Security Analysis of Index-Based Syndrome Coding for PUF-Based Key Generation

Georg T. Becker
Horst Görtz Institute for IT-Security
Ruhr-Universität Bochum, Germany
georg.becker@rub.de

Alexander Wild
Horst Görtz Institute for IT-Security
Ruhr-Universität Bochum, Germany
alexander.wild@rub.de

Tim Güneysu
Horst Görtz Institute for IT-Security
Ruhr-Universität Bochum, Germany
tim.gueneysu@rub.de

Abstract—**Physical Unclonable Functions (PUFs) as secure providers for cryptographic keys have gained significant research interest in recent years. Since plain PUF responses are typically unreliable, error-correcting mechanisms are employed to transform a fuzzy PUF response into a deterministic cryptographic key. In this context, Index-Based Syndrome Coding (IBS) has been reported as being provably secure in case of identical and independently distributed PUF responses and is therefore an interesting option to implement a highly secure key provider. In this paper we analyze the security of IBS in combination with a k-sum PUF as proposed at CHES 2011. Since for a k-sum PUF the assumption of identical and independently distributed responses does not hold, the notion of leaked bits was introduced at CHES 2011 to capture the security of such constructions. Based on a refined analysis using hamming distance characterization and machine learning techniques, we show that the entropy of the key obtained is significantly lower than expected. More precisely, we obtained from our findings that even the construction from CHES with the highest security claims only achieves a bit entropy rate of 0.39.**

Keywords—Physical Unclonable Functions, Index-Based Syndrome Coding, k-sum PUF, Fuzzy Extractor, Error-Correction

I. Introduction

Physical Unclonable Functions (PUFs) have been proposed as a promising alternative to securely generate cryptographic keys from an entropy source that can be readily provided by integrated circuits. Examples of such entropy sources are signal propagation delays, frequencies of ring-oscillators, initialization values of SRAM cells, voltage levels of sense-amplifiers or bus-keepers, and other metastability effects [5], [6], [7], [11], [14], [18]. Depending on the target device, not all entropy sources are available. For example, SRAM PUFs cannot be implemented on block memories of Xilinx FPGAs due to the global initialization of all memories. Therefore, ring oscillator PUF (RO-PUF) constructions have become quite popular since they can be easily implemented on both; FPGAs and ASICs.

A major problem for PUF-based key generation stems from the unreliability in PUF response that is due to environmental variations, e.g., from temperature or the supply voltage. To overcome this problem, techniques for error correction and entropy extraction need to be integrated into PUF constructions. A common approach is to use fuzzy extractors based on linear codes [4]. In most PUF systems a two-step error correction is applied, such as a repetition coding followed by a BCH code, e.g. in [2], [13], operating directly on the binary output of the sampled entropy source. By inclusion of the reliability information of individual bits, this process can be further improved by introducing soft-decision error correction [12], [19]. Assuming PUF constructions that directly output real values obtained from this reliability information, Yu and Devadas have proposed an alternative scheme based on Index-based Syndrome Coding (IBS) that has been reported to be lightweight and efficient [21]. A further modification of IBS was proposed in [9] called complementary IBS in which more than just one index is used to represent a single bit.

All schemes for deterministic entropy extraction require additional public information that is denoted as helper data. Although being considered public, the helper data is possible target for attacks by an adversary. In particular, an attacker can not only just observe the helper data (cf. Koeberl *et al.* [10]) but also try to modify it to yield new insights about the internals of the PUF. Delvaux *et al.* [3] showed that it is indeed possible to attack the error correction by Paral *et al.* [16] based on substring matching by manipulating the helper data.

In this context IBS was reported as being susceptible to helper data manipulation but as provably secure against passive attacks if PUF responses are independent and identically distributed (abbrev. *i.i.d.*) [21]. Hence, IBS could also be applied to PUFs that have a strong bias if the individual responses would be independent from each other. Since the latter is actually not the case for the k-sum PUF [20] that was originally proposed for use with the IBS approach [21], the same authors presented a more thorough analysis based on information leakage of the IBS syndromes [22]. Based on their analysis and findings, a metric and two constructions were proposed that should respect the number of key bits which can safely be extracted from the same k-sum PUF

Contribution. In this paper we revisit the analysis by Yu *et al.* on k-sum PUF construction and show that the syndromes of IBS leak more information than assumed before. When 13 secret bits per k-sum PUF are generated as proposed by [22], the bit entropy is significantly reduced to 0.39. Even for the conservative case that only two response bits are generated by the same k-sum PUF, significantly less entropy is extracted than expected.

Outline. This paper is structured as follows. We introduce the construction of k-sum PUFs and revisit the claims on their

This work has been partially supported by the German Federal Ministry of Education and Research BMBF (grant 16KIS0015, Project PhotonFX^2).

security in Section II. In Section III we specify two refined attacks on k-sum PUFs using IBS based by applying hamming distance characterization and machine learning techniques. A detailed discussion and conclusion on our findings are provided in Section IV.

II. Background

This section introduces the k-sum PUF construction as proposed in [20], [22], [23] that is combined with an IBS coding scheme. The IBS scheme is applied as fuzzy extractor for the sampled k-sum results that are obtained as real values (instead of binary values).

A. The k-Sum PUF Construction

A k-sum PUF consists of $2k$ ring oscillators (RO) with oscillation frequencies O_i determined by individual signal propagation delays. Two RO instances are logically grouped into a stage. A schematic view of a k-sum PUF instance is given in Figure 1.

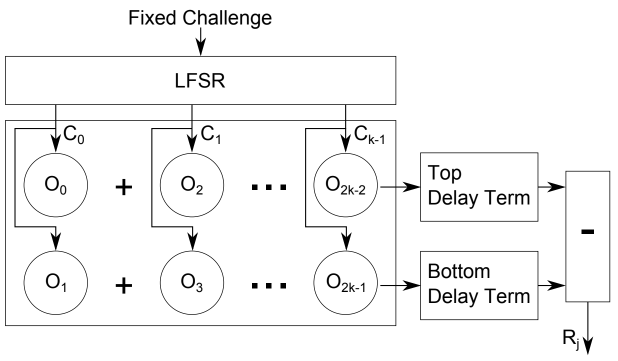

Fig. 1: The k-sum PUF construction.

Differences in generated frequencies for each RO stage are summed up while a challenge bit $C_i \in \{-1, 1\}$ defines to retain or flip the sign of a stage difference. The integer value of the sum defines the soft decision value R_j:

$$R_j = \sum_{i=0}^{k-1} C_i(O_{2i} - O_{2i+1}) = \sum_{i=0}^{k-1} C_i \cdot \Delta O_i$$

As presented in [21] the sign bit of R_j defines the polarity and the remaining bits the confidence level. In case of a negative polarity $R_j < 0$, the soft decision bit of R_j is defined as 0 otherwise as 1. The challenge bits for each soft decision value are derived from a fixed master challenge. Figure 1 shows exemplary the usage of a Linear Feedback Shift Register (LFSR) to generate the challenges.

The soft decision bits are, like the output of most PUF classes, noisy. To eliminate the noise and increase the reliability of the PUF class a fuzzy extractor called Index-Based Syndrome Coding was introduced in [21]. Considering a sequence of $q \leq 2^s$ soft decisions values $R = (R_0, \ldots, R_{q-1})$ generated from a fixed challenge and a given secret bit B. The IBS encoder is defined as:

$$P = Enc^B(R) \begin{cases} = \arg\min(R_j) & \text{if } B = 0 \\ = \arg\max(R_j) & \text{if } B = 1 \end{cases}$$

The encoder result P can be seen as s-bit pointer or index which points either to the min or the max value of R, depending on the given secret bit B. Due to the argument selection of the minimum or maximum, the soft decision value with the highest confidence level is chosen from the sequence R. P is stored as syndrome for the Index-Based Syndrome decoder.

The IBS decoder restores a secret bit B from a soft decision sequence $R' = (R'_0, \ldots, R'_{q-1})$ which is generated from the same challenges as R. The decoding process is defined as

$$B' = Dec^P(R') = \text{sign}(R'_P) = \begin{cases} = 0 & \text{if } R'_P < 0 \\ = 1 & \text{if } R'_P \geq 0 \end{cases}$$

In case $B' = B$ the decoder successfully restored the secret bit B. As the syndrome value P indexes the soft decision value R_P from R with the highest confidence level, the IBS decoder shows a very high reliability as noted in [23].

B. Security Assumptions

The information-theoretic proof on IBS given in [22] is based on the assumption of independent and identically distributed responses which is not given for k-sum PUFs. To provide evidence for the security of IBS without relying on the i.i.d. assumption, Yu *et al.* introduced the notion of leaked bits in [22] to compute IBS parameters that are secure against an adversary who employs machine-learning techniques. The main idea of their security argument is to compute how many bits are leaked by each syndrome and to determine the minimum number of bits needed to model a k-sum PUF. Based on these findings a designer can determine the maximum number of syndromes that can be revealed from a single k-sum PUF without sacrificing security. They conclude that several k-sum PUFs are needed to generate a full 128-bit key. Since modeling a k-sum PUF is very similar to modeling an Arbiter PUF, the number of bits required to model a k-sum PUF was assumed to be equal to the number of challenge-and responses needed to model an Arbiter PUF. Based on [17] $N_{CRP} \approx \frac{1}{2}\frac{k+1}{\epsilon}$ challenge and response pairs are needed to model a k-stage Arbiter PUF with an error rate of ϵ. The advantage of an attacker with $k+1$ challenge and response pairs derived from a k-stage Arbiter PUF is then assumed not to be better than guessing. To estimate the maximum information an attacker can learn from a syndrome about the PUF parameters the notion of leaked bits is used in [22]. The amount of information leaked by a syndrome S about a PUF model M is given by

$$LB(S) = I(S; M^\infty) = H(S) - H(S|M^\infty)$$

where S and M^∞ are random variables representing the syndrome word and the perfect prediction of the PUF model, respectively, and $I(Y; X) = H(Y) - H(Y|X)$ a measure for mutual information. To decrease the leaked bits for IBS a new technique called syndrome distribution shaping was introduced which uses a clobber probability p. The clobber probability defines a probability of soft decision values $R_i \in R$ to be ignored during the indexing process $Enc^B(R)$. According to [22], an increasing clobber rate increases $H(S|M^\infty)$ and hence decreases the number of leaked bits per syndrome. For details regarding the notion of leaked bits and a more formal treatment of the subject we would like to refer to [22].

By adding syndrome distribution shaping to IBS there are three parameters to choose: the syndrome size s, the clobber rate p and the number of secret bits $|B|$ that are decoded from a single k-sum PUF. Yu *et al.* propose the following parameters $s = 6$ and $p = 5/8$ for the use with a 64-bit k-sum PUF and show that 2.45 bits are leaked per syndrome for these parameters. Since Yu *et al.* assume that an attacker can not do better than guessing if just $k + 1$ bits are leaked, at most $\lfloor 65/2.45 \rfloor = 26$ bits can be generated using the same k-sum PUF. Based on their findings for $|B| = 26$ they define this as *Secure Construct 2*. For enhanced security Yu *et al.* propose to use an increased security margin of $1/2$ so that only half as many bits are leaked by a different construction. This alternative construction denoted as *Secure Construct 1* obtains only $|B| = 26/2 = 13$ bits from a single k-sum PUF. In the next section we show that these claims are actually too strong and how both of these constructs can still be successfully attacked.

III. Analysis

We now introduce two different attacks on the k-sum PUF with IBS. All experiments are conducted based on simulated k-sum PUFs in Matlab. A k-sum PUF is modeled by randomly choosing the frequency differences ΔO_i for the k RO pairs from an unbiased Gaussian distribution. This is in line with the general assumption that the process variations of ROs follow a Gaussian distribution (cf. [14]). This way of modeling a k-sum PUF can be seen as a best-case scenario from a designers perspective, since in our simulated k-sum PUFs all individual RO stages are guaranteed to be independently and identically distributed. Each key was chosen randomly and the individual challenges were also randomly generated.

Based on the amount of secret bits retrieved from a k-sum PUF instance, two different approaches are used in this work. For $|B| \leq 16$ an approach based on a *hamming distance characterization* is used. For $|B| > 16$, a *machine learning* approach is used since the hamming distance approach would become too costly in terms of computation.

A. Hamming Distance Characterization

A k-sum PUF response represents the total frequency difference of its individual RO stages. Each RO stage i generates either a negative or positive difference value depending on the challenge bit C_i. For noise-free ROs, there exist two challenges that generate a maximum difference over all frequencies, more precisely the challenge that adds all positive differences C^{max} and C^{min} that adds just negative differences. These two challenges are the binary complement of each other. The Index-Based Syndrome encoder selects the value which is next to the maximum frequency difference from a set of soft decision values. Therefore, the challenge of the indexed soft decision value will be closer related to either C^{min} or C^{max}, depending on the key bit B. This implies that indexed challenges that will be decoded to the same key bit value are more related to each other than to challenges that will be decoded to the complementary key bit value. As metric to quantify the relations between two challenges C^i and C^j we used the hamming distance (HD). Without knowledge about C^{min} and C^{max} a split of a challenge set into two groups can be done based on the hamming distance characterization of the challenges.

To run the analysis, we generated the pairwise hamming distance from a set of challenges that were indexed by our IBS model. A key bit hypothesis matrix of size $2^{|B|} \times |B|$ is generated while each row represents a unique hypothesis of $|B|$ key bits. As previously mentioned, we expect that challenges which result in the same soft decision bit, respectively the same key bit value B show a small hamming distance. So for each hypothesis, the pairwise hamming distances of challenges generating the same guessed key bit ($B_i = B_j$) are summed up and form a fitness value f. For example, for $|B| = 5$ and key hypothesis $B_{1,\dots,5} = \{1, 1, 0, 1, 0\}$ the fitness value is

$$\begin{aligned} f &= f_{ones} + f_{zeros} \\ &= HD(C^1, C^2) + HD(C^1, C^4) + HD(C^2, C^4) + HD(C^3, C^5) \end{aligned}$$

The fitness value of a hypothesis is directly related to the probability that this hypothesis is the correct key. The lower the fitness value f of a hypothesis is, the more likely the hypothesis is the correct one. So a ranking is generated by ascend sorting the hypotheses based on the fitness value.

The clobber probability introduced in [22] randomly removes soft decision values from the set R and their corresponding challenges so that the clobbered soft decision values are not considered in the indexing process. The intention of the clobber probability is to increase the complexity to run machine learning algorithms which selects the most probable values from a sorted soft decision set R. For more details, see [22]. The hamming distance metric instead, rank the key output hypotheses by analyzing the challenges. So the clobber probability do not directly diminish the ranking process. Increasing the clobber probability is in this analysis model equivalent with choosing a smaller soft decision set R of size q.

The security analysis given in [22] assumes an attacker running machine-learning techniques to model the PUF instance. The hamming distance characterization instead uses the strong relation between PUF output and challenge without modeling the PUF instance. Therefore, the reported security proof is not applicable for the hamming distance characterization.

1) Results: To quantify the results of the hamming distance characterization we used the well known definition of the Shannon Entropy

$$H = -\sum_{i=1}^{2^{|B|}} p_i \log_2 p_i.$$

Let p_i be the probability that the correct key bit hypothesis is placed at position i in the ranked list generated by the hamming distance characterization. The Probability Density Function (PDF) we used for the entropy estimation is empirically generated from 10^6 runs and is therefore just an approximation. Increasing the runs had no influence on the PDF, so we assumed that a sufficient accurate PDF is generated with a set 10^6 runs. The PDF for $|B| = 13$, $s = 6$, and $p = 5/8$

is given in Figure 3. The results of the entropy estimation are given in Table I and Figure 2 for different parameters.

$\lvert B \rvert$	Bit Entropy H				
	s=6 p=0/8	s=5 p=0/8	s=6 p=5/8	s=4 p=0/8	no IBS
16	0.104	0.218	0.311	0.458	0.965
13	0.152	0.300	0.397	0.536	0.973
10	0.242	0.413	0.506	0.629	0.980
8	0.337	0.510	0.595	0.700	0.985
6	0.473	0.628	0.697	0.780	0.989
4	0.660	0.773	0.819	0.871	0.994
2	0.898	0.936	0.951	0.966	0.999

TABLE I: Resulting bit entropy after applying the hamming distance characteristic for a 64-bit k-sum PUF and different syndrome sizes and secret bit length. To compute the PDF needed for the entropy computation, keys from one million randomly generated PUFs and challenges were used. In the most right column the hamming distance metric is applied directly on the binary output bits of the k-sum PUFs without using IBS.

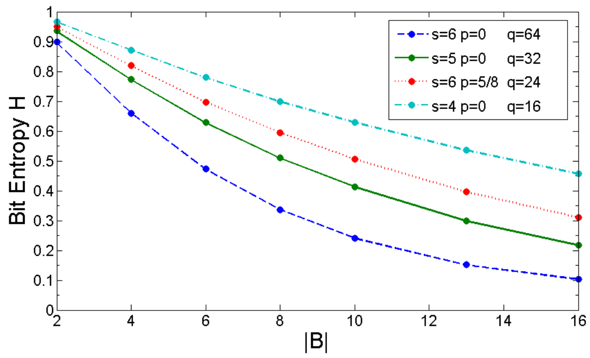

Fig. 2: Resulting bit entropy after applying the hamming distance characteristic for a 64-bit k-sum PUF and different syndrome sizes and secret bit length. Values are based on one million randomly generated PUFs and challenges.

Note that the entropy given in Table I is the bit entropy of an IBS-PUF instance for the noted parameters. A 128 bit key generated with $|B| = 13$, $s = 6$ and $p = 5/8$ from $\lceil 128/13 \rceil = 10$ different k-sum PUFs, would have a entropy of only $0.3971 \cdot 130 = 51.62$ bits instead of 130 bits.

B. Machine Learning

Modeling a k-sum PUF is very similar like modeling an Arbiter PUF. Hence the same machine learning attacks are applicable to approximate the frequencies of the k-sum PUF. The main difference between modeling a k-sum PUF and an Arbiter PUF with 64 stages is that for the k-sum PUF 64 parameters need to be determined, while a 64-stage Arbiter PUF is modeled with 65 parameters, see [17] for details on machine learning attacks on Arbiter PUFs. Moreover, the machine learning attacks presented in [17] are in need for challenge and response bits. In the case of IBS, an attacker does not have access to direct challenge and responses and hence a different approach is needed. For our machine learning attack we therefore use Evolution Strategies (ES). In comparison to other machine learning attacks, ES just requires a PUF that can be parametrized and the existence of a cost function

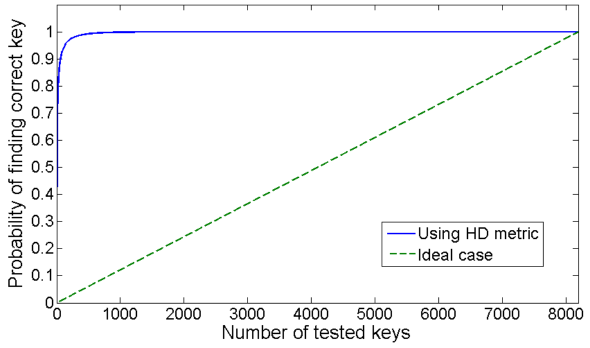

Fig. 3: Resulting probability mass function after applying the HD characterization for $|B|$=13, $s = 6$, $p = 5/8$. The dotted line represents the ideal case of full 13 bit entropy for comparison.

that returns parameters which are closest to the real PUF from a set of parameters. In the case of the k-sum PUF the PUF parameters to be modeled are the frequency differences $\vec{\Delta O} = \Delta O_0, \Delta O_1, .., \Delta O_{k-1}$.

Typically, the cost function is simply the model accuracy of a training set that is achieved by a given $\vec{\Delta O}$. But since we do not know the response bits in the case of IBS, we need a different cost function to evaluate the fitness of PUF parameters. The main idea of our cost function is that a syndrome points either to a maximum or minimum within a set of q challenges. Hence, to test how good a given PUF model performs, we compute all q PUF responses for the PUF parameter under test. We expect that for a suitable PUF model, the syndrome points to either a response that is close the maximum or to the minimum of this set. Hence, the cost value is simply the number of responses within this set of q responses that either have a larger response value or the number of challenges that have a smaller response value, whichever is smaller. Assuming a noise-free environment and no clobber rate, we expect this difference to be 0 for a perfect PUF model, since each syndrome always point to a maximum or minimum. If a clobber rate is used or the response is noisy, even the perfect PUF model would not necessarily have a cost of 0. However, ES is quite resistant to noise so that the general assumption, that the syndrome should point to either a value close to the maximum or minimum, still holds.

Roughly speaking, the idea of ES machine learning is to randomly choose μ PUF parameters $\vec{\Delta O_i}$ and then test which of these guesses performs best using the cost function. The guesses that perform best are kept as *parents* for the next *generation*, while the other guesses are discarded. In the next generation, the parameters of the best guesses from the previous generation are then randomly *mutated*, i.e., a Gaussian random variable is added to the parameter vector $\vec{\Delta O}$. Then the best guesses are again kept for the next generation while the others are discarded. This way the guesses become gradually more and more accurate and the guesses parameter vectors $\vec{\Delta O}$ will resemble the real PUF more and more. There are many different ways how the mutation in each generation is controlled. In this work we used the very popular Covariance Matrix Adaptation (CMA-ES) machine learning algorithm [8].

ES machine learning attacks are non-deterministic which means the attacks can yield different results for the same input set. Not every run finds an optimal solution and hence the machine learning algorithm needs to be run several times on the same training set. Sometimes the machine learning attack might still not find a PUF model that is accurate enough to predict all $|B|$ response bits correctly. Furthermore, there will always be one bit of uncertainty left, since we do not know the sign of the response output. Hence, a successful attack will in the best case return two equally likely key candidates, the correct key and the inverse of this key. Note that the direct output of the ES are the PUF parameters $\Delta\vec{O}_i$ and not the key bits. Hence, in a post-processing step these PUF parameters are used to compute the key candidates. Only if two ES runs yield PUF parameters that are very similar, this will result in the same key candidate. Some PUF parameters, however, will lead to different key candidates that then need to be exhaustively tested. A more efficient approach is to sort the key candidates according to the fitness value of the final ES run since the fitness value directly indicates the success of the run. It would also be possible to sort the key candidates according to the hamming distance characterization as previously discussed. The following results, however, were obtained after sorting the final fitness values.

1) Results: We performed a CMA-ES machine learning attack on a 64-stage k-sum PUF with a syndrome size of $s = 6$ and a clobber rate of $p = 5/8$ for different number of key bits $|B|$. The results of the attacks are summarized in Table II. In case of a key size of 63 as used in [23], the attack is very reliably and in all of the 200 tested runs the correct key was within the first 6 proposed key candidates. In 92% the correct key was even within the first two key candidates. Figure 4 shows the distribution of key candidates for 26 key bits, which is equal to *Secure Construct 2* from [22]. Only in 1% of the test runs the machine learning algorithm had not found a PUF model with an accuracy high enough to predict the 26 key bits. For this analysis 750 runs of the CMA-ES on 400 randomly generated PUFs and challenges were used. If the machine learning found a correct PUF model the correct key was within the first 128 key candidates and in average the correct key was at position 6.12.

Even for 13 inputs a machine learning attack is possible, although it does not perform as good as the hamming distance characterization. It is interesting to note that while syndrome distribution shaping (i.e. the clobber rate) had no impact on the hamming distance metric, it increases resistance against our machine learning attack due to the noise added to the cost function. However, the impact of the clobber rate is not as high as suggested in [22]. This can be observed in Table II which shows results of CMA-ES attacks on a 13-bit key using different clobber rates p that result in the same number of k-sum PUF responses $q = 2^s \cdot (1 - p)$. For a clobber rate of $p = 1/2$, the success rate was with 97% the same as for the case that a clobber rate of $p = 0$ was used. Only for large clobber rate of $p = 3/4$ did the success rate decrease to 93.75%. However, this decrease is by far not as significant as [22] suggests. In particular, according to the analysis in [22], $s = 6$, $p = 5/8$ should leak less information than $s = 4$ $p = 0$. But our results show that the impact of q is more important than p and hence attacking $s = 6$, $p = 5/8$ is significantly easier than $s = 4$ $p = 0$.

It is also interesting to note that it is possible to use a cost function based on the hamming distance characterization instead of the ordering. However, in an experiment this cost function performed worse than the previously applied cost function based on the ordering. Still the hamming distance characterization only depends on q and is independent of the clobber rate so that the clobber rate would have no effect on such a machine learning attack.

$\#B$	s	p	q	Success rate	Average key position
63	6	5/8	24	100%	1.55
26	6	5/8	24	99%	6.12
13	6	5/8	24	99.5%	56.17
13	4	0	16	97%	125.78
13	5	1/2	16	97.25%	120.94
13	6	3/4	16	93.75%	99.05

TABLE II: Results of CMA-ES attacks for different IBS parameters. For this experiment, 400 independent trials were performed and each trial consisted of 750 CMA-ES runs. The only exception was $|B| = 63$ for which only 200 trials and 150 runs was performed. Success rate is the percentage of keys for which the correct key was among the key candidates produced by the ES machine learning. Average key position shows the number of keys that would need to be tested until the correct key is found among the key candidates. Note that only trials in which the correct key was among the key candidates were considered.

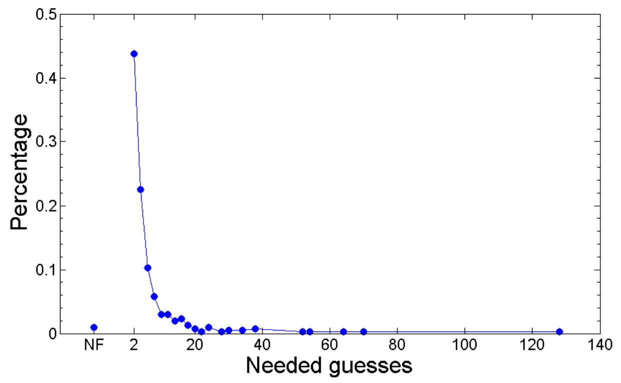

Fig. 4: Result of CMA-ES attacks on an IBS k-sum PUF with $|B| = 26$, $s = 6$, $p = 5/8$. 400 randomly generated k-sum PUFs and keys were used in this analysis. The X-axis depicts the number of key candidates that were tested until the correct key was found. "NF" depicts the number of PUFs for which the correct key was not among the key candidates, i.e., the percentage of unsuccessful CMA-ES attacks.

IV. Discussion and Conclusion

In this paper we presented two different attacks on the helper data of the IBS error correction when used in conjunction with a k-sum PUFs. The secret keys generated on such a construction have a significantly lower entropy than previously assumed. In particular, if 63 secret bits are generated using a single k-sum PUF, hardly any entropy remains. Even if only 13 secret bits are generated as proposed by [22], the bit entropy is significantly reduced to 0.39. This low bit entropy is particularly critical when being fed into a two-stage error correction based on linear codes, since then the remaining entropy might result in zero [10]. In conclusion, our results are in extreme contrast to the security analysis presented at CHES in [22]. This discrepancy between our work and [22] is due to the security analysis assuming that an attacker tries to approximate the PUF parameters using machine learning techniques. However, we showed that it is possible to derive a meaningful metric for secret hypothesis based on the hamming distance of the challenges. This alternative characterization can reduce the number of needed guesses and hence the bit entropy significantly. It is important to note that this attack does not predict the internal PUF parameters and hence the security analysis in [22] is not applicable to this attack. Instead of trying to approximate the PUF parameters, the attack predicts PUF responses solely based on the challenges. As a second contribution of this work, we also show in Section III-B that machine learning attacks that try to model the internal PUF parameters are also possible. According to our findings, we can state that proposed constructs in [22] cannot successfully defeat these attacks. We would like to remark that the use of Complementary IBS [9] with a k-sum PUF would only make matters worse. For each secret bit two indices are revealed that point to both the challenge corresponding to the maximum as well as minimum frequency. Hence, the attacker has an easier time to sort the key candidates according to the hamming distance metric.

Please note that the results presented in this paper also apply to constructs that use k-sum PUF with a different type of error correction. As can be seen from Table I, hamming distance characterization can also directly be applied to challenges without IBS or any other error correction. In this case the entropy reduction is considerably smaller than with IBS being used, but still noticeable. The analysis shows that the hamming distance characterization is not limited to IBS and for example it should also be possible to apply this technique to fuzzy extractors based on linear codes. However, such an analysis was out of the scope of this paper and interesting future work. Furthermore it is important to note that other PUF constructs suffer from the same weakness as the k-sum PUF. For example, the presented results can be directly applied one to one on Arbiter PUFs or other constructs such as the current-based PUF [15] or the recently introduced Bitline PUF [1]. These constructs have in common that they can be described with an additive linear model. In these PUFs, two responses corresponding to challenges that only differ in a few bits have a higher chance of being equal than if their challenges would have a high hamming distance. Hence, for these constructs the introduced hamming distance metric can be used to order keys according to their probability and hence reduce the key entropy.

Finally, we can state that error correction is still a major challenge for security in PUF-based key generation. Still we would like to emphasize that our results do not mean that IBS is in general a bad choice. The main issue with IBS is that the loss in entropy is directly related to the quality of the PUF to match the i.i.d. assumption. If a PUF matches the i.i.d. assumption, IBS can be a promising error correction mechanism.

References

[1] Bitline PUF: Building Native Challenge-Response PUF Capability into Any SRAM. In *CHES 2014*, volume 8731 of *LNCS*, pages 510–526. Springer, 2014.

[2] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient Helper Data Key Extractor on FPGAs. In *CHES 2008*, volume 5154 of *LNCS*, pages 181–197. Springer, 2008.

[3] J. Delvaux and I. Verbauwhede. Attacking PUF-Based Pattern Matching Key Generators via Helper Data Manipulation. In *CT-RSA 2014*, volume 8366 of *LNCS*, pages 106–131. Springer, 2014.

[4] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, 2004.

[5] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon Physical Random Functions. In *CCS 2002*, pages 148–160. ACM, 2002.

[6] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In *CHES 2007*, volume 4727, pages 63–80. Springer, 2007.

[7] T. Güneysu. Using Data Contention in Dual-ported Memories for Security Applications. *Signal Processing Systems*, 67(1):15–29, 2012.

[8] N. Hansen. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer, 2006.

[9] M. Hiller, D. Merli, F. Stumpf, and G. Sigl. Complementary IBS: Application Specific Error Correction for PUFs. In *HOST 2012*, pages 1–6. IEEE, 2012.

[10] P. Koeberl, J. Li, A. Rajan, and W. Wu. Entropy Loss in PUF-Based Key Generation Schemes: The Repetition Code Pitfall. In *HOST 2014*, pages 44–49. IEEE, 2014.

[11] R. Maes, P. Tuyls, and I. Verbauwhede. Intrinsic PUFs from Flip-Flops on Reconfigurable Devices. In *WISSec 2008*, 2008.

[12] R. Maes, P. Tuyls, and I. Verbauwhede. Low-Overhead Implementation of a Soft Decision Helper Data Algorithm for SRAM PUFs. In *CHES 2009*, volume 5747 of *LNCS*, pages 332–347. Springer, 2009.

[13] R. Maes, A. Van Herrewege, and I. Verbauwhede. PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator. In *CHES 2012*, volume 7428 of *LNCS*, pages 302–319. Springer, 2012.

[14] A. Maiti, J. Casarona, L. McHale, and P. Schaumont. A Large Scale Characterization of RO-PUF. In *HOST 2010*, pages 94–99. IEEE, 2010.

[15] M. Majzoobi, G. Ghiaasi, F. Koushanfar, and S. Nassif. Ultra-low power current-based PUF. In *ISCAS 2011*, pages 2071–2074. IEEE, 2011.

[16] Z. Paral and S. Devadas. Reliable and Efficient PUF-Based Key Generation Using Pattern Matching. In *HOST 2011*, pages 128–133. IEEE, 2011.

[17] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *CCS 2010*, pages 237–249. ACM, 2010.

[18] P. Simons, E. van der Sluis, and V. van der Leest. Buskeeper PUFs, a Promising Alternative to D Flip-Flop PUFs. In *HOST 2012*, pages 7–12. IEEE, 2012.

[19] V. van der Leest, B. Preneel, and E. van der Sluis. Soft Decision Error Correction for Compact Memory-Based PUFs Using a Single Enrollment. In *CHES 2012*, volume 7428 of *LNCS*, pages 268–282. Springer, 2012.

[20] M. M. Yu and S. Devadas. Recombination of Physical Unclonable Functions. In *GOMATech 2010*. United States. Dept. of Defense, 2010.

[21] M. M. Yu and S. Devadas. Secure and Robust Error Correction for Physical Unclonable Functions. *IEEE Design & Test of Computers*, 27(1):48–65, 2010.

[22] M. M. Yu, D. M'Raïhi, R. Sowell, and S. Devadas. Lightweight and Secure PUF Key Storage Using Limits of Machine Learning. In *CHES 2011*, volume 6917, pages 358–373. Springer, 2011.

[23] M. M. Yu, R. Sowell, A. Singh, D. M'Raïhi, and S. Devadas. Performance Metrics and Empirical Results of a PUF Cryptographic Key Generation ASIC. In *HOST 2012*, pages 108–115. IEEE, 2012.