Soft Decision Error Correction for Compact Memory-Based PUFs using a Single Enrollment

Vincent van der Leest[1], Bart Preneel[2], and Erik van der Sluis[1]

[1] Intrinsic-ID, Eindhoven, The Netherlands
http://www.intrinsic-id.com
[2] KU Leuven Dept. Electrical Engineering-ESAT/SCD-COSIC and IBBT, Belgium
http://www.kuleuven.be

Abstract. Secure storage of cryptographic keys in hardware is an essential building block for high security applications. It has been demonstrated that Physically Unclonable Functions (PUFs) based on uninitialized SRAM are an effective way to securely store a key based on the unique physical characteristics of an Integrated Circuit (IC). The start-up state of an SRAM memory is unpredictable but not truly random as well as noisy, hence privacy amplification techniques and a Helper Data Algorithm (HDA) are required in order to recover the correct value of a full entropy secret key. At the core of an HDA are error correcting techniques. The best known method to recover a full entropy 128-bit key requires 4700 SRAM cells. Earlier work by Maes et al. has reduced the number of SRAM cells to 1536 by using soft decision decoding; however, this method requires multiple measurements (and thus also power resets) during the storage of a key, which will be shown to be an unacceptable overhead for many applications. This article demonstrates how soft decision decoding with only a single measurement during storage can reduce the required number of SRAM cells to 3900 (a 17% reduction) without increasing the size of en-/decoder. The number of SRAM cells can even be reduced to 2900 (a 38% reduction). This does increase cost of the decoder, but depending on design requirements it can be shown to be worthwhile. Therefore, it is possible to securely store a 128-bit key at a very low overhead in an IC or FPGA.

1 Introduction

Due to submicron process variations during manufacturing, every transistor in an IC has slightly different physical properties. These properties can be measured and since the process variations are uncontrollable, they result in features that cannot be copied. Therefore, it is possible to create an electronic device with a unique electronic fingerprint that offers a very strong resistance against cloning.

Physically Unclonable Functions (PUFs) are based on an electronic circuit that measures the responses of hardware to random input challenges. These responses depend on the unique and uncontrollable physical properties of the device and allow to authenticate the device. PUF responses are inherently noisy, due to the presence of noise during the measurements. Helper Data Algorithms

(HDAs) based on forward error correction have been developed to correct this noise. This paper presents an improved soft decoding algorithm for HDAs. In soft decoding, decisions are not based on the 0 or 1 value of a bit but on the probability for a bit to take the value 0 or 1.

SRAM memories have specific properties, which make them very suitable for use as PUFs: it turns out that uninitialized SRAM contains an unpredictable value because of the unbalance between two transistors; this unbalance depends on process variations and is hard to control. The unpredictable start-up value can be used for secure storage of a key. This is a convenient alternative to key storage in EEPROM, because EEPROM brings additional costs and is typically not available when a new technology node is rolled out. In addition to these benefits, SRAM PUFs are also more secure than EEPROM since the key is completely absent when the device is powered off. The SRAM values observed do contain entropy but they are not truly random. This can be resolved by using a larger SRAM in combination with privacy amplification (as shown by Guajardo et al. in [6]).

1.1 Related Work

Pappu [14] introduced the concept of PUFs in 2001 under the name Physical One-Way Functions. The proposed technology was based on obtaining a response (scattering pattern) when shining a laser on a bubble-filled transparent epoxy wafer. In 2002 this principle was translated by Gassend et al. [5] into Silicon Physical Random Functions. These functions make use of the manufacturing process variations in ICs, with identical masks, to uniquely characterize each IC. For this purpose the frequency of ring oscillators were measured. Using this method (now known as a Ring Oscillator PUF), they were able to characterize ICs. In 2004 Lee et al. [9] proposed another PUF that is based on delay measurements, the Arbiter PUF.

Besides intrinsic PUFs based on delay measurements a second type of PUF in ICs is known: the memory-based PUF. These PUFs are based on the measurement of start-up values of memory cells. This memory-based PUF type includes SRAM PUFs, which were introduced by Guajardo et al. in 2007 [6]. Furthermore, so-called Butterfly PUFs were introduced in 2008 by Kumar et al. [8], D Flip-Flop PUFs by Maes et al. [11] in 2008, and recently Buskeeper PUFs by Simons et al. [15] in 2012.

The first HDAs for generating cryptographic keys from PUFs were introduced by Linnartz et al. [10] in 2003 (as Shielding Functions) and Dodis et al. [4] in 2004 (as Fuzzy Extractors). After these introductions, secure use of Fuzzy Extractors was discussed by Boyen in [2]. A first efficient hardware implementation of an HDA was described in [1] by Bösch et al. The first HDA using soft decision error correction for memory-based PUFs was proposed by Maes et al. [12, 13] in 2009.

1.2 Our Contribution

This paper introduces a new soft decision decoding scheme for HDAs used in PUF implementations. To the best of our knowledge, this is the first ever soft decision decoder for memory-based PUFs that only requires a single PUF measurement during enrollment. This approach offers a significant increase in practical usability over the method proposed in [12, 13], as will be shown in Sect. 3.

Besides using only a single enrollment measurement, the soft decision decoding scheme as introduced in this paper allows for an efficient hardware implementation. For that reason the construction only uses simple linear block codes such as repetition, Reed-Muller (RM), and Golay codes. This paper will show that these soft decision decoders offer substantial added value over their hard decision counterparts from [1] and are also efficiently implementable in hardware (in contrast to more complex codes such as BCH and LDPC).

1.3 Paper Outline

Section 2 introduces the concept of HDAs. The state of the art of soft decision decoding in PUF HDAs is presented in Sect. 3. The problem of the known method for soft decision decoding is discussed together with how our proposed method can improve this. When this has been established, Sect. 4 describes the newly proposed method in more detail. We will compare the performance of our new method to known implementations of hard decision decoding. Results of this comparison can be found in Sect. 5. Conclusions are drawn in Sect. 6.

2 Helper Data Algorithms

2.1 Construction for Secure Key Storage

As stated earlier, an important application of PUFs is secure key storage [16]. Memory-based PUFs can be used for this purpose. In this paper we use SRAM PUFs as the example of memory-based PUFs. Of all memory-based PUFs, SRAM is the one with the best performance regarding both reproducibility and entropy (as demonstrated in [3]). Secure key storage with PUFs makes use of an HDA to securely store and reconstruct the key. Different constructions of HDAs exist. The implementation used in this paper is depicted in Fig. 1. We distinguish two phases in this HDA: enrollment and reconstruction.

Enrollment. During enrollment the key is programmed into the device, comparable to the key programming phase for other secure key storage mechanisms. First, the response of the targeted PUF is measured. This response is called the reference PUF response (R) and is the input of the Fuzzy Extractor [2, 4, 10]. This Fuzzy Extractor (FE) derives a cryptographic key from a random secret and computes helper data W by xor-ing the encoded secret with R. In the reconstruction phase, W enables FE to reconstruct the exact same ("programmed") cryptographic key from a new response of this specific PUF. The helper data is stored in non-volatile memory attached to the device and is public information.

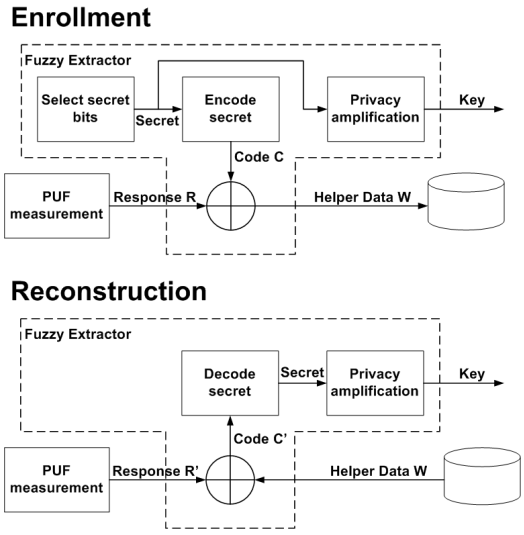

Fig. 1. Enrollment and reconstruction for the HDA.

Reconstruction. In the reconstruction phase the same PUF is measured again and its response (R', which is slightly different from R) is input to the FE. The FE uses W and R' to reconstruct the cryptographic key that was "programmed" during enrollment. If R' is close enough to R, the original key will be successfully reconstructed using information reconciliation.

2.2 Fuzzy Extractors

This section explains in more detail the most important building blocks of a Fuzzy Extractor.

Secret Encoding. Secret encoding is performed during enrollment and consists of selecting a random secret and encoding this secret with the chosen error correction code. In this paper we use linear error correcting codes with length n, dimension k and minimum Hamming distance d, which are listed as $[n, k, d]$ codes. The encoded secret C is xor-ed with R (this method is called code-offset technique) to create the value W, which will be used during reconstruction.

Information Reconciliation. In Fig. 1 information reconciliation can be found as "Decode secret" during reconstruction. This can be done after R' has been xor-ed with W to create C', which differs from C at the same positions that R' differs from R. Hence if R' and R are sufficiently close together (depending on how many errors can be corrected by the selected code construction), C' can be corrected into C and decoded into the secret encoded at enrollment.

Privacy Amplification. As an attacker may have partial information on the PUF (due to non-randomness in the response), the selected secret should be compressed into a cryptographic key with maximum entropy. This minimizes the knowledge of the attacker about the value of the key. According to [6] the secrecy rate for SRAM PUFs is 0.76, which indicates that for deriving a key of 128 bits with full entropy a secret of $\lceil 128/0.76 \rceil = 171$ bits is required. This is the number of secret bits that will be used for our analyses. This paper will not focus on privacy amplification, but note that compression can be achieved (for example) with a cryptographic hash function.

3 Soft Decision Decoding

3.1 State of the Art

When using soft decision decoding, reliability information about incoming bits is provided along with the bit-value of '0' or '1'. In other words, every bit at the input of a soft decision decoder is accompanied by a value that indicates the confidence level of this specific bit. Soft decision decoders can use this additional information to improve their error correcting capabilities; it is well known that on a typical Gaussian channel soft decision decoding results in an improvement of about 2 dB over hard decision. The goal is to help the decoder to output the most likely transmitted codeword and decrease the error rate at its output.

Soft decision decoding for memory-based PUFs has only been used in the literature by Maes et al. in [12, 13]. Their proposal is the following:

- **During enrollment** several measurements of the (SRAM) PUF are performed. Based on these measurements an error probability for each PUF bit is derived and stored together with the helper data. The more stable the response of a specific PUF bit is during these multiple enrollment measurements, the higher the confidence level of the value of this bit will be.
- **During reconstruction** error probabilities from enrollment are used to indicate the confidence level of each individual bit. It is proven in [12, 13] that using this soft decision information, less PUF bits are required to successfully reconstruct the secret bits that are used for the cryptographic key.

3.2 Motivation for Construction

The problem with the method from [12, 13] is that multiple enrollment measurements are required. This has the following consequences:

- Non-volatile storage will be required in the device containing the PUF. Values of the multiple enrollment measurements need to be added in order to obtain the error probability of each bit. A key business case for PUFs is the replacement of non-volatile key storage (as described in Sect. 1). Therefore, the method from [12, 13] gives up on the essential advantage of PUFs for key storage and introduces additional process steps (introducing extra delay), costs, footprint while decreasing security (because of possible attacks on non-volatile memory).

– The size of the required storage grows with the number of measurements performed. For example, when 3 enrollment measurements are performed, the sum value of each PUF bit can take on any integer value between 0 and 3. Therefore, this requires an additional 2 bits of storage per PUF bit. With 7 measurements 3 bits are required (and so on).
– Multiple measurements (and additional processing) leads to a longer time required for enrolling each PUF. This could lead to problems when enrolling millions of devices in production lines.

To solve these practical problems, we propose a new method for soft decision decoding. The requirements for this new method are the following:

– It should only use one measurement during enrollment (and reconstruction).
– It should be efficiently implementable in hardware.

The methods proposed in this paper will only focus on HDAs using the code-offset technique with linear block codes. Other codes, such as LDPC and convolutional codes, are more complex to decode and not well suited to deal with the limited amount of data available in PUF implementations. Therefore they will not be considered in this paper.

3.3 Our Proposal

The previous section has motivated why we propose a new low footprint HDA construction. This HDA should require as few PUF bits as possible in combination with low algorithmic complexity, while avoiding the implementation issues from the previous soft decision decoding method.

The method for soft decision coding proposed here is based on the concatenated codes from [1]. Figure 2 shows the flowcharts of encoding and decoding in the proposed HDA. Encoding is performed in a similar manner as for the hard decision construction (and is thus only based on a single PUF measurement). During decoding however, there are two differences with the hard decision construction:

– The repetition decoder is replaced by a quantizer, which derives probabilistic information from a single reconstruction measurement.
– The second decoder is a soft decision decoder (using the probabilistic information from the quantizer).

When using the repetition decoder as a quantizer, it "weighs" the amount of ones and zeros at its (non-probabilistic) input and outputs a (probabilistic) value between 0 and 1 that corresponds to this input. An input string consisting of s bits with i ones and $s-i$ zeros will be converted by the quantizer into an output value of i/s. These strings of length s are the sum (modulo 2) of the repetitive output of the encoder with noise of the PUF measurement. Hence, without noise the value i would either be 0 or s. So the closer i is to one of these values, the more confident the soft output value of the quantizer will be.

The soft values at the output of the quantizer are used as input for the soft decision decoder. Candidate soft decision decoders are described in Sect. 4.

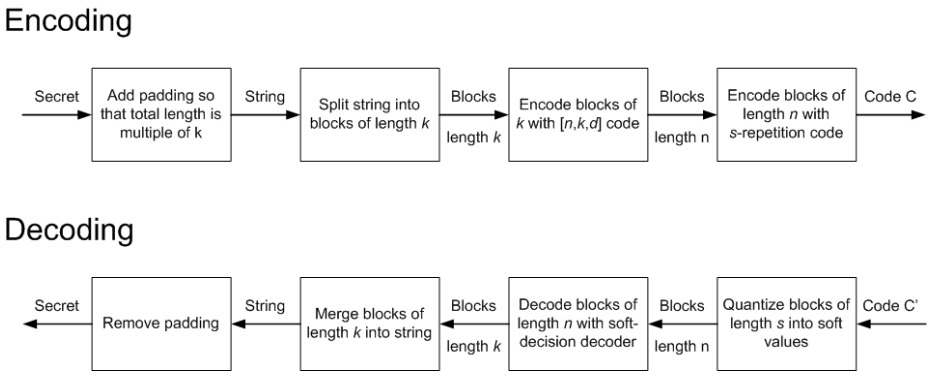

Fig. 2. Encoding and decoding as used in the proposed HDA.

4 Soft Decision Decoders

We propose two methods for performing soft decision decoding, which are based on well-known error correction codes and more in particular on concatenated codes as described in [1]. Furthermore, it is important that both methods are implementable in hardware without too much overhead on resources. This rules out the more complex (BCH) codes from [1], since it is not possible to convert them into a hardware efficient soft decision code (and even the resource efficiency of some hard decision BCH implementations is questionable). To illustrate cost effectiveness of both solutions, a comparison between the hard- and soft decision implementations of these codes will be given in Sect. 5.4.

4.1 Brute-Force RM Decoder

The first proposed method is brute-force, which can be used for codes with a limited set of codewords (that is, with a small dimension k). In this method the soft input of the decoder is compared to all possible codewords. Based on Euclidean Distance, the most likely codeword from the list is selected to be decoded. In our analysis we use this method for evaluating soft decision decoding with two concatenated codes. The two constructions are repetition in combination with the Reed-Muller[16,5,8] code and with Reed-Muller[8,4,4]. It is clear that both

Algorithm 1: Brute-Force Soft Decision Reed-Muller Decoder

Input: String of size n consisting of soft values between 0 and 1.
Actions:
1. Calculate Euclidean Distance of input string to all possible codewords of RM code.
2. Select codeword of length n with lowest Euclidean Distance to input.
3. Decode codeword to corresponding encoded secret bits.
Output: Binary string of size k.

codes only have a limited number of codewords (32 and 16 respectively). Algorithm 1 describes how the proposed brute-force soft decision decoder works. In both constructions the repetition decoders are used as quantizers to create the soft input for the RM decoders, as described in Sect. 3.3.

4.2 Hackett Decoder

The second method is a concatenated code using repetition and Golay[24,12,8], where the Golay decoder is used for soft decision decoding as described in [7]. Again the repetition code is used as a quantizer, which produces soft values that are used as input for soft decision Golay decoder. The algorithm of this Golay decoder is described in Algorithm 2 and is visualized in Fig. 3.

Algorithm 2: Hackett Soft decision Golay Decoder

Input: String of size n consisting of soft values between 0 and 1.
Actions:
1. Convert input to corresponding hard (binary) values.
2. Based on soft input values, select 4 bits from input with least confidence.
3. Calculate overall parity of hard values.
   if parity is even -> flip least confident bit.
4. Initialize values required for loop: ED_min $= \infty$, $y =$ "hard values" and $k = 0$.
5. Error correct y using hard decision Golay[24,12,8].
6. Calculate Euclidean Distance of resulting string to soft input.
   if Euclidean Distance $<$ ED_min -> Replace ED_min and $z = y$.
   if k < 7 -> flip two bits (as described in Table 1) to get new y, go back to step 5.
   else -> Decode codeword z to corresponding encoded secret bits.

Output: Binary string of size k.

Furthermore, Table 1 provides an overview of how the 8 different patterns are created from the original value of y by flipping bits (all possible patterns with even weight consisting of 4 bits). In this table b_0 denotes the least confident bit, b_1 the second least confident, etc. According to [7] only these 8 patterns are required (and not all 16 possibilities when flipping 4 bits), because it is already assured that the parity of all values of y are odd, which will lead to an odd number of errors. It is also claimed that hard decision decoding of an even number of errors rarely yields a codeword closer to the soft input than decoding with an odd number of errors. Therefore, patterns with an even parity are not used in this decoder.

5 Soft vs. Hard Decision Comparison

This section is dedicated to demonstrating the added value of soft decision decoding for PUF implementations. For that purpose the soft decision implementations from the previous section are compared to their hard decision counterparts based

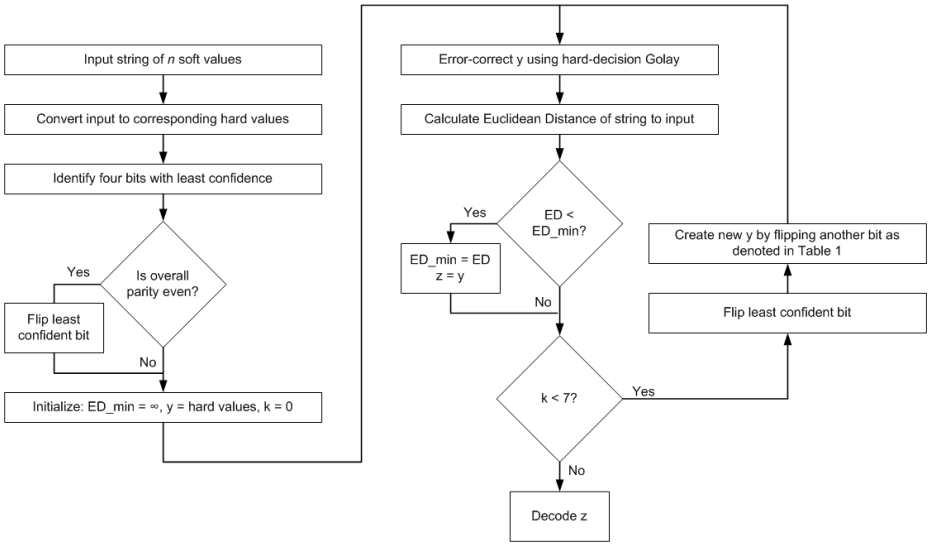

Fig. 3. Flowchart of Hackett decoder.

Table 1. Bits flipped in comparison to initial value of y for different values of k.

k	b_0	b_1	b_2	b_3	Bits flipped compared to $k-1$
0	0	0	0	0	-
1	1	0	0	1	b_0 and b_3
2	0	0	1	1	b_0 and b_2
3	1	0	1	0	b_0 and b_3
4	0	1	1	0	b_0 and b_1
5	1	1	0	0	b_0 and b_2
6	0	1	0	1	b_0 and b_3
7	1	1	1	1	b_0 and b_2

on error correcting performance and required resources. This will show that the number of PUF bits required for successfully reconstructing keys is much smaller when using soft decision decoding and that the additional resources required are limited.

The performance of the proposed soft decision decoders will not be compared to those from [12, 13]. Even though the performance of the decoders from [12, 13] is better than those presented here, this is an unfair comparison. Those decoders require multiple enrollment measurements, which leads to the problems listed in Sect. 3.2. The decoders proposed in this paper and their hard decision counterparts do not have these problems and can therefore be compared fairly.

The system used for context in this section derives 171 secret bits from an SRAM PUF. This example has been taken from [6] and is also referred to in [12, 13] and [1]. The bit error rate of the PUF data (noise of PUF measurement)

is called ϵ and will be 15%, which is similar to these same references and is a good representation of noise on SRAM used at operating temperatures ranging from -40°C to +85°C (industrial temperature standard). Furthermore, the False Rejection Rate (FRR) of the 171 secret bits should be lower than 10^{-6} (i.e. the probability of incorrectly decoding a secret key with $\epsilon = 0.15 <$ one in a million).

5.1 Hard Decision Concatenated Codes

The FRR of the hard decision decoders can be calculated using a set of formulas. The first step is to calculate the error probability of the repetition decoder. This probability depends on the length of the repetition code as well as the value of ϵ and is defined as follows: When the number of bit errors at the input of the repetition decoder is higher than half of the length of the code (repetition code can only be of odd length, to avoid equal number of zeros and ones), the output of the decoder will be incorrect. This leads to the following formula:

$$P_{\mathrm{e}_{rep}} = \sum_{i=\lceil s/2 \rceil}^{s} \binom{s}{i} \epsilon^i (1-\epsilon)^{s-i} = 1 - \sum_{i=0}^{\lfloor s/2 \rfloor} \binom{s}{i} \epsilon^i (1-\epsilon)^{s-i} \ .$$

Using the error probability of the repetition decoder in combination with the parameters of the hard decision code, the error probability of the concatenated code can be derived. When the number of errors from the repetition decoder is too high for the hard decision decoder to correct ($> \lfloor (d-1)/2 \rfloor$), the outcome of the hard decision decoder will be incorrect. The corresponding formula, where $t = \lfloor (d-1)/2 \rfloor$, is:

$$P_{\mathrm{e}_{code}} = \sum_{i=t+1}^{n} \binom{n}{i} P_{\mathrm{e}_{rep}}^i (1 - P_{\mathrm{e}_{rep}})^{n-i} = 1 - \sum_{i=0}^{t} \binom{n}{i} P_{\mathrm{e}_{rep}}^i (1 - P_{\mathrm{e}_{rep}})^{n-i} \ .$$

Finally when these error probabilities are known, the total FRR of the key can be calculated (this step has been omitted in [1], which may give the false impression that the results from this paper differ from those in [1]). A key can only be reconstructed successfully when all required output blocks from the concatenated decoder (and thus all secret bits) are correct. Dividing the length of the secret by k (number of secret bits per decoding) leads to the number of output blocks that need to be decoded correctly to reconstruct the key successfully. In other words the FRR of the hard decision decoders is defined as:

$$FRR_{key} = 1 - (1 - P_{\mathrm{e}_{code}})^{\lceil secret\ length/k \rceil} = 1 - (1 - P_{\mathrm{e}_{code}})^{\lceil 171/k \rceil} \ .$$

5.2 Soft Decision Simulation Results

Unfortunately, it is not straightforward to define formulas for calculating the FRR of the soft decision decoders. In order to be able to evaluate the performance of these systems, the decoders have been simulated. These simulations have been

performed by encoding random secrets with the concatenated encoders, adding random noise with $\epsilon = 0.15$, and decoding the resulting string with the soft decision decoders as described in earlier sections. The results of this simulation can be found in Fig. 4. This figure displays the FRRs of the three different concatenated soft decision decoders as a function of the length of the repetition code. Note that the repetition code can be of even length in this system, since it is used as a quantizer when decoding (and not as a hard decision repetition decoder). A threshold is set in the figure at an FRR of 10^{-6}, which allows us to derive the shortest repetition length required to achieve an FRR (for a key based on 171 secret bits) below this threshold.

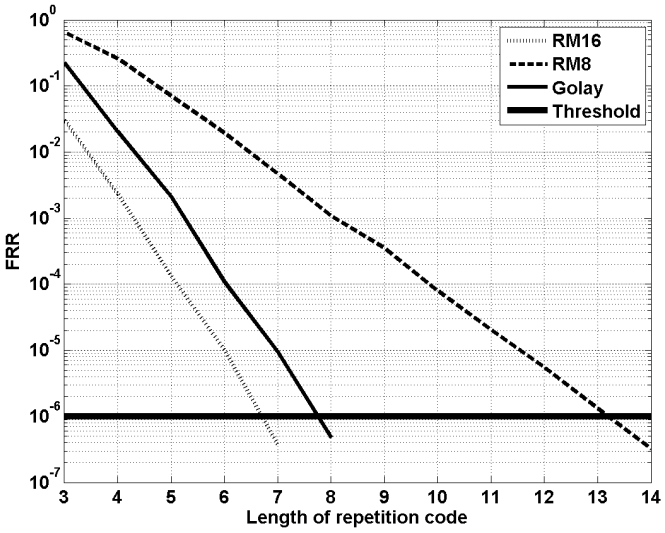

Fig. 4. Simulation results for soft decision decoders of RM[16,5,8], RM[8,4,4], and Golay[24,12,8] codes.

5.3 Comparison

Based on the formulas from Sect. 5.1 and the simulation results from Sect. 5.2 the performance of the hard- and soft decision decoders can be compared. Table 2 shows the amount of SRAM required to reconstruct a key based on 171 secret bits with a total FRR below 10^{-6}. We conclude that the amount of SRAM required for soft decision decoding is significantly lower than that of hard decision decoding (RM[16,5,8]: 47% decrease, RM[8,4,4]: 44%, Golay: 38%). This shows that the soft decision decoders are very suitable when implementing an SRAM PUF HDA, which is optimized on the amount of SRAM required.

Table 2. Results hard and soft decision codes (deriving 171 secret bits, FRR $< 10^{-6}$).

Code	Type	Rep. Length	FRR	Amount of SRAM (Bytes)[a]
RM[16,5,8]	Hard	13	$1.6 \cdot 10^{-7}$	$\lceil 171/5 \rceil * 16 * 13/8 =$ **910**
RM[16,5,8]	Soft	7	$3.7 \cdot 10^{-7}$	$\lceil 171/5 \rceil * 16 * 7/8 =$ **490**
RM[8,4,4]	Hard	25	$3.4 \cdot 10^{-7}$	$\lceil 171/4 \rceil * 8 * 25/8 =$ **1075**
RM[8,4,4]	Soft	14	$3.3 \cdot 10^{-7}$	$\lceil 171/4 \rceil * 8 * 14/8 =$ **602**
Golay[24,12,8]	Hard	13	$4.0 \cdot 10^{-7}$	$\lceil 171/12 \rceil * 24 * 13/8 =$ **585**
Golay[24,12,8]	Soft	8	$4.8 \cdot 10^{-7}$	$\lceil 171/12 \rceil * 24 * 8/8 =$ **360**

[a] Calculation method for the amount of SRAM:
$\lceil$required secret bits (171) / secret bits per codeword$\rceil$ = # required codewords
# of codewords * length codewords * repetition length = # of bits / 8 = # of Bytes

5.4 Resource Estimates

Besides a comparison based on the amount of SRAM required for each code construction, one can also compare the codes based on the total amount of resources required. For this purpose the total footprint of each construction will be estimated. One could also compare codes based on either timing or power consumption. However, we believe that these comparison are much less important when comparing hard and soft decision coders. When PUFs are used for key storage, silicon area is the main cost factor (note that this silicon is mostly inactive). Power and delay overhead play only a very minor role, since the coders are only active when generating the key (at start-up of a device). This one-time operation can be done within 1 millisecond at any realistic clock and will therefore not consume much time or power. Therefore, area will be the only factor in this comparison. Results on timing have been added to Table 3 for informational purposes only. Power consumption by the soft decision decoders has been found to be negligible in comparison to regular IC operation (since there is only consumption during a short time at power-up) and is therefore omitted.

The total footprint consists of the following components: the encoder, quantizer/repetition coder, decoder and SRAM. The required resources for each component are estimated based on synthesis and the results can be found in Table 3. In this table all estimates are based on area-optimized IC implementations. These constructions have also been implemented on FPGA, so they can be used on FPGAs with uninitialized SRAM. Since these FPGAs are rare however, the main focus of this paper is on IC implementations. Furthermore, all estimates are denoted in GE[3] and for SRAM a size of 1 GE per bit has been used as a reasonable estimate[4].

[3] GE – Gate Equivalent is a measure of area in any technology. 1 GE is the area of a NAND2 (standard drive strength).

[4] For TSMC 65nm standard cell library raw gate density $\approx$ 854 Kgate/mm^2, while SRAM cells are 0.499μm^2 (6T) [17]. Hence, one SRAM cell is $<$ 0.5 gates (1mm^2 / 854000 = 1.17μm^2). This is without read-out circuitry, which provides significant overhead for small SRAMs. Considering a factor 2 overhead, an SRAM cell $\approx$ 1 GE.

Table 3. Resource estimates for different code constructions.

Code	Type	Dec. clks	Encoder	Quant./Rep.	Decoder	SRAM	**Total**
RM[16,5,8]	Hard	±200	0.12 kGE	0.14 kGE	0.75 kGE	7.3 kGE	**8.3 kGE**
RM[16,5,8]	Soft	±400	0.12 kGE	0.10 kGE	1.1 kGE	3.9 kGE	**5.2 kGE**
RM[8,4,4]	Hard	±100	55 GE	0.19 kGE	0.5 kGE	8.6 kGE	**9.3 kGE**
RM[8,4,4]	Soft	±200	55 GE	0.14 kGE	0.6 kGE	4.8 kGE	**5.6 kGE**
Golay[24,12,8]	Hard	±15	0.30 kGE	0.14 kGE	1.0 kGE	4.7 kGE	**6.1 kGE**
Golay[24,12,8]	Soft	±150	0.30 kGE	0.13 kGE	3.0 kGE	2.9 kGE	**6.3 kGE**

Table 3 shows that soft decision decoding results in a substantial decrease in the SRAM size and in most cases additional overhead for the soft decoder is small. Soft decision decoders require more registers than their hard decision counterparts, since all codeword bits are now represented by a multi-bit soft value. This alone adds 24 to 72 Flip-Flops to the implementations. These are needed independent of speed/area trade-offs. Another unavoidable footprint increase is the calculation and comparison of distances. The speed/area trade-off is mainly determined by the amount of parallelism used here. Finally, the Hackett soft decision Golay decoder introduces additional logic for (among others) the selection of weak bits.

The choice of an HDA implementation depends on the parameter that should be optimized. In this example, when optimizing on the amount of SRAM used by the code construction, the soft decision Golay code should be implemented. If the total footprint of the implementation needs to be minimized, the soft decision RM[16,5,8] is the preferred choice. What is most important to notice is that the results clearly show the benefit of soft decision decoding.

The added value of soft decision decoding will increase even further when an HDA requires error correction with either a lower FRR, a higher ϵ, a larger number of secret bits or multiple keys. In those cases the SRAM will become an even more dominant factor in the total footprint of the implementation. Therefore, it will be more important to decrease the amount of SRAM required. An example in which 5 128-bits keys need to be stored with these code constructions can be found in Table 4. Here we conclude that the soft decision Golay code is favourable when SRAM is the dominant component of the footprint.

Table 4. Estimation of total footprint for different code constructions storing 5 keys.

Code	Type	Encoder	Quant./Rep.	Decoder	SRAM	**Total**
RM[16,5,8]	Hard	0.12 kGE	0.14 kGE	0.75 kGE	36.4 kGE	**37.4 kGE**
RM[16,5,8]	Soft	0.12 kGE	0.10 kGE	1.1 kGE	19.6 kGE	**20.9 kGE**
RM[8,4,4]	Hard	55 GE	0.19 kGE	0.5 kGE	43.0 kGE	**43.7 kGE**
RM[8,4,4]	Soft	55 GE	0.14 kGE	0.6 kGE	24.1 kGE	**24.9 kGE**
Golay[24,12,8]	Hard	0.30 kGE	0.14 kGE	1.0 kGE	23.4 kGE	**24.8 kGE**
Golay[24,12,8]	Soft	0.30 kGE	0.13 kGE	3.0 kGE	14.4 kGE	**17.8 kGE**

Note: In this section the amount of non-volatile memory required to store helper data (outside of the chip) has not been taken into account. The helper data size (in bytes) is equal to that of the SRAM, hence it is clear that this size in-/decreases linearly with the SRAM size.

6 Conclusions

This paper presents a new and efficient method of soft decision error correction decoding that can be used in HDAs for memory-based PUFs. This new method is based on hard decision decoding using concatenated codes as proposed in [1], where the repetition decoder is replaced by a quantizer that creates the input for a soft decision decoder. It results in a code construction for HDAs that requires less PUF bits for error correction. Furthermore, the proposed method of soft decision decoding can be implemented efficiently in hardware and does not suffer from the same practical problems as the soft decision construction from [12, 13].

Using several (hardware) implementations of soft decision decoders, the added value of soft decision decoding has been demonstrated for an HDA that derives 171 secret bits with an FRR below 10^{-6} while $\epsilon = 0.15$. The soft decision decoders decrease the number of PUF bits that are required to derive the secret bits in comparison to their hard decision counterparts by 38% to 47%. This decrease of PUF bits comes at only a limited cost in hardware resources of the decoder, which becomes even less significant when the size of the PUF becomes more dominant in the total footprint of the HDA. The optimal HDA implementation can be chosen based on the parameter that should be kept as small as possible (number of PUF bits, total footprint of HDA, etc.) in combination with the values of FRR, ϵ and secret size.

Acknowledgements. This work has been supported in part by the European Commission through the FP7 programme under contracts 238811 UNIQUE and 216676 ECRYPT II, by the IAP program P6/26 BCRYPT of the Belgian state, and by the Research Council KU Leuven through GOA TENSE (GOA/11/007). The authors would like to thank the anonymous referees for constructive comments.

References

1. Bösch, C., Guajardo, J., Sadeghi, A.R., Shokrollahi, J., Tuyls, P.: Efficient helper data key extractor on FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES'08. LNCS, vol. 5154, pp. 181–197. Springer-Verlag, Heidelberg (2008)
2. Boyen, X.: Reusable cryptographic fuzzy extractors. In: CCS'04. pp. 82–91. ACM, New York, NY, USA (2004), http://doi.acm.org/10.1145/1030083.1030096
3. Claes, M., van der Leest, V., Braeken, A.: Comparison of SRAM and FF PUF in 65nm technology. In: Laud, P. (ed.) NordSec'11. LNCS, vol. 7161, pp. 47–64. Springer-Verlag, Heidelberg (2011)

4. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT'04. LNCS, vol. 3027, pp. 523–540. Springer-Verlag, Heidelberg (2004)
5. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: CCS'02. pp. 148–160. ACM, New York, NY, USA (2002), http://doi.acm.org/10.1145/586110.586132
6. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES '07. LNCS, vol. 4727, pp. 63–80. Springer-Verlag, Berlin, Heidelberg (2007)
7. Hackett, C.: An efficient algorithm for soft-decision decoding of the (24, 12) extended Golay code. IEEE Transactions on Communications 29(6), 909–911 (Jun 1981)
8. Kumar, S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The butterfly PUF protecting IP on every FPGA. In: Tehranipoor, M., Plusquellic, J. (eds.) IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08). pp. 67–70. IEEE Computer Society (2008)
9. Lee, J., Lim, D., Gassend, B., Suh, G., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: IEEE Symposium on VLSI Circuits 2004. pp. 176–179. IEEE (2004)
10. Linnartz, J.P., Tuyls, P.: New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Kittler, J., Nixon, M.S. (eds.) AVBPA'03. LNCS, vol. 2688, pp. 393–402. Springer-Verlag, Heidelberg (2003)
11. Maes, R., Tuyls, P., Verbauwhede, I.: Intrinsic PUFs from flip-flops on reconfigurable devices. In: Workshop on Information and System Security (WISSec 2008). p. 17. Eindhoven, NL (2008)
12. Maes, R., Tuyls, P., Verbauwhede, I.: Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In: Clavier, C., Gaj, K. (eds.) CHES'09. LNCS, vol. 5747, pp. 332–347. Springer-Verlag, Heidelberg (2009)
13. Maes, R., Tuyls, P., Verbauwhede, I.: Soft decision helper data algorithm for SRAM PUFs. In: IEEE International Symposium on Information Theory (ISIT'09). pp. 2101–2105. IEEE Press, Piscataway, NJ, USA (2009)
14. Ravikanth, P.S.: Physical one-way functions. Ph.D. thesis (2001), aAI0803255
15. Simons, P., van der Sluis, E., van der Leest, V.: Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs. In: IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'12), in print. IEEE Computer Society (2012)
16. Skoric, B., Tuyls, P., Ophey, W.: Robust key extraction from physical uncloneable functions. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) Applied Cryptography and Network Security (ACNS'05). LNCS, vol. 3531, pp. 407–422. Springer-Verlag, Heidelberg (2005)
17. Taiwan Semiconductor Manufacturing Company Limited (TSMC): 65nm technology overview. http://www.tsmc.com/english/dedicatedFoundry/technology/65nm.htm