

A Robust SRAM-PUF Key Generation Scheme Based on Polar Codes

Bin Chen, Tanya Ignatenko, Frans M.J. Willems
Eindhoven University of Technology
Eindhoven, The Netherlands
Email: {b.c.chen, t.ignatenko, f.m.j.willems}@tue.nl

Roel Maes, Erik van der Sluis, Georgios Selimis
Intrinsic-ID, Eindhoven, The Netherlands
Email: {roel.maes, erik.van.der.sluis, georgios.selimis}@intrinsic-id.com

Abstract—Physical unclonable functions (PUFs) are relatively new security primitives used for device authentication and device-specific secret key generation. In this paper we focus on SRAM-PUFs. The SRAM-PUFs enjoy uniqueness and randomness properties stemming from the intrinsic randomness of SRAM memory cells, which is a result of manufacturing variations. This randomness can be translated into the cryptographic keys thus avoiding the need to store and manage the device cryptographic keys. Therefore these properties, combined with the fact that SRAM memory can be often found in today's IoT devices, make SRAM-PUFs a promising candidate for securing and authentication of the resource-constrained IoT devices. PUF observations are always effected by noise and environmental changes. Therefore secret-generation schemes with helper data are used to guarantee reliable regeneration of the PUF-based secret keys. Error correction codes (ECCs) are an essential part of these schemes. In this work, we propose a practical error correction construction for PUF-based secret generation that are based on polar codes. The resulting scheme can generate 128-bit keys using 1024 SRAM-PUF bits and 896 helper data bits and achieve a failure probability of 10^{-9} or lower for a practical SRAM-PUFs setting with bit error probability of 15%. The method is based on successive cancellation combined with list decoding and hash-based checking that makes use of the hash that is already available at the decoder. In addition, an adaptive list decoder for polar codes is investigated. This decoder increases the list size only if needed.

I. INTRODUCTION

The Internet of Things (IoT) is a network, in which billions of devices are connected. While such a network is expected to bring tremendous economic benefits to industry and society, its use also comes with security problems. Most of IoT devices operate in resource-constrained and distributed environments. As a result traditional password-based security and centralized key management systems with costly secure elements cannot be easily deployed in IoT networks.

Physical unclonable functions (PUFs) are low-cost hardware intrinsic security primitives that possess an intrinsic randomness (unique device ‘*fingerprint*’) due to the inevitable process variations during manufacturing. Therefore, PUFs can be used to realize cryptographic applications, such as identification, authentication and cryptographic key generation [1], [2], that require random, unique and unpredictable keys. Since the device-unique randomness can be translated into a cryptographic key, PUFs can act as trust anchors avoiding the need for key storage.

There are several types of structures to realize PUFs, such as Flip-Flops PUFs [3], Butterfly PUFs [4], Ring Oscillator

PUFs [5] and static random-access memory (SRAM) PUFs [6]. Among them, SRAM-PUFs are one of the most popular PUF constructions because they are easy to manufacture and do not require extra investments. SRAM-PUFs also enjoy the properties that, while being easily evaluated (after a device power-up), they are unique, reproducible, physically unclonable and unpredictable [7]. However, SRAM-PUFs cannot be straightforwardly used as cryptographic keys, since their observations are not exactly reproducible due to environmental condition changes such as time, temperature, voltage and random noise. Therefore, error correction techniques are necessary to mitigate these effects and generate reliable keys.

Error correction techniques become essential blocks of secret-generation schemes [8]–[10]. In these schemes two terminals observe measurements of the same PUF. The encoder (first terminal) creates a secret-key and a so-called helper data, based on its PUF observation. This helper data facilitates reconstruction of the secret key from the noisy observation of the PUF at the decoder (second terminal). Since the helper data is communicated from the encoder to the decoder, the secrecy leakage (information that it provides about the secret key) should be negligible.

For practical implementation of key generation schemes on resource-constrained PUF devices, especially for IoT applications, it is crucial to construct good error correction codes to maintain a good trade-off between reliability, implementation complexity and secrecy leakage. Most of existing works [10]–[14] that use simple error correction codes are impractical for real applications, where environmental variations lead to error rates of up to 25% in PUF observations. These high error rates require (simple) ECCs of low rates. On the other hand, security applications impose requirements on the minimum (fixed) secret key size that need to be generated from a given finite block-length SRAM cells. As a result, one need to use powerful high-rate ECCs, which typically have high complexity.

Therefore, in this work we propose to use polar codes that are capacity-achieving and have low encoding and decoding complexity. Polar codes have been also investigated for the Slepian-Wolf problem [15] and key generation [16]. For finite block-length, it was shown that good performance of polar codes can be achieved by implementing enhanced decoding algorithms based on the classical successive cancellation decoder (SCD) [17]–[19].

Here we propose a new and efficient key generation build-

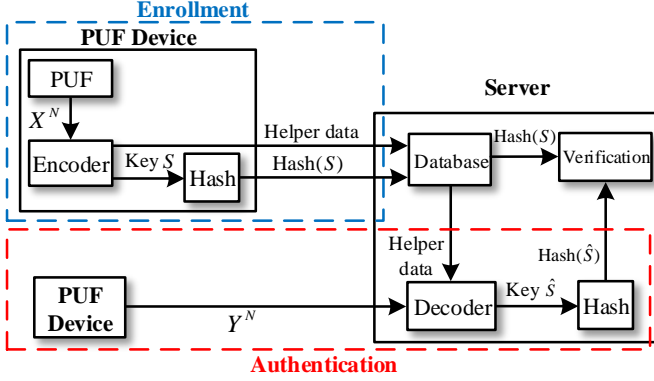


Fig. 1. Secret-generation system for PUFs

ing block for SRAM-PUFs key generation based on application of polar codes in a syndrome-based secret-generation scheme [20]. To guarantee the performance in terms of reliability and security, and to decrease the required memory size of this scheme, we (1) exploit the efficient decoding algorithm based on successive cancellation and list decoding to reliably regenerate the secret, (2) prove zero-leakage for the proposed scheme, and (3) use a puncturing scheme to shorten the code length and reduce the complexity. Our simulation results show that 10^{-9} key regeneration failure probability can be achieved with less SRAM-PUF and helper data bits than before. Using puncturing for polar coding schemes results in flexibility in getting the required code rates, which is crucial since key sizes in practical applications are typically fixed.

II. SECRET GENERATION BASED ON SRAM-PUFS

SRAM-PUFs are a result of the read-outs of the power-up state of an SRAM array. The cell values of SRAM array after power up go into one of two states: 0 or 1. It has been experimentally demonstrated [21] that due to the independent random nature of process variations on each SRAM cell, SRAM patterns demonstrates excellent PUF behavior, i.e. empirical probability of number of cells that go in state 1 is close to 0.5. Therefore in this paper we assume that SRAM-PUFs are binary-symmetric, hence for enrollment and authentication PUF pairs (X^N, Y^N) it holds that

$$\Pr\{(X^N, Y^N) = (x^N, y^N)\} = \prod_{n=1}^N Q(x_n, y_n), \quad (1)$$

where $Q(0, 1) = Q(1, 0) = p/2$ and $Q(0, 0) = Q(1, 1) = (1 - p)/2$ and $0 \leq p \leq 1/2$.

It is our goal to share a PUF-based secret key S between a PUF-device and a server, see Fig. 1. During the enrollment phase, the encoder observes SRAM-PUF measurement X^N and based on it generates a secret key S and helper data W , as $(S, W) = \text{enc}(X^N)$. Here $\text{enc}(\cdot)$ is an encoder mapping. Since the key is used for cryptographic purposes, it has to be uniformly distributed. Moreover, the helper data is assumed to be publicly available, and thus it should leak no information about the key, i.e. $I(S; W) = 0$.

Next during the secret regeneration phase, the decoder observes the authentication SRAM-PUF measurement Y^N and the corresponding helper data W . The decoder now forms

an estimate of the secret key as $\hat{S} = \text{dec}(Y^N, W)$, with $\text{dec}(\cdot)$ being a decoder mapping. To make an authentication decision the server compares the hash of the estimated secret key, $\text{Hash}(\hat{S})$ with $\text{Hash}(S)$.¹ The authentication decision is positive only if the hashes are the same and thus the secret reconstruction was successful. Hence to ensure the system reliability, the error or failure probability $\Pr\{\hat{S} \neq S\}$ should be small.

The secret-generation problem is closely related to the Slepian-Wolf coding problem and is often realized using syndrome construction, where the helper data is the syndrome of the enrollment observation. Due to high error rates in SRAM-PUFs, 15%–25%, combined with demands of having $\Pr\{\hat{S} \neq S\}$ of 10^{-9} in practical application, powerful codes are required for reliable key generation. In this paper we explore the use of polar codes for SRAM-PUF secret generation based on syndrome construction.

III. POLAR CODES

As a family of linear block codes, a binary polar code can be specified by $(N, K, \mathcal{F}, u^{\mathcal{F}})$, where $N = 2^n$ is the block length, K is the number of information bits encoded per codeword, \mathcal{F} is a set of indices for the $N - K$ frozen bits positions from $\{1, 2, \dots, N\}$ and $u^{\mathcal{F}}$ is a vector of frozen bits. The frozen bits are assigned by a fixed binary sequence, which is known to both the encoder and the decoder.

A. Code Construction of Polar Codes

Polar codes are channel specific codes, which means that a polar code designed for a particular channel might not have an optimal performance for other channels. Therefore, calculation of channel reliability and selection of good channels is a critical step for polar coding, which is often referred to as *polar code construction*. The original construction of polar codes is based on the Bhattacharyya bound approximation [17]. Later works [22], [23] improve on this approximation, however, at the cost of higher complexity.

B. Encoding of Polar Codes

For an (N, K, \mathcal{F}) polar code, the encoding operation for a vector of information bits, \mathbf{u} , is performed using a generator matrix,

$$\mathbf{G}_N = \mathbf{G}_2^{\otimes \log N}, \quad (2)$$

where $\mathbf{G}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and \otimes denotes the Kronecker product. Given the data sequence U , the codewords are generated as

$$V = U\mathbf{G}_N = U^{\mathcal{F}^c}(\mathbf{G}_N)_{\mathcal{F}^c} + U^{\mathcal{F}}(\mathbf{G}_N)_{\mathcal{F}}, \quad (3)$$

where $\mathcal{F}^c \triangleq \{1, 2, \dots, N\} \setminus \mathcal{F}$ corresponds to the non-frozen bits indices. Then $U^{\mathcal{F}^c}$ is the data sequence, and $U^{\mathcal{F}}$ are the frozen bits, which are usually set to zero.

¹A one-way cryptographic hash function is used to generate a hash value of the key and verify whether the key is recovered exactly. The design and security properties of such one-way cryptographic hash functions is beyond the scope of this paper.

C. Decoding of Polar Codes

Polar codes achieve the channel capacity asymptotically in code length, when decoding is done using the successive-cancellation (SC) decoding algorithm, which sequentially estimates the bits \hat{u}_i , where $0 \leq i \leq N$.

When polar decoder decodes the i th bit, \hat{u}_i is estimated based on the channel output y^N and the previous bit decisions $\hat{u}_1, \hat{u}_2, \dots, \hat{u}_{i-1}$, denoted by \hat{u}_1^{i-1} . It uses the following rules:

$$\hat{u}_i = \begin{cases} u_i, & \text{if } i \in \mathcal{F} \\ 0, & \text{if } i \in \mathcal{F}^c \text{ and } L(y_1^N, \hat{u}_1^{i-1}) \geq 1, \\ 1, & \text{if } i \in \mathcal{F}^c \text{ and } L(y_1^N, \hat{u}_1^{i-1}) < 1 \end{cases} \quad (4)$$

where $L_N^i(y_1^N, \hat{u}_1^{i-1}) = \frac{Pr(0|y_1^N, \hat{u}_1^{i-1})}{Pr(1|y_1^N, \hat{u}_1^{i-1})}$ is the i th likelihood ratio (LR) at length N , which determines the probability of a non-frozen bit. LRs can be computed recursively using two formulas:

$$\begin{aligned} L_{N/2}^{2i-1}(y_1^N, \hat{u}_1^{2i-2}) \\ = \frac{L_{N/2}^i(y_1^{N/2}, \hat{u}_o^{2i-2} \oplus \hat{u}_e^{2i-2}) L_{N/2}^i(y_1^{N/2+1}, \hat{u}_e^{2i-2}) + 1}{L_{N/2}^i(y_1^{N/2}, \hat{u}_o^{2i-2} \oplus \hat{u}_e^{2i-2}) + L_{N/2}^i(y_1^{N/2+1}, \hat{u}_e^{2i-2})} \end{aligned} \quad (5)$$

and

$$\begin{aligned} L_N^{2i}(y_1^N, \hat{u}_1^{2i-1}) &= \left[L_{N/2}^i(y_1^{N/2}, \hat{u}_o^{2i-2} \oplus \hat{u}_e^{2i-2}) \right]^{1-2\hat{u}_{2i-1}} \\ &\cdot L_{N/2}^i(y_1^{N/2+1}, \hat{u}_e^{2i-2}), \end{aligned} \quad (6)$$

where \hat{u}_o^{2i-2} and \hat{u}_e^{2i-2} denote, respectively, the odd and even indices part of \hat{u}_1^{2i-2} . Therefore, calculation of LRs at length N can be reduced to calculation of two LRs at length $N/2$, and then recursively broken down to block length 1. The initial LRs can be directly calculated from the channel observation.

Since the cost of implementing these multiplications and divisions operations in hardware is very high, they are usually avoided and performed in the logarithm domain using the following f and g functions:

$$f(L_1, L_2) = 2 \tanh^{-1} \left(\tanh \left(\frac{L_1}{2} \right) \tanh \left(\frac{L_2}{2} \right) \right) \quad (7)$$

$$\approx \text{sign}(L_1 \cdot L_2) \cdot \min(|L_1|, |L_2|), \quad (8)$$

$$g(L_1, L_2) = (-1)^{1-2\hat{u}_{2i-1}} \cdot L_1 + L_2, \quad (9)$$

where $L_1 = \log \left(L_{N/2}^i(y_1^{N/2}, \hat{u}_o^{2i-2} \oplus \hat{u}_e^{2i-2}) \right)$ and $L_2 = \log \left(L_{N/2}^i(y_1^{N/2+1}, \hat{u}_e^{2i-2}) \right)$ are log-likelihood ratios (LLRs). In practical implementations, the *minimum* function can be used to approximate the f function, according to (8).

IV. SECRET-GENERATION SCHEMES BASED ON POLAR CODES

In this section we show how secrets and helper data can be constructed using a polar code in PUF-based key generation schemes. A generic secret-generation system is illustrated in Fig. 1. There X^N is a PUF measurement during enrollment and Y^N is a noisy PUF measurement at authentication, which are observed by the encoder and decoder, respectively. First, in

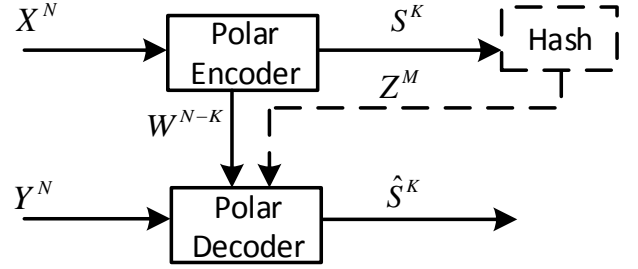


Fig. 2. Polar codes based syndrome coding scheme. Dashed line indicate the extra operation for polar code with HA-SCL decoding.

Section IV-A, we present the secret-generation system based on syndrome construction using the polar coding. Then, in Section IV-B we discuss how the decoding for secret generation can be redesigned to optimize the system performance for PUF applications. Finally, Section IV-C provides our security analysis for the proposed construction.

A. Polar Codes based Syndrome Construction

Fig. 2 illustrates the polar code based syndrome coding scheme that realizes an enrollment phase (encoder) and key regeneration phase (decoder).

1) *Enrollment phase*: In the enrollment phase, a codeword $C^N = X^N \mathbf{G}_N^{-1}$ is generated for each PUF observation X^N . Then, the syndrome encoder selects the secret key S^K and helper data W^{N-K} based on the constructed codeword. Since $\mathbf{G}_N^{-1} = \mathbf{G}_N$, the helper data and the secret key are generated during a polar encoding procedure by extracting the bits as²

$$\begin{aligned} W^{N-K} &\triangleq (X^N \mathbf{G}_N^{-1})_{\mathcal{F}} = (X^N \mathbf{G}_N)_{\mathcal{F}} = C^N[\mathcal{F}], \\ S^K &\triangleq (X^N \mathbf{G}_N^{-1})_{\mathcal{F}^c} = (X^N \mathbf{G}_N)_{\mathcal{F}^c} = C^N[\mathcal{F}^c], \end{aligned} \quad (10)$$

where \mathcal{F} and \mathcal{F}^c are the index sets for the syndrome and the secret key. These sets are defined as

$$\begin{aligned} \mathcal{F} &\triangleq \{i \in \{1, 2, \dots, N\} : H(C_i | Y^N, C_1^{i-1}) \geq \delta\}, \\ \mathcal{F}^c &\triangleq \{1, 2, \dots, N\} \setminus \mathcal{F}, \end{aligned} \quad (11)$$

where $\delta \triangleq 2^{-N\beta}$ and $\beta \in [0, 1/2]$.

An example of the syndrome encoding procedure for a $(8, 3, \{1, 2, 3, 4, 6\})$ polar codes is shown in Fig. 3, where the data flows from right to left. Due to flexibility of the polar code construction, an arbitrary code rate $R = K/N$ can be selected without re-constructing the code.

2) *Key regeneration phase*: In the key regeneration phase, the syndrome decoder observes the authentication sequence Y^N and also receives the public helper data W^{N-K} . The decoder can compute \hat{S}^K using a modified version of the SC decoding algorithm, given in Section III-C, i.e., as

$$\hat{S}^K = \text{SCD}(Y^N, W^{N-K}), \quad (12)$$

where the polar decoder $\text{SCD}(\cdot)$ is given by Algorithm 1.

² The difference compared to conventional polar codes is that the helper data specifies a coset of the linear polar code instead of fixed all-zeros.

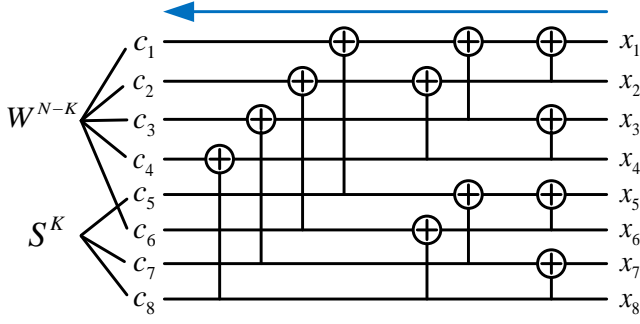


Fig. 3. Encoding graph of $(8, 3, \{1, 2, 3, 4, 6\})$ polar codes

Algorithm 1 Decoding Algorithm for Syndrome construction

Input: The observations Y^N from PUFs, the public helper data W^{N-K} .

Output: The estimated secret \hat{S}^K

```

1: for  $i = 1$  to  $N$  do
2:   Compute  $LLR_i$  with the observed  $Y^N$  from Eq. (7-9)

3:   if  $i \in \mathcal{F}$  then
4:      $\hat{C}_i = W_j$ 
5:   else if  $i \in \mathcal{F}^c$  and  $LLR_i > 0$  then
6:      $\hat{C}_i = 0$ 
7:   else
8:      $\hat{C}_i = 1$ 
9:   end if
10:   $\hat{S}^K \leftarrow \hat{C}^N[\mathcal{F}^c]$ 
11: end for
12: return  $\hat{S}^K$ 

```

B. Decoder Optimization for PUF-based Secret Generation

Note that although the SC decoder could asymptotically achieve channel capacity as N increases, the performance of the SC decoder is still not good enough at short and moderate block length size for error correction in PUFs due to the poor polarization. Therefore, next we present hash-aided SC list (HA-SCL) decoding that allows us to achieve good trade-off between error-correction performance and complexity.

In order to optimize the error-correction performance, we would like to track multiple possible decision paths instead of only one as the SC decoder does. However, considering the all 2^K possible paths is impractical and too complex. The SCL decoding algorithm [19] uses a breadth search method to explore the possible decoding paths efficiently while saving L most reliable paths as candidates at each level. Thus this technique also allows us to restrict the decoding complexity.

Next note that in the SCL decoding process, the correct codewords are on the decoding list but they are not always the most likely ones, which leads to decoding errors. This issue can be solved by combining the SCL algorithm with a cyclic redundancy check (CRC) code, which could further improve the error correction performance [18]. For security purposes, we replace the CRC function by a more secure hash function, which is already part of the authentication system. This hash

is used to detect and select the valid path from the output of list decoder. In this way, our HA-SCL decoder outputs L candidate sequences and selects the hash-valid sequence.

By using the HA-SCL decoder, M bits hash value $Z^M = H(S^K)$ is produced by the hash function at the encoder and is used at the decoder. Since the decoder knows W^{N-K} and Z^M in advance, it could recover the secret key by performing the polar decoding, as shown in Fig. 2, using

$$\hat{S}^K = \text{SCLD}(Y^N, W^{N-K}, Z^M), \quad (13)$$

where $\text{SCLD}(\cdot)$ is the polar decoder with the SCL decoding algorithm of [19].

C. Security Analysis

In this section, we analyze the secrecy for the proposed syndrome based polar coding scheme. Note that security of our construction is characterized by the information that the helper data leaks about the generated secret key. Therefore we must show that $I(S^K; W^{N-K}) = 0$. We re-write (10) as

$$\begin{aligned} W^{N-K} &= (X^N \mathbf{G})_{\mathcal{F}} = X^N \mathbf{G}_{\mathcal{F}}, \\ S^K &= (X^N \mathbf{G})_{\mathcal{F}^c} = X^N \mathbf{G}_{\mathcal{F}^c}, \end{aligned} \quad (14)$$

where generator matrix $\mathbf{G}_{\mathcal{F}}$ for frozen bits (helper data) and generator matrix $\mathbf{G}_{\mathcal{F}^c}$ for information bits (key) with dimensions $N \times (N-K)$ and $N \times K$ are obtained by selecting the corresponding columns of \mathbf{G}_N . Then, we obtain

$$\begin{aligned} I(S^K; W^{N-K}) &= H(S^K) + H(W^{N-K}) - H(S^K, W^{N-K}) \\ &= H(X^N \mathbf{G}_{\mathcal{F}^c}) + H(X^N \mathbf{G}_{\mathcal{F}}) - H(X^N \mathbf{G}) \\ &\stackrel{a}{=} \text{rank}(\mathbf{G}_{\mathcal{F}^c}) + \text{rank}(\mathbf{G}_{\mathcal{F}}) - \text{rank}(\mathbf{G}) \\ &\stackrel{b}{=} K + (N - K) - N = 0, \end{aligned}$$

where in (a) we use the uniformity of the SRAM-PUF observations; in (b) the fact that the generator matrices are linearly independent, as $\mathbf{G}_{\mathcal{F}}$ and $\mathbf{G}_{\mathcal{F}^c}$ are non-overlapping, since $\mathcal{F}^c \cap \mathcal{F} = \emptyset$. Thus we prove that the proposed polar syndrome coding scheme has zero-leakage.

V. PERFORMANCE AND COMPLEXITY COMPARISONS

In this section, we present the performance results of the polar code based error correction schemes for SRAM-PUFs with average bit error probability between 15% and 30%. Inputs to the polar decoder, including the information set, frozen bit vector and channel output vector, determine the error correction ability and computational complexity. In order to create reliable PUF-based secret generation systems, we focus on the scheme with 128-bit keys and failure probabilities in the range of 10^{-6} to 10^{-9} .

We construct polar codes with block length $N = 1024$. In order to provide a flexible code rate and use less SRAM-PUFs bits in PUF-based secret generation with fixed size key, arbitrary block-length polar codes can be obtained by puncturing. For any puncturing pattern, $N' = N - m$ PUFs bits and m random bits used as punctured bits are the input to the polar encoder. At the decoder, m zero-valued LLRs for decoding are assigned to the corresponding punctured bits. In the following sections, both SC and HA-SCL decoding algorithms for polar

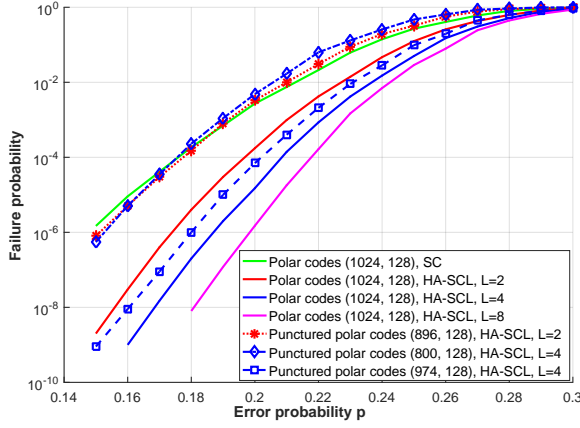


Fig. 4. Failure rate performance comparison of the different decoding schemes.

codes and punctured polar codes are simulated to compare the resulting error correction performance and complexity.

A. Failure Probability

The most important performance criterion for PUF-based secret generation is the error or failure probability of the key regeneration. Fig. 4 shows the performance of polar code based syndrome coding schemes with the SC and HA-SCL decoding algorithms.

We can see that the failure rate for polar codes with SC decoding is close to 10^{-6} at 15% and HA-SCL decoding can further reduce the failure rate to less than 10^{-9} at 15% as list size L increases. However, the latter comes at the cost of extra computational complexity and memory. We can also observe that punctured polar codes with block lengths not being a power of two achieve similar failure rates as conventional polar codes by using larger L for performance compensation. For strong reliability applications, the proposed polar code (1024, 128) with $L = 2$ and the punctured polar code (974, 128) with $L = 4$ can be used to achieve an error rate of 10^{-9} at error probability of 15%.

B. Complexity and Memory Requirements

The number of required SRAM-PUF bits and helper data size is another important performance criterion closely related to implementation complexity. The proposed polar code based syndrome coding scheme requires 1024 SRAM-PUF bits and 896 helper data bits, if we implement the SC decoding algorithm. The corresponding decoding computational complexity is given by $\mathcal{O}(N \log N)$ for this case. Since a list decoder outputs a group of L reliable candidates, the decoding computational complexity of the HA-SCL algorithm increases to $\mathcal{O}(LN \log N)$.

Table I summarizes performance properties for the proposed polar codes and reference designs, including the achievable key regeneration failure rate, the required SRAM-PUF size, helper data size and error probability p of the SRAM-PUFs. An SRAM-PUF with an error probability of 15% or lower and failure rates 10^{-6} and 10^{-9} are targeted for different use cases. From Table I, we can clearly see that our polar code based schemes outperform the previous designs

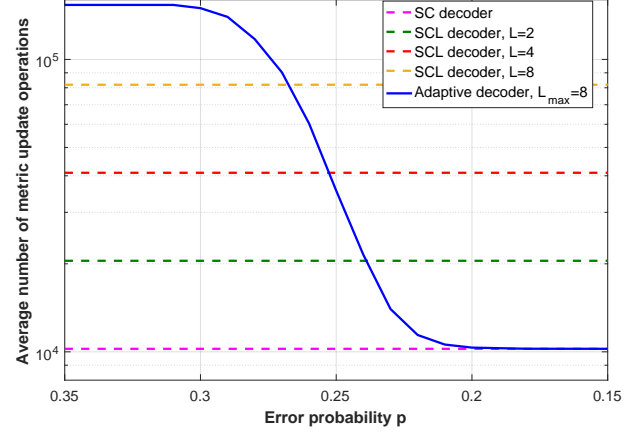


Fig. 5. Complexity comparison of the different polar decoding schemes.

in terms of the error correction performance, SRAM-PUF bits and helper data bits requirements. Note that the required SRAM-PUF size could be further reduced by using punctured polar codes and increasing the list size L , but at the cost of computational and memory complexity for decoder.

C. Adaptive Decoder

The HA-SCL decoding algorithm achieves a good performance but has higher complexity $\mathcal{O}(LN \log N)$ than SC decoding, as L and N increase, and as a consequence relatively high latency. The complexity issue of the SCL can be improved by using an adaptive decoder, which consists of two components, SC and SCL decoders. This adaptive decoder, only implements the SCL decoder and increases the value of L , when the SC decoder output has an invalid hash vector.

Fig. 5 shows the comparison of complexity between the adaptive decoder, single SC decoder and single HA-SCL decoder with different L . Computational complexity is defined in terms of the average number of metric update operations $f(\cdot)$ and $g(\cdot)$ in (7) and (9). The maximum L_{max} is set to 8 to ensure the reliability and security. As expected, the SC decoder has the lowest complexity with poor performance with respect to the failure rate; and the SCL decoder has higher complexity in terms of L . Furthermore, we see that the average number of computations of the adaptive decoder is drastically reduced as the error probability p decreases.

The adaptive decoder also reduce the effect of decoding latency, since there is very little chance to use the complex SCL decoder. Therefore, the adaptive decoder could achieve the same reliability with a single SCL decoder and provide similar computational complexity and decoding latency with a single SC decoder when p is small.

VI. DISCUSSION

Note that another way to realize PUF-based authentication is using the *code-offset construction*. In this construction, a selected error correction codes is used to encode a key chosen during the enrollment phase into codeword $C^N = SG$. The helper data W^N is defined as the offset $W^N = C^N \oplus X^N$. During the key regeneration phase, the helper data W^N is added to a PUF authentication sequence Y^N . The decoder observe a codeword corrupted with the measurement noise

TABLE I
COMPARISON OF FAILURE RATE, PUF AND HELPER DATA SIZE FOR POLAR CODES AND REFERENCE DESIGNS.

Code construction	Failure probability	PUF (bit)	Helper Data (bit)	p
Code-Offset RM-GMC [11]	10^{-6}	1536	13952	15%
Compressed DSC [14]	10^{-6}	974	1108	15%
Polar SC	10^{-6}	1024	896	15%
Punctured polar HA-SCL, L=2	10^{-6}	896	896	15%
Punctured polar HA-SCL, L=4	10^{-6}	800	896	15%
BCH Rep. [12]	10^{-9}	2226	2052	13%
GC RM [13]	$5.37 \cdot 10^{-10}$	2048	2048	14%
GC RS [13]	$3.47 \cdot 10^{-10}$	1024	1024	14%
Polar HA-SCL, L=2	10^{-9}	1024	896	15%
Punctured polar HA-SCL, L=4	10^{-9}	974	896	15%

e^N , i.e., $\tilde{C}^N = W^N \oplus Y^N = C^N \oplus e^N$. Therefore, polar codes based code-offset construction can also be directly applied to realize a secret-sharing system with chosen secret keys.

By designing the same polar code construction, the two secret key generation schemes with the syndrome construction and code-offset construction are equivalent in terms of error correction performance but they differ in their helper data storage requirements. In particular, the syndrome construction requires less storage for the helper data. Moreover, the proposed polar codes based syndrome coding construction potentially has more applications, since the secret keys need not be known to the manufacturer, while the code-offset construction requires the key to be assigned to the PUF devices during the manufacturing process.

VII. CONCLUSION

In this paper, we investigated practical secret-generation schemes based on polar code with syndrome construction that treat the SRAM-PUF observations as a codeword of a polar code and generate helper data as a syndrome of SRAM-PUFs using frozen bits of the polar code. Our simulation results show that with this approach high secret generation reliability can be achieved together with high security. Furthermore, the proposed scheme requires less SRAM-PUF bits and helper data bits compared to existing schemes, which leads to the reduction in memory requirements.

The proposed scheme has higher complexity requirements on hardware than simple algebraic codes used in the previous schemes. Therefore it can be used in the scenarios for secret key generation between small IoT devices and servers, which have sufficient resources for decoding. For future work, we intend to investigate the techniques for encoder and decoder optimization to further reduces the complexity of decoding thus making it also suitable for small IoT devices.

ACKNOWLEDGMENT

This work was funded by Eurostars-2 joint programme with co-funding from the EU Horizon 2020 programme under the E! 9629 PATRIOT project.

REFERENCES

- [1] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, June 2007, pp. 9–14.
- [2] S. U. Hussain, M. Majzoobi, and F. Koushanfar, "A built-in-self-test scheme for online evaluation of physical unclonable functions and true random number generators," *IEEE Trans. on Multi-Scale Computing Systems*, vol. 2, no. 1, pp. 2–16, Jan 2016.
- [3] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic PUFs from flip-flops on reconfigurable devices," in *3rd Benelux workshop on information and system security*, vol. 17, 2008, p. 2008.
- [4] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls, "The butterfly PUF protecting IP on every FPGA," in *IEEE Int. Workshop on Hardware-Oriented Security and Trust*, June 2008, pp. 67–70.
- [5] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *9th ACM Conference on Computer and Communications Security*, Washington, DC, USA, 2002, pp. 148–160.
- [6] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for ip protection," in *Cryptographic Hardware and Embedded Systems (CHES)*, Vienna, Austria, 2007, pp. 63–80.
- [7] R. Maes and I. Verbauwhede, *Physically unclonable functions: a study on the state of the art and future research directions*, 2010, pp. 3–37.
- [8] U. M. Maurer, "Secret key agreement by public discussion from common information," *IEEE Trans. on Inf. Theory*, vol. 39, no. 3, pp. 733–742, May 1993.
- [9] R. Ahlswede and I. Csiszar, "Common randomness in information theory and cryptography. i. secret sharing," *IEEE Trans. Inf. Theory*, vol. 39, no. 4, pp. 1121–1132, Jul 1993.
- [10] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Int. Conf. on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, 2004, pp. 523–540.
- [11] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs," in *Cryptographic Hardware and Embedded Systems (CHES)*, Lausanne, Switzerland, 2009, pp. 332–347.
- [12] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *Cryptographic Hardware and Embedded Systems (CHES)*, Leuven, 2012, pp. 302–319.
- [13] S. Puchinger, S. Muelich, M. M. Bossert, M. Hiller, and G. Sigl, "On error correction for physical unclonable functions," in *Int. ITG Conf. on Systems, Communications and Coding*, 2015, pp. 1–6.
- [14] M. Hiller, M. D. Yu, and G. Sigl, "Cherry-picking reliable PUF bits with differential sequence coding," *IEEE Trans. on Inf. Forens. and Sec.*, vol. 11, no. 9, pp. 2065–2076, Sept 2016.
- [15] S. B. Korada and R. Urbanke, "Polar codes for Slepian-Wolf, Wyner-Ziv, and Gelfand-Pinsker," in *IEEE Information Theory Workshop on Information Theory (ITW)*, Cairo, Egypt, Jan 2010, pp. 1–5.
- [16] R. A. Chou, M. R. Bloch, and E. Abbe, "Polar coding for secret-key generation," *IEEE Trans. on Inf. Theory*, vol. 61, no. 11, pp. 6213–6237, Nov 2015.
- [17] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.
- [18] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1668–1671, October 2012.
- [19] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.
- [20] S. C. Draper, A. Khisti, E. Martinian, A. Vetro, and J. S. Yedidia, "Using distributed source coding to secure fingerprint biometrics," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, April 2007, pp. 129–132.
- [21] G.-J. Schrijen and V. van der Leest, "Comparative analysis of SRAM memories used as PUF primitives," in *Conf. on Design, Automation and Test in Europe*, San Jose, CA, USA, 2012, pp. 1319–1324.
- [22] R. Mori and T. Tanaka, "Performance and construction of polar codes

on symmetric binary-input memoryless channels,” in *IEEE Int. Symp. Inf. Theory*, June 2009, pp. 1496–1500.

- [23] I. Tal and A. Vardy, “How to construct polar codes,” *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, Oct 2013.