Key-recovery Attacks on Various RO PUF Constructions via Helper Data Manipulation

Jeroen Delvaux and Ingrid Verbauwhede
ESAT/COSIC and iMinds, KU Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email: {firstname.lastname}@esat.kuleuven.be

Abstract—**Physically Unclonable Functions (PUFs) are security primitives that exploit the unique manufacturing variations of an integrated circuit (IC). They are mainly used to generate secret keys. Ring oscillator (RO) PUFs are among the most widely researched PUFs. In this work, we claim various RO PUF constructions to be vulnerable against manipulation of their public helper data. Partial/full key-recovery is a threat for the following constructions, in chronological order. (1) Temperature-aware cooperative RO PUFs, proposed at HOST 2009. (2) The sequential pairing algorithm, proposed at HOST 2010. (3) Group-based RO PUFs, proposed at DATE 2013. (4) Or more general, all entropy distiller constructions proposed at DAC 2013.**

I. Introduction

With the ubiquity of integrated circuits (ICs) in our everyday lives, cryptographic algorithms have become an important building block. Hereby, one heavily relies on the ability to store secret information. Traditionally, binary keys are stored in programmable on-chip non-volatile memory (NVM): EEPROM and its successor Flash are the main technologies. However, an attacker can easily gain physical access to the IC. Hardware attacks, either invasive or noninvasive, are thus a significant threat. The NVM approach tends to be vulnerable [5], as the key is stored permanently in electrical form. Additional circuitry to protect the key is usually complemented by practical drawbacks: costly, bulky, battery powered, ...

Physically Unclonable Functions (PUFs) have been proposed as a more secure alternative. Silicon PUFs quantify the unique manufacturing variability of nanoscale structures. The secret is stored in intrinsic physical features of an IC, resulting in some remarkable security advantages. First, PUFs are often assumed to be resistant against invasive attacks. One can argue that invasion damages the physical structure of an IC. Second, keys are inherently unique for each manufactured sample of an IC and there is no need to explicitly program them. Third, the key is only generated and stored in on-chip volatile memory (VM) whenever key-dependent operations have to be performed, as such posing limits on the attacker's time frame.

Ring oscillator (RO) PUFs are very popular, inter alia because they can be implemented on FPGA. We describe their high-level architecture in section II. Unfortunately, PUF bits by themselves do not result in reproducible and uniformly distributed keys, as discussed in section III. Helper data constructions are therefore required: we describe several proposals in sections IV and V, applicable to RO PUFs in particular. We claim them to be vulnerable against statistical attacks, hereby manipulating the public helper data. Partial or even full key recovery might be possible, as discussed in section VI. An extensive reflection of our findings is given in section VII. Section VIII concludes the work.

II. Ring Oscillator PUFs

RO PUFs quantify the manufacturing variability of identically laid-out oscillators. Each RO, consisting of an odd number of inverters, will have a unique frequency f. Frequencies are typically measured by counting rising or falling edges on a wire connecting two subsequent inverters. Figure 1 shows the PUF architecture as originally proposed in [6]. One can distinguish four components: an array of N ROs, multiplexers to access individual ROs, counters providing a frequency measurement and a comparator.

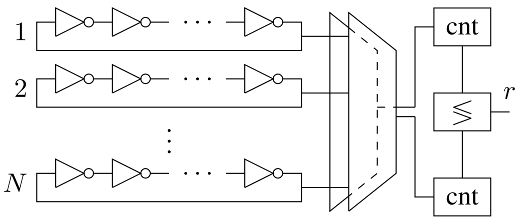

Fig. 1. RO PUF as originally proposed.

A pairwise frequency comparison ($\Delta f \lessgtr 0$) generates a single response bit r. There are $N(N-1)/2$ pairwise comparisons, although their response bits are interdependent. Consider the following minimal example, given three ROs: $RO_A.f < RO_B.f$ and $RO_B.f < RO_C.f$ implicates $RO_A.f < RO_C.f$. The total PUF entropy is only $log_2(N!)$ bit as there are $N!$ ways to sort the frequency values. We hereby assume the ideal case, with all permutations equally likely.

For convenience, the ring oscillators are typically laid-out as a two-dimensional array on the IC. Without loss of generality, we still label each RO with a univariate index $i \in [1\ N]$. The multiplexer-counter-comparator architecture might greatly vary. Consider the following two extreme cases, for example: a dedicated counter per RO and a single counter accessing all ROs via a giant multiplexer.

III. Motivation for Helper Data Constructions

Unfortunately, PUF response bits are not directly usable as a secret key because of two issues: they are not perfectly reproducible and non-uniformly distributed. We list the root causes and provide examples for RO PUFs in particular. Helper data constructions are required to resolve the former issues. Hereby, public helper bits are generated during a one-time post-manufacturing enrollment phase. They are stored in (off-chip) NVM and assist with every key reconstruction. However: the secrecy of the response bits should be preserved.

A. *Reliability*

Noise in CMOS transistors (and interconnect) is the main responsible for the reliability issue. We consider noise as an unavoidable random time-dependent phenomenon. Instability of the environment, mostly defined by the IC supply voltage and the outside temperature, worsens the problem. Note that RO frequencies increase with both increasing supply voltage and decreasing temperature. The larger the nominal frequency discrepancy $|\Delta f|$ for a given pairwise comparison, the more reliable the corresponding response bit.

B. *Entropy*

PUF response bits are non-uniformly distributed, reducing the entropy of the key. Bias is a major concern hereby: the probability of a bit to be '1' (or '0') might not be equal to 50%. Correlations between the bits are another symptom. Asymmetries in the PUF lay-out are one potential root cause. Systematic manufacturing variations, which are spatially correlated, form another root cause. As illustrated in figure 2, only random variations are desired. Furthermore, the occasional occurrence of $\Delta f = 0$ (counter values are discrete) introduces bias given that either '1' or '0' has to be returned.

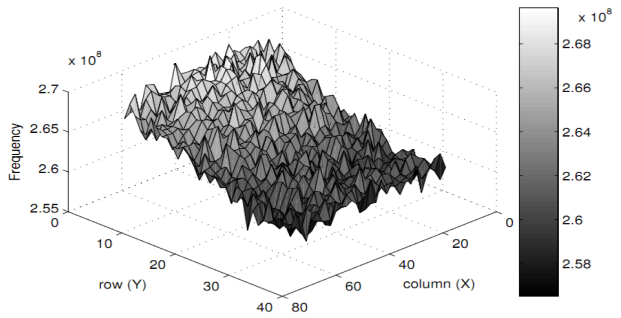

Fig. 2. Example frequency topology of a RO array [4]: $f(x, y)$, for an IC-aligned xy-plane. The linear trend corresponds with systematic variability. Only the random surface roughness is desired.

IV. RO Pair Selection

There is a variety of methods to select pairs from a RO array. Their goal is to output many high-entropy bits, possibly with an incentive towards reliability. We now discuss four approaches, in order of increasing complexity.

A. *Chain of Neighbors*

Pairing neighboring ROs is perhaps the most intuitive approach. The reduced impact of spatial correlations is the main advantage [3]. For disjunct pairs, $\lfloor N/2 \rfloor$ independent bits can be generated. By sharing ROs across pairs, up to $N-1$ independent bits can be generated.

B. *1-out-of-k Masking*

A 1-out-of-k masking scheme [6] is applied to a fixed set of RO pairs, such as a chain of neighbors. The pairs are partitioned into groups, each containing k pairs. During enrollment, the pair which maximizes $|\Delta f|$ is selected within each group, favoring reliability as such. The corresponding indices are saved in public helper NVM. Parameter k represents a trade-off between reliability and efficiency.

C. *Sequential Pairing Algorithm*

The sequential pairing algorithm [8] selects up to $\lfloor N/2 \rfloor$ disjunct pairs. The frequency discrepancy $|\Delta f|$ of every selected pair does exceed a given threshold Δf_{th}. Pairing information is again stored in public helper NVM. Algorithm 1 provides simplified pseudocode. In the original proposal, one requires frequency measurements at two environmental extremes.

Algorithm 1: Sequential Pairing (Simplified)

```
Input: Frequency measurements $RO_i.f$ with $i \in [1\ N]$
       Frequency discrepancy threshold $\Delta f_{th}$
Output: List of pairs $\{RO_i, RO_j\}$
Sort frequencies in descending order and store indices
as vector $\pi$: $RO_{\pi(1)}.f > RO_{\pi(2)}.f > \ldots > RO_{\pi(N)}.f$
$i \leftarrow 1$
for $j \leftarrow \lceil \frac{N}{2} \rceil + 1$ to $N$ do
    if $RO_{\pi(i)}.f - RO_{\pi(j)}.f > \Delta f_{th}$ then
        Create pair $\{RO_{\pi(i)}, RO_{\pi(j)}\}$
        $i \leftarrow i + 1$
```

D. *Temperature-aware Cooperative*

Temperature-aware cooperative RO PUFs [7] operate within a user-defined temperature range: $T \in [T_{min},\ T_{max}]$. An on-chip temperature sensor is assumed to be available, posing limits on the applicability. Furthermore, RO frequencies are assumed to be linearly dependent on the temperature.

Neighboring ROs are paired, without overlap, leading to a total of $\lfloor N/2 \rfloor$ pairs. A frequency discrepancy threshold Δf_{th} is employed to assess their reliability. Pairs are classified in three groups, as illustrated on figure 3. Good pairs obey $|\Delta f(T)| > \Delta f_{th}$ within the whole operating range: they generate one reliable bit each. Bad pairs obey $|\Delta f(T)| \leq \Delta f_{th}$ within the whole operating range: they are discarded. Some pairs are stable except for an interval $[T_l,\ T_h]$ around their crossover point: they cooperate to generate reliable bits, assisted by public helper data.

For every cooperating pair, one does store the values of T_l and T_h in public helper NVM. Outside the crossover interval, no help is required. Although one has to compensate for the crossover: the response bit is inverted if $T > T_h$. Within the crossover interval, one does rely on another cooperating pair with a nonintersecting crossover region. Its index is stored in public helper NVM as well.

Pairs cooperate in a masked manner, to prevent leakage of their response bits. Consider a first cooperating pair, requesting help and having response bit r_{c1}. A masking response bit r_{g1},

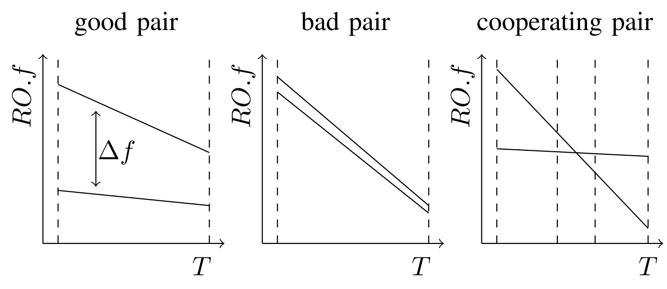

Fig. 3. Temperature-aware cooperative RO PUF: classification of RO pairs. The outer dashed lines represent the operating range: $[T_{min},\ T_{max}]$.

originating from a corresponding good pair, is assigned. A second cooperating pair, providing help and having response bit r_{ci}, should satisfy the following constraint: $r_{c1} \oplus r_{g1} = r_{ci}$. Note that r_{g1} and r_{ci} are both stable within the crossover interval of the first cooperating pair, allowing for reconstruction.

However, we claim that the proposed masking scheme is not necessarily free from leakage. The second cooperating pair should be selected at random and hence not with a deterministic procedure that iterates over all candidates until the masking constraint is met. Otherwise, one exposes the following information for all non-selected candidates: $r_{cj} \neq r_{ci}$.

V. GROUP-BASED RO PUF

Group-based RO PUFs have first been introduced at HOST 2010 [8]. As the initial design had several shortcomings, the authors redefined their construction at DATE 2013 [9]. For ease of understanding, we make abstraction of the gradual development. In traditional designs, ROs are paired to generate a response bit. The group-based approach is very different in this regard. ROs are partitioned into groups, with their size not limited to two anymore.

The so-called entropy distiller, the main novelty, has been introduced in parallel at DAC 2013 too [10], although with more experimental evidence. Its use is not limited to the group-based approach. Employment with the pair selection methods of section IV is a possibility as well. We will consider both use cases for our attacks.

The high-level architecture is represented by figure 4. We explicitly indicate the IC boundaries and interfaces to clarify an attacker's point of view. Like this, we also stress that all building blocks do require an on-chip implementation. The resulting key is stored in on-chip VM, for as long as needed. An application with key-dependent operations communicates with the user, either directly or indirectly. We now discuss the building blocks separately.

A. Entropy Distiller

The entropy distiller eliminates systematic manufacturing variations. They are modeled via polynomial regression on the two-dimensional RO frequency map $f(x, y)$. The residuals represent the desired random variations. An expression for the polynomial of degree p is given below. Experiments in [10] indicate $p = 2$ and $p = 3$ as good values, given an array of 16×32 ROs. Coefficients $\beta_{i,j}$ may be determined in a least mean squares manner. They are stored as public helper data. A subtraction procedure removes systematic variations for every regeneration of the key.

$$f(x, y) = \sum_{i=0}^{p} \sum_{j=0}^{i} \beta_{i,j} x^{i-j} y^{j}.$$

B. Grouping Algorithm

The grouping algorithm partitions the ROs into groups $G_1, G_2, \ldots$ The partitioning is strict: every RO is assigned to exactly one group. Within a group, every possible pair of ROs does exceed a frequency discrepancy threshold Δf_{th}, favoring reliability. Response bits will be extracted for each group independently. Algorithm 2 provides pseudocode for the grouping procedure. It optimizes the available entropy of $\sum_j log_2(|G_j|!)$ bits: having few large groups is more beneficial than having many small groups.

Algorithm 2: GROUPING

```
Input: Frequency measurements $RO_i.f$ with $i \in [1\ N]$
       Frequency discrepancy threshold $\Delta f_{th}$
Output: Group assignments $RO_i.group$
Sort frequencies in descending order and store indices
as vector $\pi$: $RO_{\pi(1)}.f > RO_{\pi(2)}.f > \ldots > RO_{\pi(N)}.f$
$RO_0.f \leftarrow \infty$
for $i \leftarrow 1$ to $N$ do
    $last(i) \leftarrow 0$
for $i \leftarrow 1$ to $N$ do
    $j \leftarrow 1$
    while $RO_{last(j)}.f - RO_{\pi(i)}.f \leq \Delta f_{th}$ do
        $j \leftarrow j + 1$
    $RO_{\pi(i)}.group \leftarrow j$
    $last(j) \leftarrow \pi(i)$
```

C. Kendall Coding

For every group G_j, there will be a particular order of the RO frequencies. A binary representation is required. Table I illustrates two schemes, assuming there are four ROs (A, B, C and D) in the group and hence $4! = 24$ possible orders. The most compact representations do require $\lceil log_2(|G_j|!) \rceil$ bit, as illustrated for the second column.

Order	Compact	Kendall
ABCD	00000	000000
ABDC	00001	000001
ACBD	00010	000100
ACDB	00011	000110
ADBC	00100	000011
ADCB	00101	000111
BACD	00110	100000
BADC	00111	100001
BCAD	01000	110000
BCDA	01001	111000
BDAC	01010	101001
BDCA	01011	111001

Order	Compact	Kendall
CABD	01100	010100
CADB	01101	010110
CBAD	01110	110100
CBDA	01111	111100
CDAB	10000	011110
CDBA	10001	111110
DABC	10010	001011
DACB	10011	001111
DBAC	10100	101011
DBCA	10101	111011
DCAB	10110	011111
DCBA	10111	111111

TABLE I. CODING OF OSCILLATOR FREQUENCY ORDER.

However, to facilitate the subsequent error-correction step, a non-minimum length coding scheme is proposed. One observes that errors mostly occur in form of a flip, e.g. BACD to BCAD. Using Kendall coding, one bit is generated for every

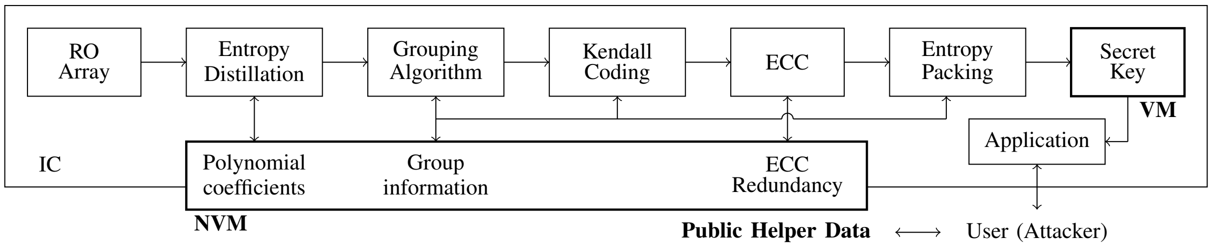

Fig. 4. Group-based RO PUF.

possible pair of ROs, requiring $|G_j|(|G_j|-1)/2$ bits in total. Error-correction requirements are relaxed in terms of error rate, as there is only one error per flip. Unfortunately, the workload increases quadratically with the group size $|G_j|$.

D. Error-Correcting Code

Incoming bits are clustered in blocks, which are all error-corrected independently. An error-correcting code (ECC) construction, able to correct t errors per block, is employed hereby. The first generated instance of each block is considered as a reference. Public helper data allows regenerated instances to be error-corrected, so that they are identical to the reference.

E. Entropy Packing

Kendall coding is noted to be non-uniform: many bit vectors are never used. To maintain entropy, conversion to a compact coding scheme (as in table I) is proposed. However, please note that the problem is only fixed partially, since $|G_j|!$ is not a power of two, given $|G_j| > 2$.

VI. Attacks via Helper Data Manipulation

Before discussing the specifics for each RO PUF construction, we first describe the common statistical framework of our attacks. PUF response bits are considered one by one (or in small groups). For each iteration, two or more hypotheses H_i provide a statement about the bits of concern, of which exactly one is correct. Every hypothesis corresponds with a specific manipulation of the public helper data. We exploit differences in key regeneration failure rate to assess their correctness.

The attacks on constructions (1) and (2) are generally applicable. Furthermore, we make no assumption about the application: an inability to reconstruct the key should affect the observable behavior of any useful application. The attacks on constructions (3) and (4) are case-specific, so we limit ourselves to an illustration. Furthermore, they rely on maliciously reprogrammed keys, assuming their reconstruction failures to be observable. However, this assumption is often (if not mostly) satisfied in practice: consider for instance all applications where some form of encrypted data is presented to the user.

For generality, we assume all constructions to employ an ECC as a final reliability measure, which is actually a common practice. The absence of an ECC can be considered as the degenerate case $t = 0$. For ease of explanation, we assume all bits to fit within a single ECC block. However, extension to multiple blocks is fairly straightforward. The probability density function (PDF) of the numbers of errors (at the ECC input) is particularly useful to quantify failure behavior. A binomial distribution might provide an accurate model for large blocks, although our attacks do not depend on this assumption.

Figure 5 provides an illustration, in case of two hypotheses H_0 and H_1. The nominal PDF serves as a reference: failures rarely occur in practice, assuming well-chosen ECC parameters. PDFs corresponding to helper data hypotheses are slightly shifted with respect to each other and hence distinguishable. The common offset originates from additional errors, intentionally and symmetrically introduced to accelerate the attack.

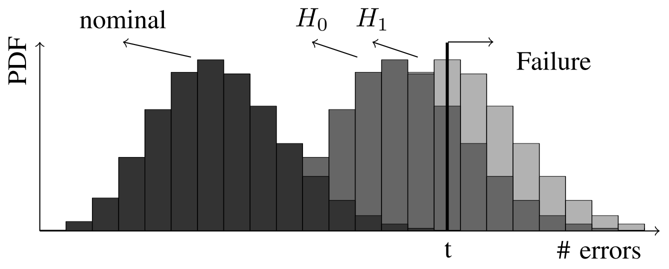

Fig. 5. Distinguishing hypotheses by observing key generation failure rates.

A. RO PUF with Sequential Pairing

Key recovery is fairly straightforward for the sequential pairing algorithm. Consider two RO pairs, resulting in response bits r_1 and r_2. We formulate two hypotheses as shown below. To distinguish them, we swap the order of the two pairs in public helper NVM. If H_0 is correct, the failure rate is not modified. However, if H_1 is correct, the failure rate does increase. Matching r_1 with all other response bits $r_2, r_3, \ldots r_{\lfloor N/2 \rfloor}$, only two possible values remain for the secret key. For the final decision, the performance of two corresponding sets of ECC helper data can be compared.

$$H_0 : r_1 = r_2. \quad H_1 : r_1 \neq r_2.$$

To accelerate the attack, more errors can be injected. For instance by swapping additional pairs, accordingly for the helper data of H_0 and H_1. Initially, the additional pairs can be chosen at random. After revealing some response bit relations however, one can select these pairs which will introduce a pair of erroneous bits for sure.

B. Temperature-aware cooperative RO PUF

An attacker can retrieve the response bit relations for all cooperating pairs of a temperature-aware cooperative RO

PUF. Consider a first cooperating pair, having response bit r_{c1} and requesting assistance. A second cooperating pair, having response bit r_{ci}, provides assistance. Consider another cooperating pair, having response bit r_{cj}. We formulate two hypotheses as shown below. Helper data is modified so that r_{cj} provides assistance, assuming reliability for the given temperature. If H_0 is correct, the failure rate is not modified. However, if H_1 is correct, the failure rate does increase. To accelerate the attack, more errors can be injected. For instance, via manipulation of the interval boundaries T_l and T_h.

$$H_0 : r_{ci} = r_{cj}. \quad H_1 : r_{ci} \neq r_{cj}.$$

C. Group-based RO PUF

An attacker can retrieve the full key for group-based RO PUFs, due to the ability to directly reprogram the key. By injecting steep polynomials into the entropy distiller, one can completely overshadow random frequency variations. The attacker's intended pattern can be superimposed onto the original spatial correlation map hereby. Via repartitioning of the groups, one can force bits to be either '1' or '0'. Also the remaining helper bits, which represent the ECC redundancy, are updated accordingly.

Consider the example of figure 6a, given an array of 4×10 ROs. The attacker injects strong gradients in the horizontal direction via a quadratic surface, as represented by the grayscale. We repartition the groups so that they all contain two ROs. The responses of G_2 to G_{20} are fully determined by the attacker. The response of G_1 however, is fully determined by random frequency variations. Note that one could employ a tilted plane as well, if G_1 would cover a single column only.

Suppose the ROs of G_1 to belong to the same group for the original partitioning too. Then their frequency order is of direct interest, as their corresponding response bit r_1 does influence a subkey. Consider the two hypotheses shown below. We compute a set of ECC helper data for both cases. The failure rate is expected to be lower for the correct hypothesis. Injecting additional errors is straightforward: we just compute the ECC redundancy given some inverted bit values.

$$H_0 : r_1 = 0. \quad H_1 : r_1 = 1.$$

D. Entropy Distiller with RO Pairing

Entropy distillers can be employed with all RO pairing schemes of section IV. We limit ourselves to the first two schemes, as there is a stand-alone attack for the latter two schemes. The attack methodology is similar as before. Figure 6b provides an illustration for 1-out-of-k masking, using $k = 5$ and applied to a non-overlapping chain of neighbors. Figure 6c provide an illustration for an overlapping chain of neighbors. It might be very difficult to isolate a single response bit, as illustrated for figure 6c: four response bits are fully determined by random variations. By increasing the number of hypotheses (2^4), one can still perform the attack however.

VII. Discussion

Former helper data constructions have all been proposed to solve reliability and entropy issues, as discussed in section III. However, a well-established standard solution is available as

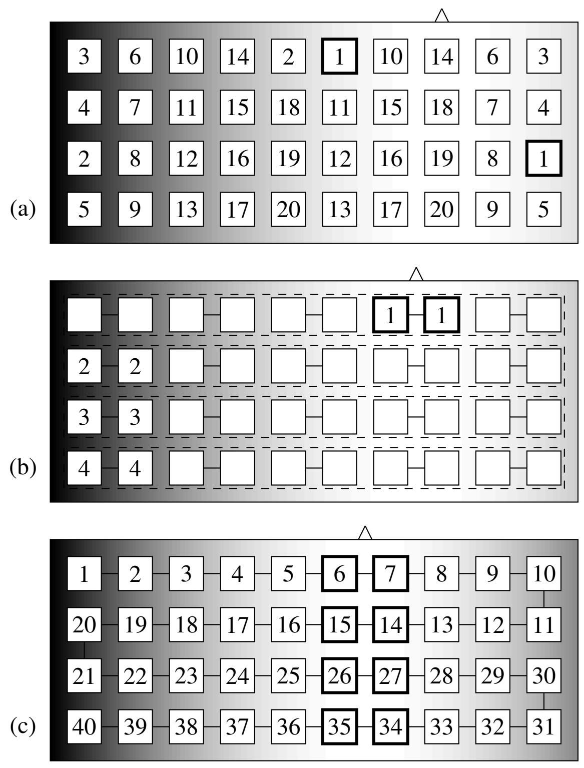

Fig. 6. Attacking entropy distiller constructions. The extremum of the quadratic patterns is marked with a triangle symbol. (a) Group based RO PUF. (b) 1-out-of-k masking. (c) neighbor pairing.

well: the so-called fuzzy extractor [2]. We briefly discuss its architecture. Afterwards, we argue why helper data should be considered as public always, implicating that an attacker has both read and write access. Finally, we formulate best practices for both the users and developers of helper data schemes.

A. Fuzzy Extractor

Fuzzy extractors can be used with any PUF architecture: their use is not limited to RO PUFs. Their definition is very generic, but typical implementations always rely on an ECC and a cryptographic hash function, as shown in figure 7. Latter constructions deal with reliability and entropy respectively, in a sequential manner.

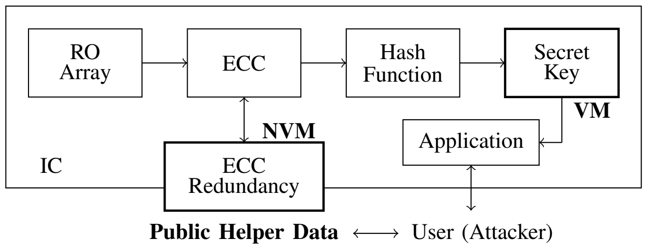

Fig. 7. RO PUF with fuzzy extractor.

B. Helper Data Considerations

In principle, programmable helper NVM can be implemented off-chip as well as on-chip. Therefore, we have drawn it on the IC boundary in figures 4 and 7. In the off-chip case, an attacker has full control: reading and modifying data is straightforward via the interface. However, also in the on-chip case, memory contents should be considered as public. Remember that PUFs have been proposed as a more secure alternative for on-chip NVM. Labelling on-chip helper NVM as private would undermine the need for PUFs. A motivated attacker, able to afford expensive equipment, can still obtain read/write access.

Furthermore, off-chip helper NVM is highly preferable because of three reasons. First, the overall efficiency might decrease in the opposite case: on-chip NVM remains while the PUF and its post-processing logic are extra. Second, on-chip NVM is expensive, as the standard CMOS manufacturing flow is insufficient. Third, typical FPGA platforms, for which RO PUFs are particularly interesting, do not contain on-chip NVM.

Secure and competitive PUF solutions do not pose read or write constraints on their helper data. For the fuzzy extractor, solid theory has been developed. The ECC constructions of [2] result in a rather limited entropy loss, which is compensated by the hash function, in addition to the initial non-uniformity. An extension of the architecture to counter manipulation attacks is described in [1]. The RO PUF constructions under attack [7]–[10] do consider leakage as a threat. Manipulation is never mentioned however, although their prototypes are all developed on FPGA platforms without on-chip NVM (Xilinx Spartan-3 and Xilinx XC4010XL).

C. Best Practices

We encourage the use of fuzzy extractors, as solid helper data theory has been developed. Newly proposed schemes should be compared to this common reference. If efficiency (area, speed, power/energy, memory) and/or security (quality of the key, helper data leakage and manipulation, side-channels etc.) is not expected to improve, there is little argumentation to promote their use.

However, a thorough comparison is generally lacking: we strive for better practices in this regard. Sometimes the fuzzy extractor's existence is not even mentioned, as for the group-based RO PUF and the entropy distiller proposal for instance. Note that the former construction actually borrows its ECC notion. We question the hardware efficiency of many proposals. Consider the collective overhead of group-based RO PUFs for instance. Or consider temperature-aware cooperative RO PUFs, having applicability issues. Besides a temperature sensor, one does require an extension of the IC manufacturing flow (see [7]: measuring RO frequencies, disconnecting bad RO pairs from power supply, etc.).

Furthermore, many proposals are rather vague about their use of helper data. The precise storage format, parsing procedure and/or sanity checks are typically not specified. Although subtle differences might impact security tremendously. We provide a few examples and strive again for better practices. For the sequential pairing algorithm, pairs of RO indices are stored. However, there is no recommendation to store a pair's indices in an either randomized or sorted order. Otherwise there is direct leakage of the full key. The re-use of ROs across pairs should also be prohibited somehow. For group-based PUFs, it is not clear whether grouping helper data is transferred three times or only once, with the former case offering more opportunities for an attacker.

VIII. Conclusion and Further Work

Like any other PUF, RO PUFs do require helper data constructions in order to generate reproducible and uniformly distributed keys. However, we showed various constructions to be vulnerable against manipulation of their public helper data. By observing system failure rates, an attacker can retrieve the key, or at least obtain some information about it. Actually, many more helper data constructions have been proposed in literature, not necessarily limited to the RO PUF. We do not claim to have studied them all and we advise to use them with great care. Instead, we encourage the use of fuzzy extractors, the well-established reference solution. We strive for better practices when proposing new helper data schemes. The following two items should be present: (1) an all-inclusive comparison with the reference solution and (2) a very precise specification of its helper data use.

Acknowledgment

This work was supported in part by the European Commission through the ICT programme under contract FP7-ICT-2011-317930 HINT. In addition, this work is supported by the Research Council of KU Leuven: GOA TENSE (GOA/11/007), by the Flemish Government through FWO G.0550.12N and the Hercules Foundation AKUL/11/19. Jeroen Delvaux is funded by IWT-Flanders grant no. 121552. The authors would like to thank Dries Schellekens (ESAT/COSIC, KU Leuven) for his valuable comments.

References

[1] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky and A. Smith, "Secure Remote Authentication Using Biometric Data," in *Eurocrypt*, pp. 147-163, May 2005.

[2] Y. Dodis, R. Ostrovsky, L. Reyzin and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97-139, Mar. 2008.

[3] A. Maiti and P. Schaumont, "Improving the Quality of a Physical Unclonable Function Using Configurable Ring Oscillators," in *Field Programmable Logic and Applications*, FPL 2009, pp. 703-707, Aug. 2009.

[4] P. Sedcole and P.Y.K. Cheung, "Within-die delay variability in 90nm fpgas and beyond, in *Field Programmable Technology*, FPT 2006, pp. 97-104, Dec. 2006.

[5] S. Skorobogatov, "Semi-invasive attacks - a new approach to hardware security analysis, Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory, Apr. 2005.

[6] G.E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference*, DAC 2007, pp. 9-14, Jun. 2007.

[7] C.E. Yin and G. Qu, "Temperature-aware cooperative ring oscillator PUF," in *Hardware-Oriented Security and Trust*, HOST 2009, pp. 36-42, Jul. 2009.

[8] C.E. Yin and G. Qu, "Lisa: Maximizing RO PUF's Secret Extraction," in *Hardware Oriented Security and Trust*, HOST 2010, pp. 100-105, Jun. 2010.

[9] C.E. Yin, G. Qu and Q. Zhou, "Design and implementation of a group-based RO PUF," in *Design, Automation & Test in Europe*, DATE 2013, pp. 416-421, Mar. 2013.

[10] C.E. Yin and G. Qu, "Improving PUF security with regression-based distiller," in *Design Automation Conference*, DAC 2013, pp. 1-6, May 2013.