Nested Tailbiting Convolutional Codes for Secrecy, Privacy, and Storage

Thomas Jerkovits
thomas.jerkovits@dlr.de
German Aerospace Center
Weßling, Germany

Onur Günlü
guenlue@tu-berlin.de
TU Berlin
Berlin, Germany

Vladimir Sidorenko
Gerhard Kramer
vladimir.sidorenko@tum.de
gerhard.kramer@tum.de
TU Munich
Munich, Germany

ABSTRACT

The key agreement problem with biometric or physical identifiers and two terminals for key enrollment and reconstruction is considered. A nested convolutional code construction that performs lossy compression with side information is proposed. Nested convolutional codes are an alternative to nested polar codes and nested random linear codes that achieve all points of the key-leakage-storage regions of the generated-secret and chosen-secret models for long block lengths. Our design uses a convolutional code for vector quantization during enrollment and a subcode of it for error correction during reconstruction. Physical identifiers with small bit error probability are considered to illustrate the gains of the proposed construction. One variant of nested convolutional codes improves on all previous constructions in terms of the key vs. storage rate ratio but it has high complexity. Another variant of nested convolutional codes with lower complexity performs similarly to previously designed nested polar codes. The results suggest that the choice of convolutional or polar codes for key agreement with identifiers depends on the complexity constraints.

CCS CONCEPTS

• **Security and privacy → Information-theoretic techniques**.

KEYWORDS

nested codes, information privacy, tailbiting, convolutional codes, physical unclonable functions

ACM Reference Format:
Thomas Jerkovits, Onur Günlü, Vladimir Sidorenko, and Gerhard Kramer. 2020. Nested Tailbiting Convolutional Codes for Secrecy, Privacy, and Storage. In *2020 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec'20), June 22–24, 2020, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3369412.3395063

1 INTRODUCTION

Irises and fingerprints are biometric identifiers used to authenticate and identify individuals, and to generate secret keys [4]. In a digital device, there are digital circuits that have outputs unique to the device. One can generate secret keys from such physical unclonable functions (PUFs) by using their outputs as a source of randomness. Fine variations of ring oscillator (RO) outputs, the start-up behavior of static random access memories (SRAM), and quantum-physical readouts through coherent scattering [36] can serve as PUFs that have reliable outputs and high entropy [11, 17]. One can consider them as physical "one-way functions" that are easy to compute and difficult to invert [32].

There are several security, privacy, storage, and complexity constraints that a PUF-based key agreement method should fulfill. First, the method should not leak information about the secret key (negligible *secrecy leakage*). Second, the method should leak as little information about the identifier (minimum *privacy leakage*). The privacy leakage constraint can be considered as an upper bound on the secrecy leakage via the public information of the first enrollment of a PUF about the secret key generated by the second enrollment of the same PUF [12]. Third, one should limit the *storage* rate because storage can be expensive and limited, e.g., for internet-of-things (IoT) device applications. Similarly, the hardware cost, e.g., hardware area, of the encoder and decoder used for key agreement with PUFs should be small for such applications.

There are two common models for key agreement: the *generated-secret (GS)* and the *chosen-secret (CS) models*. An encoder extracts a secret key from an identifier measurement for the GS model, while for the CS model a secret key that is independent of the identifier measurements is given to the encoder by a trusted entity. In the classic key-agreement model introduced in [1] and [30], two terminals observe correlated random variables and have access to a public, authenticated, and one-way communication link; an eavesdropper observes only the public messages called *helper data*. The regions of achievable secret-key vs. privacy-leakage (key-leakage) rates for the GS and CS models are given in [18, 25]. The storage rates for general (non-negligible) secrecy-leakage levels are analyzed in [22], while the rate regions with multiple encoder and decoder measurements of a hidden source are treated in [15]. There are other key-agreement models with an eavesdropper that has access to a sequence correlated with the identifier outputs, e.g., in [6, 8, 12, 21]. This model is not realistic for PUFs, unlike physical-layer security primitives and some biometric identifiers that are continuously available for physical attacks. PUFs are used for *on-demand* key reconstruction, i.e., the attack should be performed during execution, and an invasive attack applied to obtain a correlated sequence permanently changes the identifier output [11]. Therefore, we assume

IH&MMSec '20, June 22–24, 2020, Denver, CO, USA

ACM ISBN 978-1-4503-7050-9/20/06…$15.00
https://doi.org/10.1145/3369412.3395063

that the eavesdropper cannot obtain a sequence correlated with the PUF outputs.

Two classic code constructions for key agreement are code-offset fuzzy extractors (COFE) [10] and the fuzzy commitment scheme (FCS) [20], which are based on a one-time padding step in combination with an error correcting code. Both constructions require a storage rate of 1 bit/symbol due to the one-time padding step. A Slepian-Wolf (SW) [37] coding method, which corresponds to syndrome coding for binary sequences, is proposed in [5] to reduce the storage rate so that it is equal to the privacy-leakage rate. It is shown in [13] that these methods do not achieve the key-leakage-storage boundaries of the GS and CS models.

Wyner-Ziv (WZ) [41] coding constructions that bin the observed sequences are shown in [13] to be optimal deterministic code constructions for key agreement with PUFs. Nested random linear codes are shown to asymptotically achieve boundary points of the key-leakage-storage region. A second WZ-coding construction uses a nested version of polar codes (PCs) [3], which are designed in [13] for practical SRAM PUF parameters to illustrate that rate tuples that cannot be achieved by using previous code constructions can be achieved by nested PCs.

A closely related problem to the key agreement problem is Wyner's wiretap channel (WTC) [40]. The main aim in the WTC problem is to hide a transmitted message from the eavesdropper that observes a channel output correlated with the observation of a legitimate receiver. There are various code constructions for the WTC that achieve the secrecy capacity, e.g., in [2, 24, 27, 29], and some of these constructions use nested PCs, e.g., [2, 27]. Similarly, nested PCs are shown in [7] to achieve the strong coordination capacity boundaries, defined and characterized in [9].

We design codes for key agreement with PUFs by constructing nested convolutional codes. Due to the broad use of nested codes in, e.g., WTC and strong coordination problems, the proposed nested convolutional code constructions can be useful also for these problems. A summary of the main contributions is as follows.

- We propose a method to obtain nested tailbiting convolutional codes (TBCCs) that are used as a WZ-coding construction, which is a binning method used in various achievability schemes and can be useful for various practical code constructions.
- We develop a design procedure for the proposed nested convolutional code construction adapted to the problem of key agreement with biometric or physical identifiers. This is an extension of the asymptotically optimal nested code constructions with random linear codes and PCs proposed in [13]. We consider binary symmetric sources and binary symmetric channels (BSCs). Physical identifiers such as RO PUFs with transform coding [14] and SRAM PUFs [28] are modeled by these sources and channels.
- We design and simulate nested TBCCs for practical source and channel parameters obtained from the best PUF design in the literature. The target block-error probability is $P_B = 10^{-6}$ and the target secret-key size is 128 bits. We illustrate that one variant of nested codes achieves the largest key vs. storage rate ratio but it has high decoding complexity. Another variant of nested codes with lower decoding complexity achieves a rate ratio that is slightly greater than the rate ratio achieved by a nested PC. We also illustrate the gaps to the finite-length bounds.

This paper is organized as follows. In Section 3, we describe the GS and CS models, and give their rate regions that are also evaluated for binary symmetric sequences. We summarise in Section 4 our new nested code construction that uses convolutional codes. In Section 5, we propose a design procedure for the new nested TBCCs adapted to the key agreement with PUFs problem. Section 6 compares the estimated decoding complexity of TBCCs and PCs. Section 7 illustrates the significant gains from nested convolutional codes designed for practical PUF parameters as compared to previously-proposed nested PCs and other channel codes in terms of the key vs. storage rate ratio.

2 PRELIMINARIES

2.1 Notation

Let $\mathbb{F}_2$ denote the finite field of order 2 and let $\mathbb{F}_2^{a\times b}$ denote the set of all $a \times b$ matrices over $\mathbb{F}_2$. Rows and columns of $a \times b$ matrices are indexed by $1, \ldots, a$ and $1, \ldots, b$, and $h_{i,j}$ is the element in the i-th row and j-th column of a matrix $\mathbf{H}$. $\mathbb{F}_2^a$ denotes the set of all row vectors of length a over $\mathbb{F}_2$. With $\mathbf{0}_{a\times b}$ we denote the all-zero matrix of size $a \times b$. A linear block code over $\mathbb{F}_2$ of length N and dimension K is a K-dimensional subspace of $\mathbb{F}_2^N$ and denoted by (N, K). A variable with superscript denotes a string of variables, e.g., $X^n = X_1 \ldots X_i \ldots X_n$, and a subscript denotes the position of a variable in a string. A random variable X has probability distribution P_X. Calligraphic letters such as $\mathcal{X}$ denote sets, and set sizes are written as $|\mathcal{X}|$. $\mathrm{Enc}(\cdot)$ is an encoder mapping and $\mathrm{Dec}(\cdot)$ is a decoder mapping. $H_b(x) = -x \log x - (1-x)\log(1-x)$ is the binary entropy function, where we take logarithms to the base 2. The $*$-operator is defined as $p*x = p(1-x)+(1-p)x$. A BSC with crossover probability p is denoted by $\mathrm{BSC}(p)$. $X^n \sim \mathrm{Bern}^n(\alpha)$ is an independent and identically distributed (i.i.d.) binary sequence of random variables with $\Pr[X_i = 1] = \alpha$ for $i = 1, 2, \ldots, n$. $\mathbf{H}^T$ represents the transpose of the matrix $\mathbf{H}$. Drawing an element e from a set $\mathcal{E}$ uniformly at random is denoted by

$$e \xleftarrow{\$} \mathcal{E}. \tag{1}$$

2.2 Convolutional Codes

Denote the parameters of a block code generated by a binary convolutional encoder as (N, K), where N is the blocklength and K is the code dimension (in bits). At each time step, the convolutional encoder receives k input bits and generates n output bits. The number of clock cycles needed to encode K bits is $\ell = \frac{K}{k}$. We consider convolutional encoders with a single shift register only. The shift register consists of m delay cells, where m is also called the memory of the encoder. The bit value stored in the i-th delay cell at time step t is denoted by $s_t^{(i)} \in \mathbb{F}_2$ for $i = 1, \ldots, m$. For a given binary input vector $\mathbf{u}_t = \left(u_t^{(1)}, u_t^{(2)}, \ldots, u_t^{(k)}\right)$ of length k at time step t, the encoder outputs a binary vector $\mathbf{c}_t = \left(c_t^{(1)}, c_t^{(2)}, \ldots, c_t^{(n)}\right)$ of length n. The encoder can be described by the state-space representation

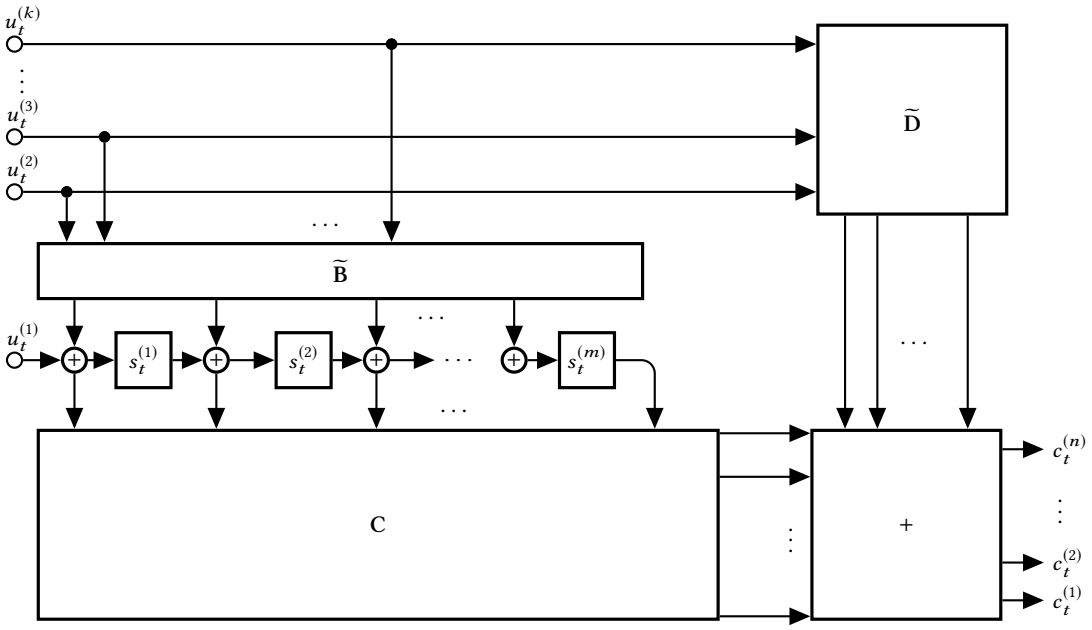

Figure 1: Encoder circuit of convolutional codes described in Section 2.2.

of the encoder circuit such that the output $\mathbf{c}_t$ is

$$\mathbf{c}_t = \mathbf{s}_t \cdot \mathbf{C}^T + \mathbf{u}_t \cdot \mathbf{D}^T \tag{2}$$

where $\mathbf{s}_t = \left(s_t^{(1)}, s_t^{(2)}, \ldots, s_t^{(m)}\right)$ is the vector describing the content of the shift register, $\mathbf{C} \in \mathbb{F}_2^{n\times m}$ is the observation matrix, and $\mathbf{D} \in \mathbb{F}_2^{n\times k}$ is the transition matrix. The content of the shift register for the next clock cycle at time step $t+1$ is then

$$\mathbf{s}_{t+1} = \mathbf{s}_t \cdot \mathbf{A}^T + \mathbf{u}_t \cdot \mathbf{B}^T \tag{3}$$

where $\mathbf{A} \in \mathbb{F}_2^{m\times m}$ is the system matrix and $\mathbf{B} \in \mathbb{F}_2^{m\times k}$ is the control matrix. For the case of a single shift register we have that the system matrix is given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{1\times(m-1)} & 0 \\ \mathbf{I}_{(m-1)\times(m-1)} & \mathbf{0}_{(m-1)\times 1} \end{bmatrix} \tag{4}$$

where $\mathbf{I}_{(m-1)\times(m-1)} \in \mathbb{F}_2^{(m-1)\times(m-1)}$ is the identity matrix. For simplicity, first entry of the input tuple $u_t^{(1)}$ is always an input to the shift register and thus we can write $\mathbf{B} = (\mathbf{e}_1^T|\widetilde{\mathbf{B}})$ and $\mathbf{D} = (\mathbf{0}_{n\times 1}|\widetilde{\mathbf{D}})$, where $\mathbf{e}_1$ is the unit row vector having a 1 in the first position and 0 everywhere else, $\widetilde{\mathbf{B}} \in \mathbb{F}_2^{m\times(k-1)}$, and $\widetilde{\mathbf{D}} \in \mathbb{F}_2^{n\times(k-1)}$. The corresponding encoder circuit is shown in Figure 1. Elements of a vector entering a square box, which represents one of the aforementioned matrices, depicts a vector-matrix multiplication, and the box with the addition symbol depicts an elementwise vector-vector addition. Therefore, the encoder of the convolutional code can be described by the three matrices $\widetilde{\mathbf{B}}$, $\mathbf{C}$, and $\widetilde{\mathbf{D}}$. We denote such an encoder by $[\widetilde{\mathbf{B}}, \mathbf{C}, \widetilde{\mathbf{D}}]$.

Using the tailbiting method from [19, Chapter 4.8], we avoid having a rate loss, unlike the zero-tail termination method. We have $N = \ell n$ and the resulting code rate is $R = \frac{k}{n}$. A *tailbiting convolutional code (TBCC)* can be represented by a tailbiting trellis using ℓ sections and 2^m states per section. The codewords correspond to all possible paths in the trellis, where starting and ending states coincide. TBCCs can be decoded by using the wrap around Viterbi algorithm (WAVA) [35]. This decoder is suboptimal but performs close to the performance of the maximum likelihood decoder.

Let A_d be the number of codewords of Hamming weight d for $d = 0, 1, \ldots, N$, which characterizes the distance spectrum of a TBCC. The weight enumerator polynomial $A(X)$ is then defined as

$$A(X) \stackrel{\text{def}}{=} \sum_{d=0}^{N} A_d X^d. \tag{5}$$

To compute the weight enumerator and to determine the distance spectrum we use the approach described in [39]. Consider the state transition matrix $\mathbf{T}(X)$ of size $2^m \times 2^m$, where every entry $t_{i,j}(X)$ is either X^d, where d is the Hamming weight of the output produced by the encoder when going from the state labeled with i to the state labeled with j, or 0 if there is no possible transition between the aforementioned states. Therefore, we have

$$A(X) = \mathrm{Tr}\left(\mathbf{T}^\ell(X)\right) \tag{6}$$

where $\mathbf{T}^\ell(X)$ denotes multiplication of the matrix $\mathbf{T}(X)$ with itself ℓ times and $\mathrm{Tr}(\cdot)$ denotes the trace.

3 PROBLEM FORMULATION

Consider the GS model in Figure 2(a), where a biometric or physical source output is used to generate a secret key. The source $\mathcal{X}$, noisy measurement $\mathcal{Y}$, secret key $\mathcal{S}$, and storage $\mathcal{W}$ alphabets are finite sets. During enrollment, the encoder observes the i.i.d. identifier output X^N, generated according to some P_X, and computes a secret key $S \in \mathcal{S}$ and public helper data $W \in \mathcal{W}$ as $(S, W) = \mathrm{Enc}(X^N)$. During reconstruction, the decoder observes a noisy source measurement

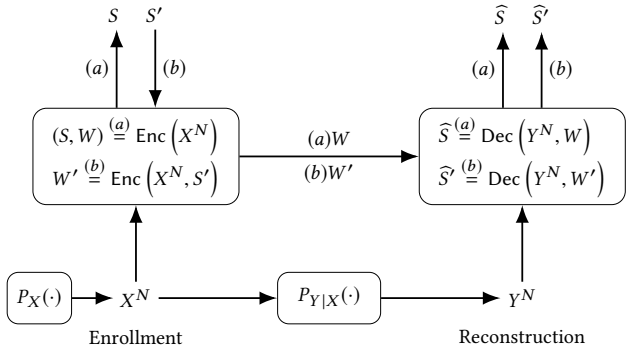

Figure 2: The (a) GS and (b) CS models.

Y^N of the source output X^N through a memoryless measurement channel $P_{Y|X}$ in addition to the helper data W. The decoder estimates the secret key as $\widehat{S} = \text{Dec}(Y^N, W)$. Furthermore, Figure 2$(b)$ shows the CS model, where a secret key $S' \in \mathcal{S}$ is embedded into the helper data as $W' = \text{Enc}(X^N, S')$. The decoder for the CS model estimates the secret key as $\widehat{S}' = \text{Dec}(Y^N, W')$.

Definition 3.1. A key-leakage-storage tuple (R_s, R_ℓ, R_w) is *achievable* for the GS and CS models if, given any $\epsilon > 0$, there is some $N \geq 1$, an encoder, and a decoder such that $R_s = \frac{\log|\mathcal{S}|}{N}$ and

$$P_{\text{B}} \stackrel{\text{def}}{=} \Pr[\widehat{S} \neq S] \leq \epsilon \qquad \text{(reliability)} \tag{7}$$

$$\frac{1}{N} I(S; W) \leq \epsilon \qquad \text{(secrecy)} \tag{8}$$

$$\frac{1}{N} H(S) \geq R_s - \epsilon \qquad \text{(key uniformity)} \tag{9}$$

$$\frac{1}{N} \log \left|\mathcal{W}\right| \leq R_w + \epsilon \qquad \text{(storage)} \tag{10}$$

$$\frac{1}{N} I(X^N; W) \leq R_\ell + \epsilon \qquad \text{(privacy)} \tag{11}$$

where for the CS model, S and W in the constraints should be replaced by, respectively, S' and W'.

The *key-leakage-storage* regions $\mathcal{R}_{\text{gs}}$ and $\mathcal{R}_{\text{cs}}$ for the GS and CS models, respectively, are the closures of the sets of achievable tuples for the corresponding models. ♢

Theorem 3.2 ([18]). *The key-leakage-storage region $\mathcal{R}_{gs}$ for the GS model is the union of the bounds*

$$0 \leq R_s \leq I(U; Y) \tag{12}$$

$$R_\ell \geq I(U; X) - I(U; Y) \tag{13}$$

$$R_w \geq I(U; X) - I(U; Y) \tag{14}$$

over all $P_{U|X}$ such that $U - X - Y$ form a Markov chain. Similarly, the key-leakage-storage region $\mathcal{R}_{cs}$ for the CS model is the union of the bounds in (12), (13), and

$$R_w \geq I(U; X). \tag{15}$$

These regions are convex sets. The alphabet $\mathcal{U}$ of the auxiliary random variable U can be limited to have size $|\mathcal{U}| \leq |\mathcal{X}| + 1$. Deterministic encoders and decoders suffice to achieve these regions.

Suppose the transform-coding algorithms proposed in [14] are applied to RO PUFs or any PUF circuits with continuous-valued outputs to obtain X^N that is almost i.i.d. according to a uniform Bernoulli random variable, i.e., $X^N \sim \text{Bern}^N(\frac{1}{2})$, and the channel $P_{Y|X}$ is a BSC(p_A) for $p_A \in [0, 0.5]$. The key-leakage-storage region $\mathcal{R}_{\text{gs,bin}}$ of the GS model for this case is the union of the bounds

$$\begin{aligned} 0 &\leq R_s \leq 1 - H_b(q * p_A) \\ R_\ell &\geq H_b(q * p_A) - H_b(q) \\ R_w &\geq H_b(q * p_A) - H_b(q) \end{aligned} \tag{16}$$

over all $q \in [0, 0.5]$ [18], which follows by using an auxiliary random variable U such that $P_{X|U} \sim \text{BSC}(q)$ due to Mrs. Gerber's lemma [41]. The rate tuples on the boundary of the region $\mathcal{R}_{\text{gs,bin}}$ are uniquely defined by the ratio $\frac{R_s}{R_w}$. We therefore use this ratio as the metric to compare our nested TBCCs with previously-proposed nested PCs and channel codes. A larger key vs. storage rate ratio suggests that the code construction is closer to an achievable point that is on the boundary of the region $\mathcal{R}_{\text{gs,bin}}$, which is an optimal tuple. We next focus on the GS model for code constructions. All results can be extended to the CS model by using an additional one-time padding step [12].

4 NESTED CONVOLUTIONAL CODE CONSTRUCTION

In this section, we sketch the main steps to obtain a nested construction for convolutional codes. Furthermore, we give two explicit algorithms to find good code constructions. The first algorithm addresses the search of a good error correcting code (N, K_{s}), denoted by $\mathcal{C}_{\text{s}}$, and the second algorithm finds a (N, K_{q}) code $\mathcal{C}_{\text{q}}$ used as a vector quantizer such that $\mathcal{C}_{\text{s}}$ is a subcode of $\mathcal{C}_{\text{q}}$, i.e., $\mathcal{C}_{\text{s}} \subseteq \mathcal{C}_{\text{q}}$.

4.1 Nested Convolutional Codes

Using the encoder circuit depicted in Figure 1, we construct two codes $\mathcal{C}_{\text{q}}$ and $\mathcal{C}_{\text{s}}$ such that $\mathcal{C}_{\text{s}} \subseteq \mathcal{C}_{\text{q}}$. Let $\mathcal{C}_{\text{q}}$ be the (N, K_{q}) TBCC with memory m and $K_{\text{q}} = \ell k_{\text{q}}$ generated by using the encoder defined by the matrices $[\widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}, \widetilde{\mathbf{D}}]$. Recall that $\mathbf{B} = (\mathbf{e}_1^T | \widetilde{\mathbf{B}})$ with $\widetilde{\mathbf{B}} \in \mathbb{F}_2^{m \times (k_{\text{q}}-1)}$ and $\mathbf{D} = (\mathbf{0}^T | \widetilde{\mathbf{D}})$ with $\widetilde{\mathbf{D}} \in \mathbb{F}_2^{n \times (k_{\text{q}}-1)}$. By removing the i-th column of $\widetilde{\mathbf{B}}$ and $\widetilde{\mathbf{D}}$ simultaneously, one obtains a new encoder that generates a code of rate $\frac{k_{\text{q}} - 1}{n}$, which is a subcode of the original code. This is true, since the new code corresponds to all codewords by encoding the original code but restricting to all inputs where $u_t^{(i)} = 0$. By "freezing" further input bits we can therefore obtain a subcode of rates

$$R_{\text{s}} = \frac{1}{n}, \frac{2}{n}, \ldots, \frac{k_{\text{q}} - 1}{n}. \tag{17}$$

To obtain codes with rates of better granularity between $\frac{1}{n}$ and $\frac{k_{\text{q}} - 1}{n}$ that are not in (17), we can freeze input bits in a time-variant manner. That is, by using the encoder ℓ times, we can freeze a different amount of input bits in different clock cycles. This allows to obtain codes of rates

$$R_{\text{s}} = \frac{\ell}{N}, \frac{\ell + 1}{N}, \ldots, \frac{K_{\text{q}}}{N}. \tag{18}$$

Algorithm 1: Search for (N, K_s) TBCC C_s, $R_\mathrm{s} = \frac{1}{n}$

Input : n, m, K_s, P_B, W_max (maximum number of iterations)
Output: $\mathbf{C} \in \mathbb{F}_2^{n\times m}$

Initialize:
$p_\mathrm{c} \leftarrow 0$
$\mathbf{C} \leftarrow \mathbf{0}$
for $w \leftarrow 1$ **to** W_{max} **do**
  $\mathbf{C}' \xleftarrow{\$} \mathbb{F}_2^{n\times m}$
  Compute A_d for the (N, K_s) TBCC generated by $[\mathbf{0}, \mathbf{C}', \mathbf{0}]$ for $d = 0, \ldots, N$ using (5) and (6)
  Find p_c' such that: $P_B^{\mathrm{UB}}(A_d, p_\mathrm{c}') = P_\mathrm{B}$
  if $p_c' \geq p_c$ **then**
    $p_\mathrm{c} \leftarrow p_\mathrm{c}'$
    $\mathbf{C} \leftarrow \mathbf{C}'$
return $\mathbf{C}$

Denote the parameters of the subcode, obtained by freezing input bits accordingly, as (N, K_s). Note that by freezing input bits in a time-variant manner, K_s is not necessarily a multiple of ℓ. Furthermore, the procedure can be applied also to add columns to $\widetilde{\mathbf{B}}$ and $\widetilde{\mathbf{D}}$ to generate a supercode. The design procedure of the nested convolutional code construction is split into two steps:

(1) Search for a good error correcting code C_s of rate $R_\mathrm{s} = \frac{1}{n} = \frac{K_\mathrm{s}}{N}$ at given target block error probability P_B by finding an appropriate matrix $\mathbf{C}$.
(2) Expand the low rate code by finding appropriate matrices $\widetilde{\mathbf{B}}$ and $\widetilde{\mathbf{D}}$ to obtain a good code of rate $R_\mathrm{q} = \frac{k_\mathrm{q}}{n} = \frac{K_\mathrm{q}}{N}$ that achieves a low average distortion q.

Note that for the first step we restrict to codes of rate $R_\mathrm{s} = \frac{1}{n}$ and hence the matrices $\widetilde{\mathbf{B}}$ and $\widetilde{\mathbf{D}}$ are vanishing. The first step can also be performed for codes of any rate $R_\mathrm{s} > \frac{1}{n}$, but then also the appropriate matrices $\widetilde{\mathbf{B}}$ and $\widetilde{\mathbf{D}}$ have to be found accordingly.

4.2 Design of a Convolutional Code for Error Correction

For fixed parameters n, m, and K_s, we try to find a matrix $\mathbf{C}$ such that the resulting (N, K_s) TBCC C_s at a given target block error probability P_B can be operated on a noisy BSC with large crossover probability p_c. To evaluate P_B we use the union bound, see, e.g., [33], and the distance spectrum of the code. This gives an upper bound on P_B under maximum likelihood decoding. The bound is given by

$$P_\mathrm{B} \leq P_B^{\mathrm{UB}}(A_d, p_\mathrm{c}) \stackrel{\mathrm{def}}{=} \sum_{d=d_\mathrm{min}}^{N} A_d \sum_{i=\lceil d/2 \rceil}^{d} \binom{d}{i} p_\mathrm{c}^i (1-p_\mathrm{c})^{d-i} \tag{19}$$

where d_min is the minimum distance of the code.

Algorithm 2: Search for (N, K_q) TBCC C_q, $R_\mathrm{q} = \frac{k_\mathrm{q}}{n}$

Input : m, k_q, k_s, W_max, $\mathbf{C}$, $\widetilde{\mathbf{B}}_\mathrm{s} \in \mathbb{F}_2^{m\times(k_\mathrm{s}-1)}$, $\widetilde{\mathbf{D}}_\mathrm{s} \in \mathbb{F}_2^{n\times(k_\mathrm{s}-1)}$
Output: $\widetilde{\mathbf{B}}_\mathrm{q} \in \mathbb{F}_2^{m\times(k_\mathrm{q}-1)}$, $\widetilde{\mathbf{D}}_\mathrm{q} \in \mathbb{F}_2^{n\times(k_\mathrm{q}-1)}$

Initialize:
$\widetilde{\mathbf{B}}_\mathrm{q} \leftarrow (\widetilde{\mathbf{B}}_\mathrm{s}|\mathbf{0})$
$\widetilde{\mathbf{D}}_\mathrm{q} \leftarrow (\widetilde{\mathbf{D}}_\mathrm{s}|\mathbf{0})$
$d \leftarrow 0$
$A \leftarrow 0$
for $w \leftarrow 1$ **to** W_{max} **do**
  $\mathbf{B}' \xleftarrow{\$} \mathbb{F}_2^{m\times(k_\mathrm{q}-k_\mathrm{s})}$
  $\mathbf{D}' \xleftarrow{\$} \mathbb{F}_2^{n\times(k_\mathrm{q}-k_\mathrm{s})}$
  $\widetilde{\mathbf{B}}_\mathrm{q}' \leftarrow (\widetilde{\mathbf{B}}_\mathrm{s}|\mathbf{B}')$
  $\widetilde{\mathbf{D}}_\mathrm{q}' \leftarrow (\widetilde{\mathbf{D}}_\mathrm{s}|\mathbf{D}')$
  Compute d_free and A_free for $[\widetilde{\mathbf{B}}_\mathrm{q}, \mathbf{C}, \widetilde{\mathbf{D}}_\mathrm{q}]$
  if $d_{free} > d$ *or* $(d_{free} = d$ *and* $A_{free} < A)$ **then**
    $d \leftarrow d_\mathrm{free}$
    $A \leftarrow A_\mathrm{free}$
    $\widetilde{\mathbf{B}}_\mathrm{q} \leftarrow \widetilde{\mathbf{B}}_\mathrm{q}'$
    $\widetilde{\mathbf{D}}_\mathrm{q} \leftarrow \widetilde{\mathbf{D}}_\mathrm{q}'$
return $\widetilde{\mathbf{B}}_q$, $\widetilde{\mathbf{D}}_q$

The design of the code C_s is performed by a purely random search of the matrix $\mathbf{C}$ as described in Algorithm 1. This algorithm searches the best TBCC of rate $R_\mathrm{s} = \frac{1}{n}$ by randomly generating different matrices $\mathbf{C}$. The matrix $\mathbf{C}$ of the code that yields the largest p_c at a given target block error probability P_B is returned as the output of Algorithm 1.

4.3 Design of a Convolutional Code for Vector Quantization

In this section, an algorithm to obtain a high rate code from an existing low rate convolutional encoder is explained. The algorithm is presented in Algorithm 2. The inputs are the system matrix, the observation matrix, and the transition matrix of the low rate code with rate

$$R_\mathrm{s} = \frac{k_\mathrm{s}}{n}. \tag{20}$$

By randomly adding $k_\mathrm{q} - k_\mathrm{s}$ columns to both, the system and the transition matrix of a code of high rate

$$R_\mathrm{q} = \frac{k_\mathrm{q}}{n} \tag{21}$$

is constructed. The algorithm performs a random search and returns the best configuration. As selection metrics, the free distance and its multiplicity are chosen. The free distance d_free of a convolutional code is defined as the minimum Hamming weight between any two differing paths in the state transition diagram [19, Chapter 3]. Due to linearity of convolutional codes, d_free is also the minimum Hamming weight over the nonzero paths. We denote by A_free the

multiplicity of paths that have Hamming weight d_{free}. To find a good high rate code, we use d_{free} and A_{free} to select the best encoder. The BEAST algorithm described in [19, Chapter 10] is a fast method to compute d_{free} and A_{free}. The selection criterion is as follows: Keep the code with largest d_{free} and in case of a tie decide for the code with smaller A_{free}.

5 DESIGN OF NESTED CONVOLUTIONAL CODES FOR PUFS

Algorithms 1 and 2 are combined to find good nested code constructions for the coding problem described in Section 3. Two TBCCs C_{s} and C_{q} of the same length N are needed such that $C_{\text{s}} \subseteq C_{\text{q}}$. Let K_{q} and K_{s} denote the dimensions of C_{q} and C_{s}, respectively, and let $R_{\text{q}} = \frac{K_{\text{q}}}{N}$ and $R_{\text{s}} = \frac{K_{\text{s}}}{N}$ denote their code rates. The objective is to maximize the key vs. storage rate ratio. Since $R_s = \frac{K_{\text{s}}}{N}$ and $R_w = \frac{K_{\text{q}} - K_{\text{s}}}{N}$, we have

$$\frac{R_s}{R_w} = \frac{K_{\text{s}}}{K_{\text{q}} - K_{\text{s}}} = \left(\frac{R_{\text{q}}}{R_{\text{s}}} - 1\right)^{-1}. \tag{22}$$

Therefore, we maximize R_{s} and minimize R_{q} simultaneously.

To reconstruct the key S of size K_{s} (in bits) the code C_{s} has to correct errors on the artifical BSC channel with crossover probability $p_{\text{c}} = q * p_{\text{A}}$ at a given target P_{B}. The code C_{q} serves as a vector quantizer with average distortion q such that [13]

$$q \leq \frac{p_{\text{c}} - p_{\text{A}}}{1 - 2p_{\text{A}}}. \tag{23}$$

The design procedure is then as follows:

(1) Choose m and n to design a TBCC of rate $R_{\text{s}} = \frac{1}{n}$ by using Algorithm 1.
(2) Obtain the corresponding value of p_{c} where the code achieves the target block error probability P_{B} by Monte Carlo simulations.
(3) Construct code C_{q} from C_{s} by using Algorithm 2 such that (23) is satisfied.

The last step in this procedure is executed by applying Algorithm 2 incrementally as follows:

(1) Initialization: Start constructing a code $C_{\text{q}}^{(0)}$ of rate $R_{\text{q}}^{(0)} = \frac{2}{n}$ from code C_{s} (Algorithm 1).
(2) Set $i \leftarrow 1$.
(3) Construct a code $C_{\text{q}}^{(i)}$ of rate $R_{\text{q}}^{(i)} = \frac{i+2}{n}$ from code $C_{\text{q}}^{(i-1)}$ (Algorithm 2).
(4) If the average distortion achieved by the code $C_{\text{q}}^{(i)}$ satisfies the constraint given in (23), stop; else increment $i \leftarrow i + 1$ and go to step (3).

The final code C_{q} is the code in the last iteration. To obtain code rates in between those steps we randomly freeze inputs of the encoder in a time-variant manner as described in Section 4. Since in each iteration the code is optimized for the minimum distance of the code, we can only freeze inputs on the last added input. This way we guarantee to preserve the minimum distance of the code for the next iteration due to $C_{\text{q}}^{(i-1)} \subseteq C_{\text{q}}^{(i)}$.

6 ESTIMATED DECODING COMPLEXITY

We compare the decoding complexities of TBCCs and PCs. Since the real complexity of decoding depends on the hardware implementation, we only estimate the complexity for both code classes by using standard decoding algorithms.

The WAVA algorithm performs standard Viterbi decoding on the tailbiting trellis of the TBCC in a circular fashion. That means the decoder runs over the trellis several times and at each iteration the probabilities of the starting states of the trellis are updated according to the probabilities of the ending states of the previous iteration. Therefore, the WAVA algorithms scales with the complexity of a standard Viterbi decoder times the number of iterations. For simplicity, we consider the worst case complexity and hence let V denote the number of maximum iterations of the WAVA decoder.

According to [26], let κ be the complexity of a standard Viterbi decoder with indices

- F for Forney trellis,
- P for precomputation,
- M for merged or minimal trellis.

We have for the total of number $\frac{N}{n}$ of trellis sections

$$\kappa_F \propto N \cdot 2^{k+m} \tag{24}$$

$$\kappa_P \propto \frac{N}{n}\left(2^{k+m} + 2^n\right) \tag{25}$$

$$\kappa_M \propto N \cdot 2^{\min\{k, n-k\}+m}. \tag{26}$$

By scaling these complexities with the maximum number of WAVA iterations V we obtain the desired complexities of decoding a TBCC. For decoding on the Forney trellis, we can reuse the branch metrics computed in the first WAVA iteration in the following $V - 1$ iterations and; therefore, we obtain

$$\kappa_F^{\text{WAVA}} \propto (n + V - 1)\frac{N}{n}2^{k+m} \tag{27}$$

$$\kappa_P^{\text{WAVA}} \propto \frac{N}{n}\left(V \cdot 2^{k+m} + 2^n\right) \tag{28}$$

$$\kappa_M^{\text{WAVA}} \propto VN \cdot 2^{\min\{k, n-k\}+m}. \tag{29}$$

Overall we have that the complexity κ^{WAVA} of decoding a TBCC is

$$\kappa^{\text{WAVA}} \propto \min\left\{\kappa_F^{\text{WAVA}}, \kappa_P^{\text{WAVA}}, \kappa_M^{\text{WAVA}}\right\}. \tag{30}$$

For error correction and vector quantization, we obtain different complexities since we have different values for k. For the error correcting code we have $k = k_{\text{s}} = 1$ and for the vector quantizer code we have $k = k_{\text{q}}$, where k_{q} is the largest value needed to achieve a rate of R_{q} such that $k_{\text{q}} = \lceil nR_{\text{q}} \rceil$. The complexity of the vector quantizer can be reduced by considering decoding over the trellis with the time-variant frozen input bit values, since all branches that do not correspond to the frozen input bit value can be removed. For simplicity, we will only consider the complexity over the time-invariant trellis.

For the PCs under successive cancellation list (SCL) decoding [38] with a list size L, we have a complexity proportional to $LN \log_2 N$. This complexity is independent of the code rate and thus applies to C_{s} and C_{q}. All decoding complexities are summarized in Table 1.

Note that for the Viterbi decoder parallelization up to a factor of 2^m can be easily achieved since all state nodes in a trellis section

Table 1: Complexities of the error correcting code C_s and vector quantizer code C_q for PCs and TBCCs.

Code class	Complexity of C_s	Complexity of C_q
TBCC κ_F^{WAVA}	$\propto (n+V-1)\frac{N}{n}2^m$	$\propto (n+V-1)\frac{N}{n}2^{k_q+m}$
TBCC κ_P^{WAVA}	$\propto \frac{N}{n}(V\cdot 2^m+2^n)$	$\propto \frac{N}{n}\left(V\cdot 2^{k_q+m}+2^n\right)$
TBCC κ_M^{WAVA}	$\propto VN\cdot 2^{1+m}$	$\propto VN\cdot 2^{\min\{k_q,n-k_q\}+m}$
PC	$\propto LN\log_2 N$	$\propto LN\log_2 N$

can be processed independently. For the SCL decoding of PCs, parallelization cannot be achieved without changing the decoder's error correction performance since each decoded bit sequentially depends on the previously decoded ones.

7 PERFORMANCE EVALUATIONS FOR PUFS

In this section, the performance of TBCCs designed by the proposed procedure for the PUF setting is presented. We consider PUF devices with $p_A = 0.0149$, target block error probability $P_B = 10^{-6}$ and a key size of $K_s = 128$ bits. These values correspond to the best RO PUF designs in the literature [16]. We construct TBCCs with rates $R_s = \frac{1}{3}$ and $R_s = \frac{1}{4}$, and with memories $m = 8$ and $m = 11$. As a reference, we also give a PC construction using the approach described in [13]. Without puncturing we can only provide a PC construction for the case of $R_s = \frac{1}{4}$, since for a key size of $K_s = 128$ and code rate $R_s = \frac{1}{3}$ we would have $N = 384$ which is not a power of two. All simulations for the PCs are performed by using SCL decoding with a list size of $L = 8$. We also compute the results for the rate $\frac{1}{8}$ PC presented in [13] but now for $p_A = 0.0149$ and give the resulting key vs. storage rate ratio $\frac{R_s}{R_w}$. All simulations for the TBCCs are performed by using the WAVA algorithm with a maximum of $V = 4$ iterations. The final results of all discussed codes are given in Table 2.

7.1 Error Correction Performance

The construction of the nested code design starts with the error correcting code C_s. We design two TBCCs with $R_s = \frac{1}{3}$ and $R_s = \frac{1}{4}$ by using Algorithm 1 with $W_{\max} = 10^4$. Results of the Monte Carlo simulations as well as the bound (19) are shown in Figures 3 and 4 for the two TBCCs.

To bound the code performance on a BSC for a given block length and code rate we use two finite length bounds, namely the *meta converse (MC)* and the *random coding union (RCU)* bound from [34]. The MC gives a lower bound and the RCU an upper bound on the block error probability. For $R_s = \frac{1}{4}$, we observe that the TBCC with $m = 11$ outperforms the PC, whereas the TBCC with $m = 8$ performs worse. We also observe that for all considered codes there is still a gap to the finite length bounds.

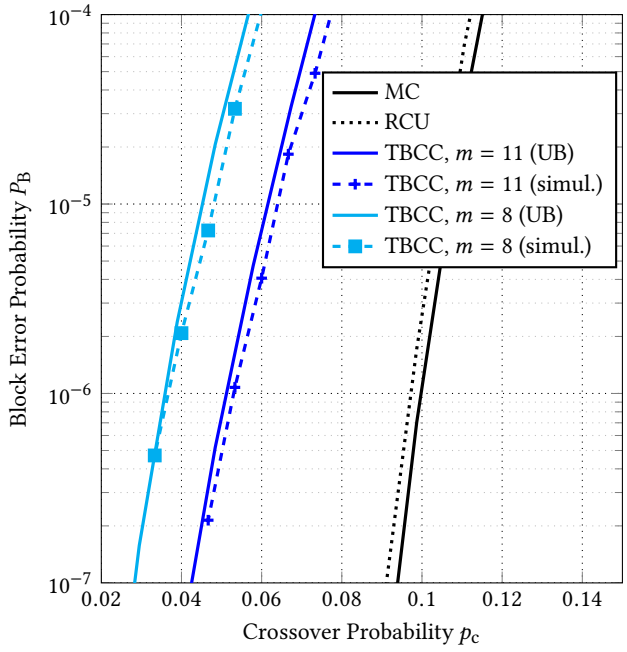

Figure 3: Error correcting performance of different codes with $K_s = 128$ bits and $R_s = \frac{1}{3}$ over a BSC with crossover probability p_c. The MC and RCU bounds for the same code parameters are given as references.

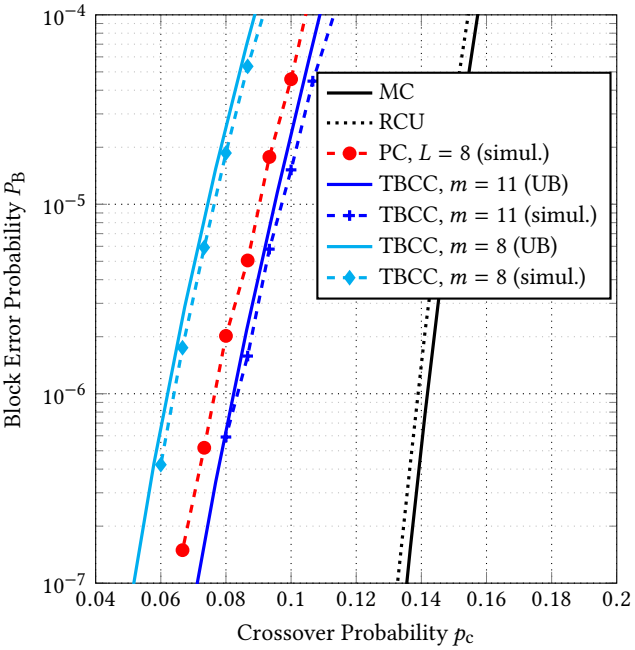

Figure 4: Error correcting performance of different codes with $K_s = 128$ bits and $R_s = \frac{1}{4}$ over a BSC with crossover probability p_c. The MC and RCU bounds for the same code parameters are given as references.

Table 2: Parameters of the designed codes for $K_s = 128$ bits, $p_A = 0.0149$ and $P_B \leq 10^{-6}$ and complexities for C_s and C_q, respectively. For the TBCCs also the type of complexity (κ_F^{WAVA},κ_M^{WAVA} or κ_P^{WAVA}) which is minimal is given. $\lceil \log_2 |\mathcal{W}| \rceil$ is the amount of helper data in bits.

Code	m	R_s	p_c	$\bar{q}$	R_q	R_w	$\lceil \log_2 \lvert\mathcal{W}\rvert \rceil$	$\frac{R_s}{R_w}$	Complexity C_s	Complexity C_q
+ TBCC	11	$\frac{1}{3}$	0.0545	0.0408	0.8047	0.4714	181	0.7072	$\kappa_P^{WAVA} \propto 2^{21.00}$	$\kappa_M^{WAVA} \propto 2^{21.58}$
■ TBCC	8	$\frac{1}{3}$	0.0365	0.0223	0.8906	0.5573	214	0.5981	$\kappa_P^{WAVA} \propto 2^{18.00}$	$\kappa_M^{WAVA} \propto 2^{18.58}$
◆ TBCC	11	$\frac{1}{4}$	0.0837	0.0709	0.6680	0.4180	214	0.5981	$\kappa_P^{WAVA} \propto 2^{21.00}$	$\kappa_M^{WAVA} \propto 2^{23.00}$
▲ TBCC	8	$\frac{1}{4}$	0.0640	0.0507	0.7441	0.4941	253	0.5059	$\kappa_P^{WAVA} \propto 2^{18.01}$	$\kappa_M^{WAVA} \propto 2^{20.00}$
✱ PC	-	$\frac{1}{4}$	0.0778	0.0648	0.6875	0.4375	224	0.5714	$\propto 2^{15.17}$	$\propto 2^{15.17}$
● PC	-	$\frac{1}{8}$	0.1819	0.1721	0.3584	0.2333	239	0.5358	$\propto 2^{16.32}$	$\propto 2^{16.32}$

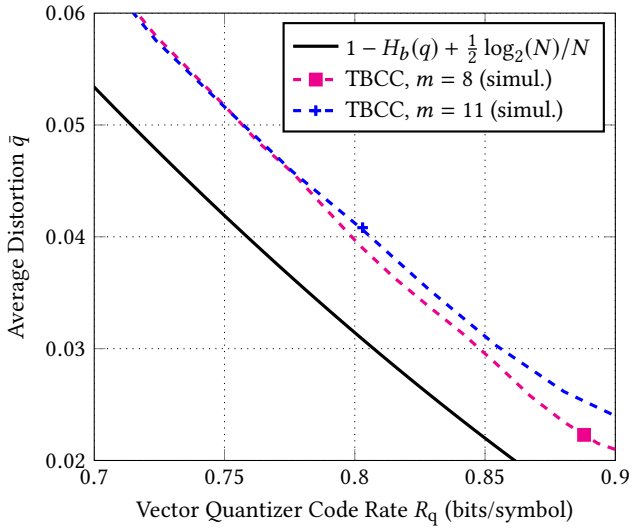

Figure 5: Code rate of the vector quantizer code C_q vs. average distortion $\bar{q}$ for $N = 384$ bits and $P_B \leq 10^{-6}$.

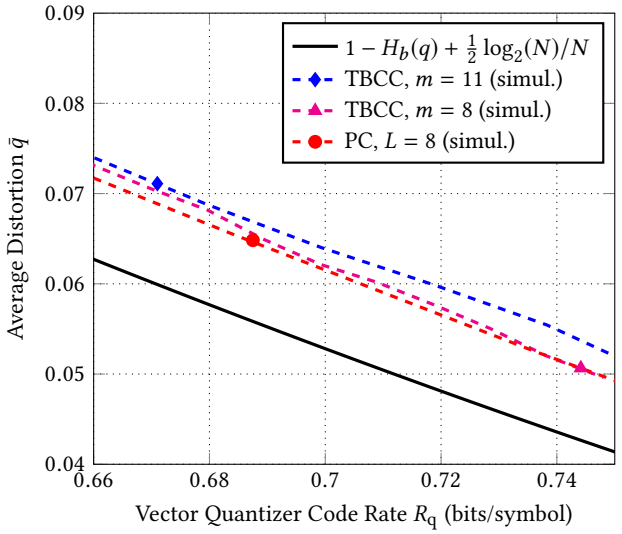

Figure 6: Code rate of the vector quantizer code C_q vs. average distortion $\bar{q}$ for $N = 512$ bits and $P_B \leq 10^{-6}$.

7.2 Vector Quantization Performance

Using the approach described in Section 4 and setting $W_{max} = 10^4$, we construct high rate codes to be used as a vector quantizer. Using Monte Carlo simulations, we plot the rate of these codes R_q vs. the measured average distortion $\bar{q}$ in Figures 5 and 6 for $N = 384$, corresponding to $R_s = \frac{1}{3}$, and for $N = 512$, corresponding to $R_s = \frac{1}{4}$, respectively.

We plot the approximate bound on the rate achieved for a given distortion from [23]. The approximated rate for block length N is

$$R_q^{(approx)} \stackrel{def}{=} 1 - H_b(q) + \frac{\log_2(N)}{2N} + O\left(\frac{1}{N}\right) \tag{31}$$

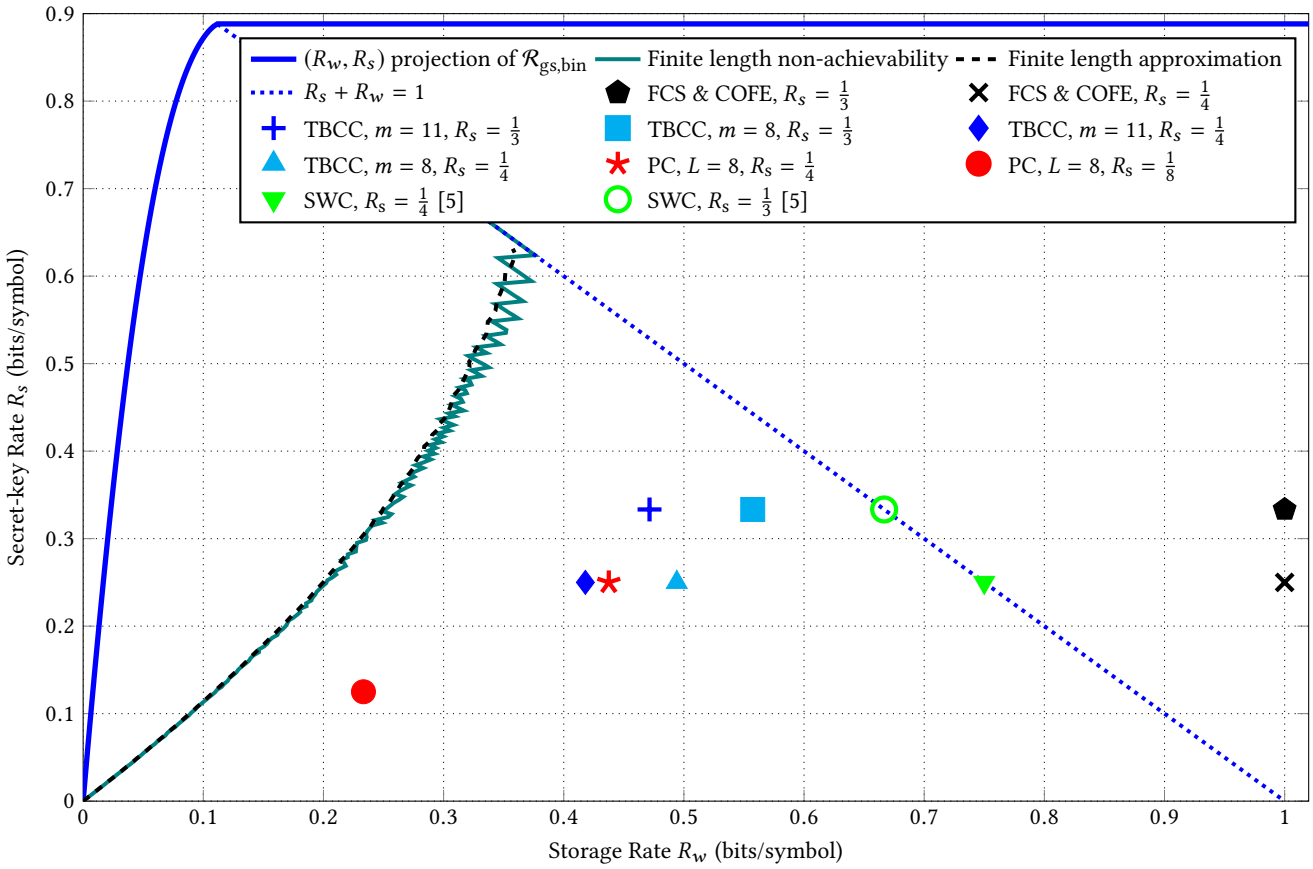

Figure 7: Storage-key rates for the GS model with $p_A = 0.0149$. The $(0.1118, 0.8882)$ bits/symbol point is the best possible point achieved by SW-coding (SWC) constructions such as polar codes (PCs) in [5], which lies on the dashed line representing $R_w + R_s = H(X)$. The PCs are designed by applying the design procedure proposed in [13] for WZ-coding with the SCL decoder with list size of L. The block-error probability satisfies $P_B \leq 10^{-6}$ and $K_s = 128$ bits for all codes. The finite length non-achievability bound and its approximation for $K_s = 128$ bits is depicted as well.

where $O(\cdot)$ denotes the big O notation. This approximation does not consider the effect of the constraint that the error correcting code designed in the previous subsection has to be a subcode of the vector quantizer of rate R_q. Therefore, this bound only gives an approximate achievable bound on the rate of the high-rate code that is used as a vector quantizer without having any constraint. The bound is plotted by neglecting the $O\left(\frac{1}{N}\right)$ term.

Using (23), we obtain the target distortion for the code to be designed, which allows to find a lower bound on the required rate R_q of the vector quantizer. The results are shown in Table 2. Observe that vector quantization performance of all codes is similar. Therefore, the code that has the best error correction performance yields the smallest rate for vector quantization, which corresponds to the smallest amount of helper data.

7.3 Overall Performance

Combining the results of the error correction and the vector quantizer performance, we can evaluate the key vs. storage rate ratio by using (22). The intermediate and final results are listed in Table 2, and the achieved (R_w, R_s) tuples for all mentioned codes are depicted in Figure 7.

For the nested WZ-coding construction, where we have a vector quantizer and an error correcting code, we plot a finite length non-achievability (converse) bound. For a fixed key size of $K_s = 128$ bits and $p_A = 0.0149$, we evaluate the MC non-achievability bound for the error correcting code and combine this bound using (23) with the non-achievability bound from [23, (2.186)] for the vector quantizer code. A slightly tighter version of the non-achievability bound for the vector quantizer code can be found in [31]. To achieve a distortion of q, any vector quantizer code of blocklength N must satisfy [23, (2.186)]

$$\sum_{j=0}^{\lfloor Nq \rfloor} \binom{N}{j} \geq 2^{N(1-R_q)}. \tag{32}$$

Similar to the achievability bound discussed in Section 7.2, (31) is used to approximate also the non-achievability bound in (32). The combination of the MC bound and the converse bound for the vector

quantizer performance establishes a non-achievability bound on the best rate tuples that can be achieved for given parameters by our WZ-coding construction. In Figure 7, we plot this non-achievability bound using (32) and its approximation using (31). Note that the zigzag behaviour of the bound in (32) is due to the floor function. We observe a gap between these bounds and achieved rate tuples by the designed codes.

As discussed above, FCS and COFE have the key vs. storage ratio of

$$\frac{R_s}{R_w} = R_s \tag{33}$$

as the storage rate is 1 bit/symbol for these constructions. The SW coding constructions such as the syndrome coding method proposed in [5] achieve the ratio

$$\frac{R_s}{R_w} = \frac{R_s}{1 - R_s} \tag{34}$$

which improves on the FCS and COFE. WZ coding constructions with nested PC we constructed for $p_A = 0.0149$ based on the design procedure given in [13] achieves even larger ratios. The largest key vs. storage rate ratio is achieved by the TBCC with $R_s = \frac{1}{3}$ and $m = 11$ such that $\frac{R_s}{R_w} = 0.7072$. These results suggest that increasing the code rate R_s and the memory size of TBCCs allows a larger key vs. storage rate ratio.

8 CONCLUSION

We proposed a nested convolutional code construction, which might be useful for various achievability schemes. For the key agreement problem with PUFs, we proposed a design procedure for the nested code construction using TBCCs to obtain good reliability, secrecy, privacy, storage, and cost performance jointly. We implemented nested convolutional codes for practical source and channel parameters to illustrate the gains in terms of the key vs. storage rate ratio as compared to previous code designs. We observe that one variant of nested convolutional codes achieves a higher rate ratio than all other code designs in the literature but it may have a high hardware cost. Another variant of nested convolutional codes with low complexity is illustrated to perform similarly to the best previous codes in the literature. We also computed known finite-length bounds for our code construction to show the gaps between the performance of the designed codes and these bounds.

ACKNOWLEDGMENTS

This work was performed while O. Günlü was with the Chair of Communications Engineering, Technical University of Munich. O. Günlü was supported by the German Federal Ministry of Education and Research (BMBF) within the national initiative for "Post Shannon Communication (NewCom)" under the Grant 16KIS1004, and by the German Research Foundation (DFG) under grant KR 3517/9-1. V. Sidorenko is on leave from the Institute for Information Transmission Problems, Russian Academy of Sciences. His work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 801434) and by the Chair of Communications Engineering at the Technical University of Munich. The work of G. Kramer was supported by an Alexander von Humboldt Professorship endowed by the BMBF.

REFERENCES

[1] Rudolf Ahlswede and Imre Csiszár. 1993. Common randomness in information theory and cryptography - Part I: Secret sharing. *IEEE Trans. Inf. Theory* 39, 4 (July 1993), 1121–1132. https://doi.org/10.1109/18.243431

[2] Mattias Andersson, Vishwambhar Rathi, Ragnar Thobaben, Jörg Kliewer, and Mikael Skoglund. 2010. Nested polar codes for wiretap and relay channels. *IEEE Commun. Lett.* 14, 8 (Aug. 2010), 752–754. https://doi.org/10.1109/LCOMM.2010.08.100875

[3] Erdal Arikan. 2009. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inf. Theory* 55, 7 (July 2009), 3051–3073. https://doi.org/10.1109/TIT.2009.2021379

[4] Patrizio Campisi. 2013. *Security and privacy in biometrics.* London, U.K.: Springer-Verlag.

[5] Bin Chen, Tanya Ignatenko, Frans M.J. Willems, Roel Maes, Erik van der Sluis, and Georgios Selimis. 2017. A robust SRAM-PUF key generation scheme based on polar codes. In *IEEE Global Commun. Conf.* Singapore, 1–6. https://doi.org/10.1109/GLOCOM.2017.8254007

[6] Remi A. Chou and Matthieu R. Bloch. 2014. Separation of reliability and secrecy in rate-limited secret-key generation. *IEEE Trans. Inf. Theory* 60, 8 (Aug. 2014), 4941–4957. https://doi.org/10.1109/TIT.2014.2323246

[7] Remi A. Chou, Matthieu R. Bloch, and Jörg Kliewer. 2015. Polar coding for empirical and strong coordination via distribution approximation. In *IEEE Int. Symp. Inf. Theory.* Hong Kong, China, 1512–1516. https://doi.org/10.1109/ISIT.2015.7282708

[8] Imre Csiszár and Prakash Narayan. 2000. Common randomness and secret key generation with a helper. *IEEE Trans. Inf. Theory* 46, 2 (Mar. 2000), 344–366. https://doi.org/10.1109/18.825796

[9] Paul W. Cuff, Haim H. Permuter, and Thomas M. Cover. 2010. Coordination Capacity. *IEEE Trans. Inf. Theory* 56, 9 (Sep. 2010), 4181–4206. https://doi.org/10.1109/TIT.2010.2054651

[10] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. 2008. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* 38, 1 (Jan. 2008), 97–139. https://doi.org/10.1007/978-3-540-24676-3_31

[11] Blaise Gassend. 2003. *Physical random functions.* Master's thesis. M.I.T., Cambridge, MA.

[12] Onur Günlü. 2018. *Key Agreement with Physical Unclonable Functions and Biometric Identifiers.* Ph.D. Dissertation. TU Munich, Germany. published by Dr. Hut Verlag.

[13] Onur Günlü, Onurcan İşcan, Vladimir Sidorenko, and Gerhard Kramer. 2019. Code Constructions for Physical Unclonable Functions and Biometric Secrecy Systems. *IEEE Trans. Inf. Forensics Security* 14, 11 (Nov. 2019), 2848–2858. https://doi.org/10.1109/TIFS.2019.2911155

[14] Onur Günlü, Tasnad Kernetzky, Onurcan İşcan, Vladimir Sidorenko, Gerhard Kramer, and Rafael F. Schaefer. 2018. Secure and reliable key agreement with physical unclonable functions. *Entropy* 20, 5 (May 2018). https://doi.org/10.3390/e20050340

[15] Onur Günlü and Gerhard Kramer. 2018. Privacy, secrecy, and storage with multiple noisy measurements of identifiers. *IEEE Trans. Inf. Forensics Security* 13, 11 (Nov. 2018), 2872–2883. https://doi.org/10.1109/TIFS.2018.2834303

[16] Onur Günlü and Rafael F. Schaefer. 2020. Low-complexity and Reliable Transforms for Physical Unclonable Functions. In *IEEE Int. Conf. Acoustics, Speech, Signal Process.* Barcelona, Spain. to appear.

[17] Tanya Ignatenko, Geert jan Schrijen, Boris Skoric, Pim Tuyls, and Frans Willems. 2006. Estimating the Secrecy-Rate of Physical Unclonable Functions with the Context-Tree Weighting Method. In *IEEE Int. Symp. Inf. Theory.* Seattle, WA, 499–503. https://doi.org/10.1109/ISIT.2006.261765

[18] Tanya Ignatenko and Frans M. J. Willems. 2009. Biometric systems: Privacy and secrecy aspects. *IEEE Trans. Inf. Forensics Security* 4, 4 (Dec. 2009), 956–973. https://doi.org/10.1109/TIFS.2009.2033228

[19] Rolf Johannesson and Kamil Zigangirov. 2015. *Fundamentals of Convolutional Coding* (2 ed.). 1–667 pages. https://doi.org/10.1002/9781119098799

[20] Ari Juels and Martin Wattenberg. 1999. A fuzzy commitment scheme. In *ACM Conf. Comp. Commun. Security.* New York, NY, 28–36. https://doi.org/10.1145/319709.319714

[21] Ashish Khisti, Suhas N. Diggavi, and Gregory W. Wornell. 2012. Secret-key generation using correlated sources and channels. *IEEE Trans. Inf. Theory* 58, 2 (Feb. 2012), 652–670. https://doi.org/10.1109/TIT.2011.2173629

[22] Manabu Koide and Hirosuke Yamamoto. 2010. Coding theorems for biometric systems. In *IEEE Int. Symp. Inf. Theory.* Austin, TX, 2647–2651. https://doi.org/10.1109/ISIT.2010.5513689

[23] Victoria Kostina. 2013. *Lossy Data Compression: Nonasymptotic Fundamental Limits.* Ph.D. Dissertation. Princeton University, NJ, USA.

[24] Onur Ozan Koyluoglu and Hesham El Gamal. 2012. Polar coding for secure transmission and key agreement. *IEEE Trans. Inf. Forensics Security* 7, 5 (Oct. 2012), 1472–1483. https://doi.org/10.1109/TIFS.2012.2207382

[25] Lifeng Lai, SiuWai Ho, and H. Vincent Poor. 2011. Privacy-security trade-offs in biometric security systems - Part I: Single use case. *IEEE Trans. Inf. Forensics Security* 6, 1 (Mar. 2011), 122–139. https://doi.org/10.1109/TIFS.2010.2098872

[26] Wenhui Li, Vladimir Sidorenko, Thomas Jerkovits, and Gerhard Kramer. 2019. On Maximum-Likelihood Decoding of Time-Varying Trellis Codes. In *International Symposium Problems of Redundancy in Information and Control Systems*. Moscow, Russia, 104–109.

[27] Ruoheng Liu, Yingbin Liang, H. Vincent Poor, and Predrag Spasojevic. 2007. Secure Nested Codes for Type II Wiretap Channels. In *IEEE Inf. Theory Workshop*. Tahoe City, CA, 337–342. https://doi.org/10.1109/ITW.2007.4313097

[28] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. 2009. A Soft Decision Helper Data Algorithm for SRAM PUFs. In *IEEE Int. Symp. Inf. Theory*. Seoul, Korea, 2101–2105. https://doi.org/10.1109/ISIT.2009.5205263

[29] Hessam Mahdavifar and Alexander Vardy. 2011. Achieving the secrecy capacity of wiretap channels using polar codes. *IEEE Trans. Inf. Theory* 57, 10 (Oct. 2011), 6428–6443. https://doi.org/10.1109/TIT.2011.2162275

[30] Ueli Maurer. 1993. Secret key agreement by public discussion from common information. *IEEE Trans. Inf. Theory* 39, 3 (May 1993), 2733–742. https://doi.org/10.1109/18.256484

[31] Lars Palzer and Roy Timo. 2016. A converse for lossy source coding in the finite blocklength regime. In *Int. Zurich Seminar Commun.* Zurich, Switzerland, 15–19. https://doi.org/10.3929/ethz-a-010645199

[32] Ravikanth Pappu. 2001. *Physical one-way functions.* Ph.D. Dissertation. M.I.T., Cambridge, MA.

[33] Gregory Poltyrev. 1994. Bounds on the decoding error probability of binary linear codes via their spectra. *IEEE Trans. Inf. Theory* 40, 4 (July 1994), 1284–1292. https://doi.org/10.1109/18.335935

[34] Yury Polyanskiy, H. Vincent Poor, and Sergio Verdu. 2010. Channel Coding Rate in the Finite Blocklength Regime. *IEEE Trans. Inf. Theory* 56, 5 (May 2010), 2307–2359. https://doi.org/10.1109/TIT.2010.2043769

[35] Rose Y. Shao, Shu Lin, and Marc P. C. Fossorier. 2003. Two decoding algorithms for tailbiting codes. *IEEE Trans. Commun.* 51, 10 (Oct. 2003), 1658–1665. https://doi.org/10.1109/TCOMM.2003.818084

[36] Boris Škorić. 2012. Quantum readout of physical unclonable functions. *Int. J. Quantum Inf.* 10, 1 (Feb. 2012), 1250001. https://doi.org/10.1142/S0219749912500013

[37] David Slepian and Jack Wolf. 1973. Noiseless coding of correlated information sources. *IEEE Trans. Inf. Theory* 19, 4 (July 1973), 471–480. https://doi.org/10.1109/TIT.1973.1055037

[38] Ido Tal and Alexander Vardy. 2015. List Decoding of Polar Codes. *IEEE Trans. Inf. Theory* 61, 5 (May 2015), 2213–2226. https://doi.org/10.1109/TIT.2015.2410251

[39] Jack K. Wolf and Andrew J. Viterbi. 1996. On the weight distribution of linear block codes formed from convolutional codes. *IEEE Trans. Commun.* 44, 9 (Sep. 1996), 1049–1051. https://doi.org/10.1109/26.536907

[40] Aaron D. Wyner. 1975. The wire-tap channel. *Bell Labs Tech. J.* 54, 8 (Oct. 1975), 1355–1387. https://doi.org/10.1002/j.1538-7305.1975.tb02040.x

[41] Aaron D. Wyner and Jacob Ziv. 1973. A theorem on the entropy of certain binary sequences and applications: Part I. *IEEE Trans. Inf. Theory* 19, 6 (Nov. 1973), 769–772. https://doi.org/10.1109/TIT.1973.1055107