larger ϵ value in ϵ-letter typical sequences associated with small block sizes n. As described in Section III, the probability of drawing too few reliable bits decreases exponentially with an increase in block size. By using a smaller block size, the prior approaches require more PUF bits to be used for each key bit, as shown in Figure 5 for the same level of p_{syn}, or alternatively higher p_{syn} for the same PUF bit / key bit ratio.

B. Convolutional Codes

Convolutional codes are a popular code class from the early days of coding theory [40] that show a good error correction performance under strict area constraints which is highly desired in the PUF context. In particular, the Viterbi algorithm [41] is an especially efficient decoding algorithm that is well-suited for hardware implementation.

C. Key Bit Error Probability

A very important performance criterion for PUF error correction is the number of input PUF bits that are required and the bit error probability of the output. Therefore, the relation between number of PUF bits n and output bit error probability p_{err} quantifies the efficiency of the decoder. Note that p_{syn} used in Section IV referred to the error probability after syndrome decoding while p_{err} quantifies the error after both syndrome decoding and ECC error correction.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2573766, IEEE Transactions on Information Forensics and Security

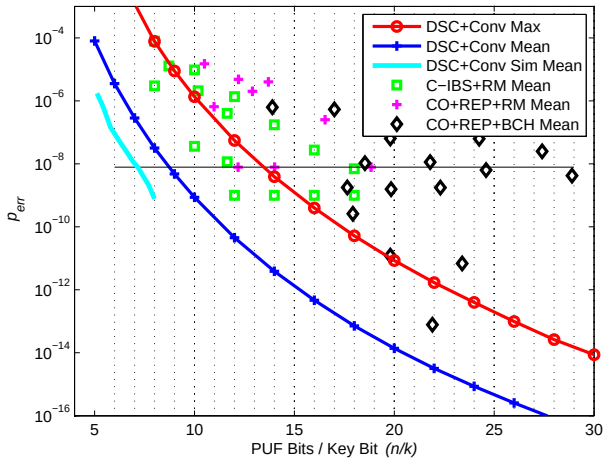


Fig. 6: Bounded mean and max key bit error probabilities of DSC concatenated with a $(2, 1, [7])$ convolutional code compared to the state of the art for an SRAM PUF with average bit error probability 15%. Again, $e_1 = 5 \cdot 10^{-4}$.

Figure 6 sets our DSC and convolutional coding approach in relation to previous work. The goal is to bring a system in the lower left corner of the diagram such that a low key error probability is achieved with a low number of PUF bits. Reference values that are all coherent with the scenario in [8] were taken from [36], [37], [17].

It can be seen, how the field moved to the left over time. The black diamonds are the code-offset fuzzy extractor results by Bösch *et al.* [18], [36]. Repetition codes were concatenated with relatively small BCH and Golay codes and decoded with hard decision decoding. The magenta crosses show the results by Maes *et al.* [8], [12] where repetition codes were concatenated with Reed-Muller codes and decoded with GMC and SDML soft decision decoders [19]. The field moved to the left compared to the older Bösch *et al.* results, so in general the approach shows an improved performance so that less PUF bits are required to achieve the same key bit error probabilities. The C-IBS results by Hiller *et al.* [17], [37] also use IBS pointers in combination with repetition codes and Reed–Muller codes and soft-decision decoding. The results overlap widely with the results in [8], [12] with a light shift to the left.

The two lines in the diagram represent DSC concatenated with a $(2, 1, [7])$ convolutional code[2]. As a first difference, the other approaches have fixed bit numbers that result in points in the diagram. For DSC, p_{max} can be chosen quasi-continuously as fine-grained as the digital representation of the reliability values of the PUF response bits allows. Applying the bounding techniques discussed in [1] gives a maximum bit error probability for each device and also quick reference values for the mean error probability.

DSC's maximum error probability is comparable to the mean error probabilities of the state of the art whereas the bounded mean error probabilities of DSC separate the field from the left corner. This demonstrates that our DSC and convolutional code concatenation is more efficient than previous work over the entire analyzed range.

The bounded results give a very quick rough estimate on the performance of a scheme. We also performed a Monte Carlo simulation campaign to measure the actual values. The cyan line, which is the leftmost line and without explicit data points, shows our simulation results that are roughly 1.5 PUF bits per key bit better than the corresponding upper bounded values.

As in [6], we aim to generate a 128 bit key with an error probability smaller as 10^{-6}. This corresponds to a target bit error probability of $e_t = 7.81 \cdot 10^{-9}$, shown by the horizontal line in Figure 6. The simulations have shown that we can reach e_t with $p_{max} = 0.027$ by indexing in average $p = 0.326$ of the available PUF bits. This specific value was measured by simulating $\nu = 1.9 \cdot 10^{11}$ PUF bits on Intel Core i7 CPUs where each CPU simulated $6.5 \cdot 10^7$ PUF bits per hour in 8 parallel threads. $1,170$ bit errors were found in total, resulting in a measured bit error probability of $e_m = 6.2 \cdot 10^{-9}$.

In addition to this mean value e_m, we also provide confidence intervals to quantify, how precise our result is. Let k_σ be a scalar to give a number as multiple of standard deviations, i.e. $k_\sigma \cdot \sigma$. The confidence interval is defined as $[e_m - \Delta e, e_m + \Delta e]$. To assess the confidence of the results, we will use Eqn. 12 [11], simplified for large ν.

$$\nu = \frac{e_m(1 - e_m) \cdot (k_\sigma)^2}{(\Delta e)^2} \tag{12}$$

$k_\sigma = 3.29$ corresponds to a 99.9% confidence interval. Solving Eqn. 12 for Δe gives $\Delta e = 6 \cdot 10^{-10}$. Therefore, we can say with a confidence of 99.9% that our setup has a bit error probability smaller than $6.8 \cdot 10^{-9}$.

Next, we solve Eqn. 12 for k_σ and set $\Delta e = e_t - e_m$. As a result, the specified maximum error probability e_t has a distance of $k_\sigma = 8.9$ standard deviations from our simulated value e_m. Therefore, the specification e_t is met with a confidence[3] of $1 - 2.5 \cdot 10^{-19}$.

The corresponding number of PUF bits is 974 to embed the required 270 code sequence bits, or 128 key bits. We expect $974 \cdot p = 317.5$ reliable PUF bits in average and require 270 reliable PUF bits to be able to index the entire code sequence[4]. The average overhead is $\frac{317.5}{270} = 1.176$. Therefore, letter typical sequences with an ϵ of 0.176 can be accepted. The solid red line with $\epsilon = 0.2$ in Figure 2 in the typicality analysis in Section III is under the practical value for $\epsilon = 0.176$, shown by the black cross. Thus, we have shown that even such a low ϵ value can be efficiently realized in practice if the block size is large enough.

As a result, the simulated DSC value in the desired left side of Figure 6 for $\frac{n}{k} = 8$ is $3,800\times$ lower than the corresponding C-IBS result. For the bounded mean values,

[2]The code parameters define that two output bits are derived for each input bit. The encoder has a memory of 7. 11 code sequence bits are affected by each information bit. See e.g. [19] for details.

[3]Let $cdf_N(\cdot)$ be the *cdf* of the Normal distribution. Then, the width of the confidence interval is given by $cdf_N(k_\sigma) - cdf_N(-k_\sigma)$

[4]The 128 key bits are encoded to $2 \cdot 128 = 256$ code sequence bits. Truncation [19] requires another $2 \cdot 7 = 14$ bits leading to 270 code sequence bits in total.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2573766, IEEE Transactions on Information Forensics and Security

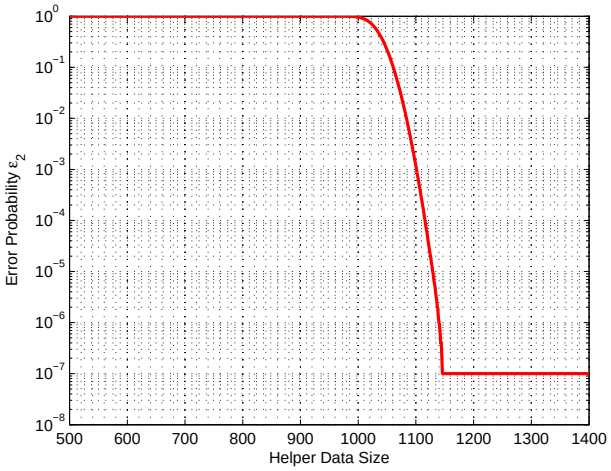


Fig. 7: Overflow error probabilities for different fixed helper data sizes and 10^7 simulated PUFs with DSC encoding with $p = 0.326$, helper data compression with $m = 2$ and a $(2, 1, [7])$ convolutional code

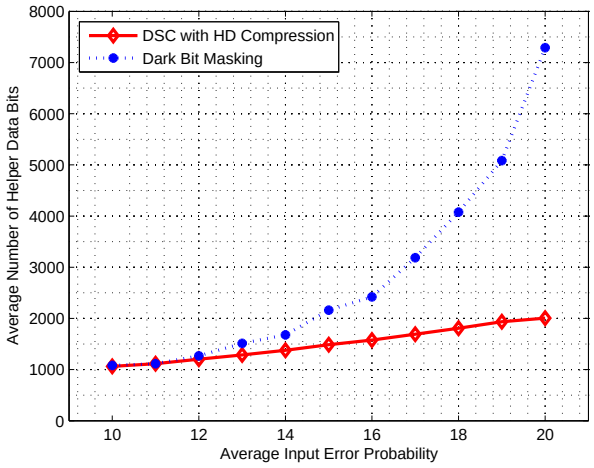


Fig. 8: Helper data sizes of DSC with helper data compression and dark bit masking for a key error probability of 10^{-9} and different input error probabilities

we have improvements between $20\times$ and $100\times$ compared to the state of the art over the $\frac{n}{k}$ range shown.

D. Yield Analysis

Figure 7 shows an empirical $(1 - cdf(l))$ function, obtained by Monte Carlo simulation, that corresponds to the overflow probability e_2 in dependency of the maximum helper data size l. According to Figure 4, in average 2.79 helper data bits have to be stored for each distance pointer for $p = 0.326$ and $m = 2$. To handle varying helper data sizes, we have to assign more helper data storage. Since we aim for a yield $y \geq 99.9\%$, we can tolerate overflows with a probability $e_2 \leq 5 \cdot 10^{-4}$.

The size of the helper data can be reduced significantly compared to the $2,176$ bits of the uncompressed version without reducing the yield. At least $1,070$ bits should be assigned for a reasonable yield. However, the error probability $e_2(l)$ decreases by several orders of magnitude for spending 5% to 10% more helper data bits. As a result, $e_2(l) \leq 5 \cdot 10^{-4}$ can be achieved in practice by $l = 1,108$, which is only 8% over the entropy of the helper data.

E. Comparison with Dark Bit Masking

The DSC setup has the same error probability as dark bit masking combined with a fuzzy commitment [42] and an identical $(2, 1, [7])$ convolutional code. We varied the average bit error probability of the distribution in [8] between 10% and 20% and obtained the parameters for a key error probability of 10^{-9} with the bounding technique discussed in [1]. Figure 8 shows the average helper data sizes of DSC with helper data compression and the fuzzy commitment with dark bit masking.

The comparison shows that DSC reduces the helper data size by up to 73% compared to the conventional approach. Therefore storing compressed differential pointers is significantly more efficient than selecting PUF bits with a bit mask when only a small fraction of PUF response bits is indexed.

VI. Implementation

After selecting the parameters for our PUF error correction module, this section presents an overview of our hardware implementation and compares the resource consumption with the state of the art. The encoding can be performed off-chip and the encoder modules only have a fraction of the complexity of the decoders. On a Xilinx Spartan-3E FPGA, the DSC encoder requires 15 slices (10 flip-flops and 25 LUTs) while the convolutional encoder only uses 10 slices (12 flip flops and 11 LUTs) which is roughly a factor of 10 smaller than the corresponding decoders (see Table I). Furthermore the encoding does not have to be part of the final implementation since it can be performed off-chip or with a different configuration bit-stream in a secure environment. Therefore, we will focus on the more important and interesting optimized decoder implementation.

A. Hardware Architecture

The block diagram in Figure 9 shows the building blocks of our DSC and convolutional code reproduction procedure.

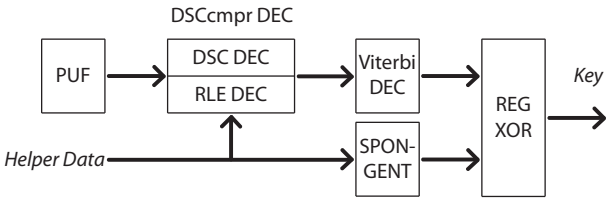


Fig. 9: DSC reproduction with helper data compression

The DSC decoder with helper data compression combines the functionality of decoding the helper data pointers, that are RLE encoded [38], and selecting the corresponding incoming

	PUF Response Bits	Helper Data Bits	Slices	Block RAM Bits	Clock Cycles
Code-Offset Golay [18]	3,696	3,824	$\geq$ 907	0	$>$ 24,024
Code-Offset RM-GMC [12]	1,536	13,952	237	32,768	10,298
C-IBS RM-GMC [17]	2,304	9,216	250	0	$\sim$ 9,000
DSC Conv. Code [1]	1,224	2,176	262	11,264	30,846
Compressed DSC Conv. Code. [2]	1,224	1,224	272	11,264	33,925
Compressed DSC Seesaw	974	1,108	249	10,752	29,243

TABLE I: FPGA implementations of reproduction procedures of the DSC and reference implementations synthesized for Xilinx Spartan 3E FPGAs

	DSCcmpr Dec	Viterbi Dec	SPONGENT	REG XOR	Entire Module
Slices Total	17	68	85	58	249
Registers	9	56	117	40	247
Logic LUTs	26	75	153	104	388
Block RAM Bits	–	10,752	–	–	10,752

TABLE II: Synthesis results of the DSC reproduction procedure implementation for Xilinx Spartan-3E FPGA

	DSCcmpr Dec	Viterbi Dec	SPONGENT	REG XOR	Entire Module
Slices Total	7	21	24	20	72
Registers	9	55	117	33	235
Logic LUTs	18	77	85	67	251
Block RAM Bits	–	10,752	–	–	10,752

TABLE III: Synthesis results of the DSC reproduction procedure implementation for Xilinx Spartan-6 FPGA

PUF response bits [2]. The helper data and the PUF outputs are read sequentially until the helper data signals that the current incoming PUF bit is the indexed one.

In this implementation, we used the Seesaw Viterbi decoder presented in [43] as ECC decoder. It contains two block RAMs that store the entire state of the Viterbi decoder each. During decoding, data is read from one RAM and written to the other. This approach has the advantage that no intermediate results are stored in registers in the data-path, which leads to a very compact area footprint.

As shown in Figure 9, the helper data is hashed onto the output of the Viterbi decoder to prevent helper data manipulation attacks as discussed in [1]. We selected SPONGENT as a lightweight hash function [44]. In [45], Jungk *et al.* demonstrated that SPONGENT is suitable for compact FPGA implementations. Therefore, we chose the implementation discussed in [45] in the smallest configuration that returns an 88-bit hash value. The REG XOR module XORs the outputs of the Viterbi decoder and the helper data hashed in the SPONGENT module and stores the result in a register. This ensures that 88 key bits are affected by each helper data bit to corrupt the key as soon as the helper data is manipulated.

B. Synthesis results

We compare our new implementation to previous work to evaluate its efficiency. Table I shows synthesis results for Xilinx Spartan 3 FPGAs and several reference implementations for the same scenario discussed in [8], [12], namely an SRAM PUF with average bit error probability 15% with distribution discussed in [8] and a desired key error probability of 10^{-6} for a 128 bit key. One can clearly see that our DSC and convolutional code implementation is by far the most efficient one in terms of required PUF outputs and helper data bits.

Tables II and III show detailed synthesis results of our architecture for Spartan-3E and Spartan-6 FPGAs. Compared to [1] and [2], the new Seesaw Viterbi decoder and SPONGENT implementation mainly reduce the number of not fully used block RAMs so that we were able to reduce the overall number to 2 while slightly decreasing the overall size of the top module. This was mainly achieved with a more balanced design by using the spare registers in slices that were already allocated for their LUTs. In addition, more advanced synthesis optimizations were applied to reduce the size.

C. Evaluation

Comparing the current state of the art to the previous DSC results in [1] and [2] shows that using precise simulation results instead of the bounded values decreases the number of PUF bits by 20%. In addition, the size of the helper data is 9% less than the previous compressed results and 50% less than the default DSC case. Our improved implementation slightly reduces the number of slices, block RAM bits and clock cycles.

The new results make DSC by far the most efficient approach for this scenario in terms of PUF and helper data bits. With DSC, we are able to generate a reliable key from 974 PUF bits and 1,108 helper data bits for $p_{max} = 0.0270$.

Therefore, we are able to reduce the number of PUF bits by 36% compared to [12] and the number of helper data bits by 71% compared to [6], which are both the most efficient approaches for each measure with a significant drawback in the other. The required number of FPGA slices for our DSC implementation is only 5% larger than the smallest reference implementation [12]. However, optimizing rigorously for area also makes our DSC implementation the slowest investigated

one in this comparison with the highest cycle counts, as shown in Table I.

VII. Conclusion

In this work, we are the first in the PUF context to quantify an algorithm-independent relationship between block size and reliability with the information theoretical concept of typicality.

We introduced Differential Sequence Coding (DSC), a pointer-based syndrome coding scheme that is able to skip unreliable PUF response bits and can treat the PUF response bits as a single, maximally reliable, block. We have shown its advantages from an information theoretical point of view and compared it to the state of the art.

Our hardware implementation requires 36% less PUF bits and 71% less helper data bits than the best reference implementations for a popular SRAM PUF scenario.

References

[1] 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.

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

[3] B. Gassend, D. Clarke, M. v. Dijk, and S. Devadas, "Silicon physical random functions," in *ACM Conference on Computer and Communications Security (CCS)*, 2002, pp. 148–160.

[4] 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.

[5] 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.

[6] 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, Heidelberg, 2007, pp. 63–80.

[7] 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, Heidelberg, 2012, pp. 283–301.

[8] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for SRAM PUFs," in *IEEE International Symposium on Information Theory (ISIT)*, 2009, pp. 2101–2105.

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

[10] M. Hiller, G. Sigl, and M. Pehl, "A new model for estimating bit error probabilities of ring-oscillator PUFs," in *International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2013.

[11] H. E. Graeb, *Analog design centering and sizing*. Springer, 2007.

[12] 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, Heidelberg, 2009, pp. 332–347.

[13] 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.

[14] O. Günlü and G. Kramer, "Privacy, secrecy, and storage with noisy identifiers," Tech. Rep., 2016. [Online]. Available: http://arxiv.org/abs/1601.06756

[15] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls, "Memory leakage-resilient encryption based on physically unclonable functions," in *Advances in Cryptology (ASIACRYPT)*, ser. LNCS, M. Matsui, Ed., vol. 5912. Springer Berlin Heidelberg, 2009, pp. 685–702.

[16] 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.

[17] 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.

[18] 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, Heidelberg, 2008, pp. 181–197.

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

[20] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2014.

[21] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, 2006.

[22] G. Kramer, "Topics in multi-user information theory," *Foundations and Trends in Communications and Information Theory*, vol. 4, no. 4-5, pp. 265–444, 2007.

[23] 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, Heidelberg, 2004, pp. 523–540.

[24] V. van der Leest, B. Preneel, and E. van der Sluis, "Soft decision error correction for compact memory-based PUFs using a single enrollment," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, E. Prouff and P. Schaumont, Eds., vol. 7428. Springer, Heidelberg, 2012, pp. 268–282.

[25] 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.

[26] M. Hiller, L. Kürzinger, G. Sigl, S. Müelich, S. Puchinger, and M. Bossert, "Low-area Reed decoding in a generalized concatenated code construction for PUFs," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2015.

[27] S. Puchinger, S. Müelich, M. Bossert, M. Hiller, and G. Sigl, "On error correction for physical unclonable functions," in *International ITG Conference on Systems, Communications and Coding (SCC)*. IEEE, 2015.

[28] 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, Heidelberg, 2012, pp. 302–319.

[29] C. Herder, L. Ren, M. van Dijk, M.-D. M. Yu, and S. Devadas, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[30] M. Hiller, M. Yu, and M. Pehl, "Systematic low leakage coding for physical unclonable functions," in *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2015.

[31] M. Yu, D. M'Raihi, R. Sowell, and S. Devadas, "Lightweight and secure PUF key storage using limits of machine learning," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, B. Preneel and T. Takagi, Eds., vol. 6917. Springer, Heidelberg, 2011, pp. 358–373.

[32] 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.

[33] M. Hiller, F. De Santis, D. Merli, and G. Sigl, "Reliability bound and channel capacity of IBS-based fuzzy embedders," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, 2012, pp. 213–220.

[34] P. Koeberl, J. Li, R. Maes, A. Rajan, C. Vishik, and M. Wjcik, "Evaluation of a PUF device authentication scheme on a discrete 0.13um SRAM," in *International Conference on Trusted Systems (INTRUST)*,

ser. Lecture Notes in Computer Science, L. Chen, M. Yung, and L. Zhu, Eds., vol. 7222. Springer Berlin / Heidelberg, 2011, pp. 271–288.

[35] M. Yu, M. Hiller, and S. Devadas, "Maximum likelihood decoding of device-specific multi-bit symbols for reliable key generation," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2015.

[36] C. Bösch, "Efficient fuzzy extractors for reconfigurable hardware," Master's Thesis, Ruhr-University Bochum, 2008.

[37] M. Hiller, "Optimized fuzzy extractor for PUFs on FPGAs," Diplomarbeit, Ulm University, 2011.

[38] S. W. Golomb, "Run-length encodings (corresp.)," *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399–401, 1966.

[39] R. Maes, "An accurate probabilistic reliability model for silicon PUFs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, G. Bertoni and J.-S. Coron, Eds. Springer, Heidelberg, 2013, pp. 73–89.

[40] D. J. Costello Jr. and G. D. Forney Jr., "Channel coding: The road to channel capacity," *Proceedings of the IEEE*, vol. 95, pp. 1150–1177, 2007.

[41] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

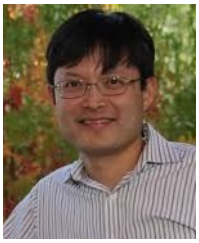
[42] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *ACM Conference on Computer and Communications Security (CCS)*, 1999, pp. 28–36.

[43] M. Hiller, L. Rodrigues Lima, and G. Sigl, "Seesaw: An area optimized FPGA Viterbi decoder for PUFs," in *Euromicro Conference on Digital System Design (DSD)*. IEEE, 2014.

[44] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "SPONGENT: A lightweight hash function," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, B. Preneel and T. Takagi, Eds., vol. 6917. Springer, Heidelberg, 2011, pp. 312–325.

[45] B. Jungk, L. Rodrigues Lima, and M. Hiller, "A systematic study of lightweight hash functions on FPGAs," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*. IEEE, 2014.

Matthias Hiller studied electrical engineering at Ulm University, Ulm, Germany and Portland State University, Portland, OR, USA. After writing his diploma thesis at Fraunhofer AISEC, Munich, he joined the Chair of Security in Information Technology at the Technical University of Munich, in 2011 where he is currently working towards a PhD. His current research interests include the design, implementation and analysis of error correction for PUFs.



Meng-Day (Mandel) Yu is Chief Scientist at Verayo, Research Affiliate for CSAIL/MIT, and is pursuing a PhD based on a research career with COSIC/KU Leuven. He was Manager of R&D Engineering at TSI, and developed a secure digital baseband radio. Prior, he was an ASIC Design Engineer and later a Systems Engineer at TeraLogic (acquired by Zoran), and developed advanced signal processing and security designs and contributed to six production silicon tape-outs. He holds BSEE/MSEE degrees from Stanford, where he was a Mayfield Fellow. With research interests in coding and security, he served on ACM and IACR program committees.

Georg Sigl Georg Sigl finished his PhD in Electrical Engineering at the Technical University of Munich in 1992 in the area of layout synthesis. Afterwards he introduced new design-for-testability concepts in telecommunication ASICs at Siemens. In 1996 he joined the automotive microcontroller department at Siemens HL (later Infineon) to develop a universal library for peripherals to be used in 16- and 32-bit microcontrollers. From 2000 he was responsible for the development of new secure microcontroller platforms in the Chip Card and Security division. Under his responsibility, two award winning platforms - the SLE88 (Cartes Sesames Award 2001) and the SLE78 (Cartes Sesames Award 2008; Innovation Award of the German Industry 2010) have been designed. In June 2010, he founded a new chair at the Technical University of Munich for Security in Electrical Engineering and Information Technology. In parallel, he is driving embedded security research as director at the Fraunhofer Institute for Applied and Integrated Security AISEC Munich.