PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon

Stefan Katzenbeisser[1], Ünal Kocabaş[1], Vladimir Rožić[3], Ahmad-Reza Sadeghi[2], Ingrid Verbauwhede[3], and Christian Wachsmann[1]

[1] Technische Universität Darmstadt (CASED), Germany
`katzenbeisser@seceng.informatik.tu-darmstadt.de`
`{unal.kocabas,christian.wachsmann}@trust.cased.de`

[2] Technische Universität Darmstadt and Fraunhofer SIT Darmstadt, Germany
`ahmad.sadeghi@trust.cased.de`

[3] KU Leuven, ESAT/COSIC, Leuven, Belgium
`{vladimir.rozic,ingrid.verbauwhede}@esat.kuleuven.be`

Abstract. Physically Unclonable Functions (PUFs) are an emerging technology and have been proposed as central building blocks in a variety of cryptographic protocols and security architectures. However, the security features of PUFs are still under investigation: Evaluation results in the literature are difficult to compare due to varying test conditions, different analysis methods and the fact that representative data sets are publicly unavailable.
In this paper, we present the first large-scale security analysis of ASIC implementations of the five most popular intrinsic electronic PUF types, including arbiter, ring oscillator, SRAM, flip-flop and latch PUFs. Our analysis is based on PUF data obtained at different operating conditions from 96 ASICs housing multiple PUF instances, which have been manufactured in TSMC 65 nm CMOS technology. In this context, we present an evaluation methodology and quantify the robustness and unpredictability properties of PUFs. Since all PUFs have been implemented in the same ASIC and analyzed with the same evaluation methodology, our results allow for the first time a fair comparison of their properties.

Keywords: Physically Unclonable Functions (PUFs), ASIC implementation, evaluation framework, unpredictability, robustness

1 Introduction

Physically Unclonable Functions (PUFs) are increasingly proposed as central building blocks in cryptographic protocols and security architectures. Among other uses, PUFs enable device identification and authentication [33,28], binding software to hardware platforms [7,14,4] and secure storage of cryptographic secrets [37,17]. Furthermore, PUFs can be integrated into cryptographic algorithms [2] and remote attestation protocols [29]. Today, PUF-based security products are already announced for the market, mainly targeting IP-protection, anti-counterfeiting and RFID applications [36,11].

PUFs typically exhibit a challenge/response behavior: When queried with a *challenge*, the PUF generates a random *response* that depends on the physical properties of the underlying PUF hardware. Since these properties are sensitive to typically varying operating conditions, such as ambient temperature and supply voltage, the PUF will always return a slightly different response each time it is stimulated. The most vital PUF properties for PUF-based security solutions are robustness and unpredictability [1]. Robustness requires that, when queried with the same challenge multiple times, the PUF should generate similar responses that differ only by a small error that can be corrected by an appropriate error correction mechanism. This is an essential requirement in PUF-based applications that must rely on the availability of data generated by or bound to the PUF and should be fulfilled under different operating conditions. Unpredictability guarantees that the adversary cannot efficiently compute the response of a PUF to an unknown challenge, even if he can adaptively obtain a certain number of other challenge/response pairs from the same and other PUF instances. With a PUF instance we denote one particular hardware implementation of a PUF design. Unpredictability is important in most PUF-based applications, such as authentication protocols, where the adversary could forge the authentication if he could predict the PUF response. Existing PUF-based security solutions typically rely on assumptions that have not been confirmed for all PUF types. For instance, most delay-based PUFs have been shown to be emulatable in software [26], which contradicts the unpredictability and unclonability properties. Hence, a systematic analysis of the security properties of real PUF implementations in hardware is fundamental for PUF-based security solutions.

In contrast to most cryptographic primitives, whose security can be related to well established (albeit unproven) assumptions, the security of PUFs relies on assumptions on physical properties and is still under investigation. The security properties of PUFs can either be evaluated theoretically, based on mathematical models of the underling physics [35,30], or experimentally by analyzing PUF implementations [10,34,8,9,16]. However, mathematical models never capture physical reality in its full extent, which means that the conclusions on PUF security drawn by this approach are naturally debatable. The main drawback of the experimental approach is its limited reproducibility and openness: Even though experimental results have been reported in literature for some PUF implementations, it is difficult to compare them due to varying test conditions and different analysis methods. Furthermore, raw PUF data is rarely available for subsequent research, which greatly hinders a fair comparison.

Our Goal and Contribution. We present the first large-scale security analysis of ASIC implementations of the five most popular electronic PUF types, including different delay-based PUFs (arbiter and ring oscillator PUFs) and different memory-based PUFs (SRAM, flip-flop and latch PUFs). Hereby, we focus on robustness and unpredictability, which are the most vital PUF properties in many security-critical applications. The ASICs have been manufactured in TSMC 65 nm CMOS technology within a multi-project wafer run and contain multiple implementations of the same PUF design. Our analysis is based on PUF data obtained from 96 ASICs at different temperatures, supply voltages and noise levels that correspond to the corner values typically tested for consumer-grade IT products. In this context, we developed an evaluation methodology for the empirical assessment of the robustness and unpredictability properties of PUFs. Since all PUFs have been implemented in the same ASIC and analyzed with the same methodology, our results allow for the first time a fair comparison of the robustness and unpredictability of these PUFs.

Our evaluation results show that all PUFs in the ASIC are sufficiently robust for practical applications. However, not all of them achieve the unpredictability property. In particular, the responses of arbiter PUFs have very low entropy, while the entropy of flip-flop and latch PUF responses are affected by temperature variations. In contrast, the ring oscillator and SRAM PUFs seem to achieve all desired properties of a PUF: Their challenge/response behavior hardly changes under different operating conditions and the entropy of their responses is quite high. Furthermore, the responses generated by different ring oscillator and SRAM PUF instances seem to be independent, which means that the adversary cannot predict the response of a PUF based the challenge/responses pairs of another PUF. However, the min-entropy, i.e., the minimum number of random bits observed in a response of the ring oscillator PUF is low, which means that some responses can be guessed with high probability.

Outline. We provide background information on PUFs in Section 2 and give an overview of the ASIC implementation of the analyzed PUFs in Section 3. We present our evaluation methodology in Section 4 and our analysis results in Section 5. Finally, we conclude in Section 6.

2 Background on PUFs

A Physically Unclonable Function (PUF) is a function that is embedded into a physical object, such as an integrated circuit [25,20]. When queried with a *challenge* x, the PUF generates a *response* y that depends on both x and the unique device-specific physical properties of the object containing the PUF. Since PUFs are subject to noise induced by environmental variations, they return slightly different responses when queried with the same challenge multiple times.

PUFs are typically assumed to be *robust*, *physically unclonable*, *unpredictable* and *tamper-evident*, and several approaches to quantify and formally define their properties have been proposed (see [1] for an overview). Informally, robustness means that, when queried with the same challenge multiple times, the PUF returns similar responses with high probability. Physical unclonability demands that it is infeasible to produce two PUFs that are indistinguishable based on their challenge/response behavior. Unpredictability requires that it is infeasible to predict the PUF response to an unknown challenge, even if the PUF can be adaptively queried for a certain number of times. Finally, a PUF is tamper-evident if any attempt to physically access the PUF changes its challenge/response behavior. The properties required from a PUF strongly depend on the application. For instance, a PUF with small challenge/response space can be easily emulated by reading out all its challenge/response pairs and creating a look-up table. While such a PUF cannot be used directly in authentication schemes (such as in [32]), it could still be used in a key storage scenario (such as in [17]), where the adversary is typically assumed not being able to interact with the PUF.

There is a variety of PUF implementations (see [20] for an overview). The most appealing ones for the integration into electronic circuits are *electronic PUFs*, which come in different flavors. *Delay-based PUFs* are based on race conditions in integrated circuits and include arbiter PUFs [15,24,18] and ring oscillator PUFs [6,32,21]. *Memory-based PUFs* exploit the instability of volatile memory cells, such as SRAM [7,9], flip-flops [19,16] and latches [31,14].

Note that memory-based PUFs can be emulated in software since the limited number of memory cells allows creating a look-up table. Further, most delay-based PUFs are subject to model building attacks that allow emulating the PUF in software [15,24,18,26]. To counter this problem, additional primitives must be used: Controlled PUFs [5] and Feed-Forward PUFs [22] use cryptographic functions or XOR-networks in hardware, respectively, to hide the responses of the underlying PUF. Furthermore, PUFs are inherently noisy and must be combined with error correction mechanisms, such as *fuzzy extractors* [3] that remove the effects of noise before the PUF response can be processed in a cryptographic algorithm. Typically, the cryptographic and error correcting components as well as the link between them and the PUF must be protected against invasive and side channel attacks.

3 The PUF ASIC

Our analysis is based on data obtained from 96 ASICs that have been manufactured in TSMC 65 nm CMOS technology within a Europractice multi-project wafer run. The ASIC has been designed within the UNIQUE[4] research project. Each ASIC implements multiple instances of three different memory-based PUFs (SRAM, flip-flop and latch PUFs) and two different delay-based PUFs (ring oscillator and arbiter PUFs). The main characteristics and number of PUF instances in the ASICs are shown in Table 1. Furthermore, the ASIC is equipped with an active core that emulates the noisy working environment of a microprocessor. When enabled, this core performs AES encryption during the PUF evaluation.

The implementation of the arbiter PUF follows the basic approach presented by Lee et al. [15] and consists of 64 delay elements and an arbiter. The delay elements are connected in a line, forming two delay paths with an arbiter placed at the end. Each challenge corresponds to a different configuration of the delay paths. More detailed, each delay element has two inputs and two outputs and

[4] `http://www.unique-project.eu/`

Table 1: Physically Unclonable Functions (PUFs) implemented in the 96 ASICs.

PUF class	**PUF type**	**Number of instances per ASIC**	**Total number of instances**	**Challenge space size**	**Response space size**
Delay-based	Arbiter	256	$24,576$	2^{64}	2
	Ring oscillator	16	$1,536$	$32,640 \approx 2^{15}$	2
Memory-based	SRAM	4 (8 kB)	384	2^{11}	2^{32}
	Flip-flop	4 (1 kB)	384	2^{8}	2^{32}
	Latch	4 (1 kB)	384	2^{8}	2^{32}

can be configured to map inputs to outputs directly (challenge bit 0) or to switch them (challenge bit 1). During the read-out of the PUF response, the input signal propagates along both paths and, depending on which of the paths is faster, a single response bit is generated. To ensure that the delay difference results from the manufacturing process variations rather than the routing of the metal lines, a symmetric layout for the delay elements and full-custom layout blocks were used. Further, to reduce any bias the capacitive loads of the connecting metal wires was balanced and a symmetric NAND-latch was used as arbiter.

The ring oscillator PUF uses the design by Suh et al. [32]. Each ring oscillator PUF consists of 256 ring oscillators and a control logic, which compares the frequency of two different oscillators selected by the PUF challenge. Depending on which of the oscillators is faster, a single response bit is generated. The individual ring oscillators are implemented using layout macros to ensure that all oscillators have exactly the same design, which is fundamental for the correct operation of the ring oscillator PUF.

The memory-based PUFs are implemented as arrays of memory elements (SRAM cells, latches, flip-flops). All these memory elements are bi-stable circuits with two stable states corresponding to a logical 0 and 1. After power-up, each memory element enters either of the two states. The resulting state depends on the manufacturing process variations and the noise in the circuit. When challenged with a memory address, the PUF returns the 32 bit data word at that address. The implementations of the memory-based PUFs follow the SRAM PUF design by Holcomb et al. [9], the flip-flop PUF design by Maes et al. [19] and the latch PUF design by Su et al. [31]. Latch and flip-flop PUFs are implemented using the standard cells from TSMC's 65 nm low-power library. The placement and implementation of the SRAM cells of the SRAM PUF has been done by TSMC's memory compiler. The latch and flip-flop PUFs are based on standard cells using a clustered strategy, where all latches or flip-flops of the same PUF instance are grouped together in single block.

The test setup consists of an ASIC evaluation board, a Xilinx Virtex 5 FPGA and a PC (Figure 1). Each evaluation board can take five ASICs and allows controlling the ASIC supply voltage with an external power supply. The interaction with the evaluation board and the ASICs is performed by the FPGA, which is connected to a PC that controls the PUF evaluation process and stores the raw

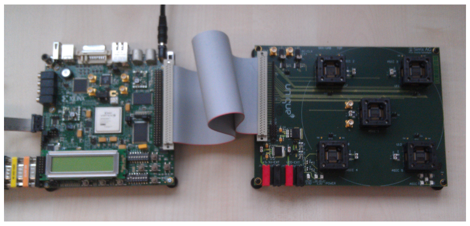


Fig. 1: Test setup with Xilinx Virtex 5 FPGA (left) and ASIC evaluation board with five PUF ASICs (right).

PUF responses obtained from the ASICs. The tests at different temperatures have been performed in a climate chamber.

4 Our Evaluation Methodology

Many PUF-based applications require PUF responses to be reliably reproducible while at the same time being unpredictable (see, e.g., [20,1]). Hence, our empirical evaluation focuses on robustness and unpredictability.

Notation. With $|x|$ we denote the length of some bitstring x. Let E be some event, then $\Pr[E]$ denotes the probability that E occurs. We denote with $\mathsf{HW}(x)$ the Hamming weight of a bitstring x, i.e., the number of non-zero bits of x. With $\mathsf{dist}(x,y)$ we denote the Hamming distance between two bit strings x and y, i.e., the number of bits that are different in x and y.

4.1 Robustness Analysis

Robustness is the property that a PUF always generates responses that are similar to the responses generated during the enrolment of the PUF. Note that PUFs should fulfil this property under different operating conditions, such as different temperatures, supply voltages and noise levels. The robustness of PUFs can be quantified by the bit error rate $\mathrm{BER} := \frac{\mathsf{dist}(y_{E_i}, y_{E_5})}{|y_{E_5}|}$, which indicates the number of bits of a PUF response y_{E_i} that are different from the response y_{E_5} observed during enrolment. We determine the maximum BER of all PUF instances in all ASICs based on challenge/response pairs collected at different ambient temperatures ($-40\,°\mathrm{C}$ to $+85\,°\mathrm{C}$), supply voltages ($\pm 10\%$ of the nominal $1.2\,\mathrm{V}$) and noise levels (active core enabled and disabled), which correspond to the corner values that are typically tested for consumer grade IT products. This shows the impact of the most common environmental factors on the BER of each PUF type. We did not test different noise levels at different temperatures and supply voltages since most PUFs (except the arbiter PUF) turned out to be hardly affected by even the maximum amount of noise the active core can generate. An overview of all test cases considered for robustness is given in Table 2. We estimate the BER of all PUFs in all ASICs using the following procedure:

Table 2: Robustness test cases.

Test	**Active Core**		**Ambient Temperature**			**Supply Voltage**			**Iter.**
Case	Off	On	$-40\,°\mathrm{C}$	$+25\,°\mathrm{C}$	$+85\,°\mathrm{C}$	1.08 V	1.2 V	1.32 V	k
E_1	×		×			×			20
E_2	×		×				×		40
E_3	×		×					×	20
E_4	×			×		×			30
E_5	×			×			×		60
E_6	×			×				×	30
E_7	×				×	×			20
E_8	×				×		×		40
E_9	×				×			×	20
E_{11}		×		×			×		60

Step 1: Sample challenge set generation. A sample challenge set $\mathcal{X}'$ is generated for each PUF type (arbiter, ring oscillator, SRAM, flip-flop and latch PUF) and used in all subsequent steps. For all but the arbiter PUF the complete challenge space is used as a sample set. Since the arbiter PUF has an exponential challenge space, we tested it for $13,000$ randomly chosen challenges, which is a statistically significant subset and representative for the whole challenge space.

Step 2: Enrolment. For each PUF instance, the response y_i to each challenge $x_i \in \mathcal{X}'$ is obtained under nominal operating conditions (test case E_5) and stored in a database $\mathtt{DB}_0$.

Step 3: Data acquisition. For all test cases E_p in Table 2, each PUF instance is evaluated k times on each $x_i \in \mathcal{X}'$ and its responses are stored in a database $\mathtt{DB}_p$ for $p = 1, \ldots, 11$.

Step 4: Analysis. For each PUF instance, the maximum BER between its responses in $\mathtt{DB}_0$ and its responses in $\mathtt{DB}_1$,...,$\mathtt{DB}_{11}$ over all $x_i \in \mathcal{X}'$ is computed.

4.2 Unpredictability Analysis

Unpredictability ensures that the adversary cannot efficiently compute the response of a PUF to an unknown challenge, even if he can adaptively obtain a certain number of other challenge/response pairs from the same and other PUF instances [1]. This is important in most PUF-based applications, such as authentication protocols, where the adversary can forge the authentication when he can predict a PUF response. Note that unpredictability should be independent of the operating conditions of the PUF, which could be exploited by an adversary.

The unpredictability of a PUF implementation can be estimated empirically by applying statistical tests to its responses and/or based on the complexity of the best known attack against the PUF [20,1]. Statistical tests, such as the DIEHARD [23] or NIST [27] test suite, can in principle be used to assess the

Table 3: Unpredictability test cases.

Test Case	**Active Core** Off	On	**Ambient Temperature** $-40\,^{\circ}\mathrm{C}$	$+25\,^{\circ}\mathrm{C}$	$+85\,^{\circ}\mathrm{C}$	**Supply Voltage** 1.08 V	1.2 V	1.32 V
E_{13}	×		×				×	
E_{14}	×			×			×	
E_{15}	×				×		×	
E_{16}	×			×		×		
E_{17}	×			×				×

unpredictability of PUF responses. However, since these test suites are typically based on a series of stochastic tests, they can only indicate whether the PUF responses are random or not. Moreover, they require more input data than the memory-based PUFs and ring oscillator PUFs in the ASIC provide. Similar as in symmetric cryptography, the unpredictability of a PUF can be estimated based on the complexity of the best known attack. There are attacks [26] against delay-based PUFs that emulate the PUF in software and allow predicting PUF responses to arbitrary challenges. These attacks are based on machine learning techniques that exploit statistical deviations and/or dependencies of PUF responses. However, emulation attacks have been shown only for simulated PUF data and it is currently unknown how these attacks perform against real PUFs [26]. Another approach is estimating the entropy of the PUF responses based on experimental data. In particular, *min-entropy* indicates how many bits of a PUF response are uniformly random. The entropy of PUFs can be approximated using the context-tree weighting (CTW) method [39], which is a data compression algorithm that allows assessing the redundancy of bitstrings [10,34,8,16].

We assess the unpredictability of PUFs using Shannon entropy, which is a common metric in cryptography and allows establishing relations to other publications that quantify the unpredictability of PUFs using entropy (such as [35,32,9,1]). We estimate the entropy and min-entropy of the responses of all available PUFs. Specifically, we first check whether PUF responses are biased by computing their Hamming weight and estimate an upper bound of the entropy of PUF responses using a compression test. Eventually, we approximate the entropy and min-entropy of the responses of all available PUFs. Our entropy estimation is more precise than previous approaches since it considers dependencies between the individual bits of the PUF responses. Furthermore, to get an indication of whether responses of *different* PUF instances are independent, we compute the Hamming distance between responses of different PUF instances.

We assess the unpredictability of all available PUFs at different temperatures and supply voltage levels (Table 3) to determine the effects of environmental variations on the unpredictability using the following procedure: We assess the unpredictability of all PUFs in the ASICs using the following procedure:

Step 1: Sample challenge set generation. For each PUF type, a sample challenge set $\mathcal{X}'$ is generated that is used in all subsequent steps. For all but the arbiter PUF, the complete challenge space is used as a sample challenge set. Since the arbiter PUF has an exponential challenge space, we again test it only for $13,000$ challenges. The subsequent analysis steps require $\mathcal{X}' := \{x' \in \mathcal{X}''|\ \mathsf{dist}(x, x') \leq k\}$, which includes a set $\mathcal{X}''$ of randomly chosen challenges and all challenges that differ in at most k bits from the challenges in $\mathcal{X}''$ (that may be known to the adversary).

Step 2: Data acquisition. For all test cases E_q in Table 3, each PUF instance is evaluated on each $x_i \in \mathcal{X}'$ and the responses y are stored in a database $\mathtt{DB}_q$.

Step 3: Analysis. For each test case E_q, the responses in $\mathtt{DB}_q$ are analyzed as detailed in the following items:

Step 3a: Hamming weight. For each PUF instance, the average Hamming weight of all its responses y_i in $\mathtt{DB}_q$ is computed, which indicates whether the responses are biased towards 0 or 1.

Step 3b: CTW Compression. For each PUF instance, a binary file containing all its responses in $\mathtt{DB}_q$ is generated and compressed using the context-tree weighting (CTW) algorithm [38]. The resulting compression rate is an estimate of the upper bound of the entropy of the PUF responses.

Step 3c: Entropy estimation. For each PUF instance, the entropy and min-entropy of all its responses in $\mathtt{DB}_q$ is estimated as detailed in the next paragraph.

Step 3d: Hamming distance. For each PUF type, the Hamming distance $\mathsf{dist}(y, y')$ of all pairs of responses (y, y') in $\mathtt{DB}_q$ generated by pairwise different PUF instances for the same challenge x is computed. While all previous steps consider only responses of the *same* PUF instance, the Hamming distances indicate whether responses of *different* PUF instances are independent. This is important to prevent the adversary from predicting the responses of one PUF implementation based on the challenge/response pairs of another (e.g., his own) PUF implementation, which would contradict the unpredictability property.

Entropy Estimation. Let x be the PUF challenge for which the adversary should predict the response y. Further, let $Y(x)$ be the random variable representing y. Moreover, let $W(x)$ be the random variable representing the set of all responses of the PUF except y, i.e., $W(x) = \{y'|y' \leftarrow \mathsf{PUF}(x');\ x' \in \mathcal{X} \setminus \{x\}\}$. We are interested in the conditional entropy

$$\mathbf{H}(Y|W) = -\sum_{x \in \mathcal{X}} \Pr\left[Y(x), W(x)\right] \cdot \log_2 \Pr\left[Y(x)|W(x)\right] \tag{1}$$

and the conditional min-entropy

$$\mathbf{H}_\infty(Y|W) = -\log_2\left(\max_{x \in \mathcal{X}}\left\{\Pr\left[Y(x)|W(x)\right]\right\}\right), \tag{2}$$

which quantify the average and minimal number of bits of y, respectively, that cannot be predicted by the adversary, even in case all other responses in $W(x)$ are known.[5] Hence, $2^{-\mathbf{H}_\infty(Y|W)}$ is an information-theoretic upper bound for the probability that an adversary guesses the PUF response y to challenge x.

However, computing Equations 1 and 2 for $W(x)$ is difficult since (1) the sizes of the underlying probability distributions are exponential in the response space size, and (2) the complexity of computing $\mathbf{H}(Y|W)$ grows exponentially with the challenge space size of the PUF to be analyzed. Hence, Equations 1 and 2 can at most be estimated by making assumptions on the physical properties of the PUFs that reduce the size of $W(x)$. In the following, we explain how we estimated these entropies for each PUF type and discuss the underlying assumptions.

Memory-based PUFs. A common assumption on memory-based PUFs is that spatially distant memory cells are independent [20,1]. A similar assumption has been used by Holcomb et al. [9], who estimate the entropy of SRAM PUF responses based on the assumption that individual bytes of SRAM are independent. However, physically neighboring memory cells can strongly influence each other, in particular when they are physically connected.[6] Hence, our entropy estimation considers dependencies between neighboring memory cells (which could be exploited by an adversary) while assuming that spatially distant memory cells are independent. More specifically, we compute the entropy of the PUF response bit $Y_{i,j}$ of the memory cell at row i and column j of the underlying memory under the worst case assumption that the values of all neighboring memory cells $W'(x) = (Y_{i-1,j}, Y_{i,j+1}, Y_{i+1,j}, Y_{i,j-1})$ are known, i.e., we compute Equations 1 and 2 for $W'(x)$.

Ring Oscillator PUFs. The ring oscillator PUFs in the ASICs compare the oscillation frequency of two ring oscillators O_i and O_j selected by the PUF challenge $x = (i, j)$ and returns a response $Y(i, j)$, depending on which of the two oscillators was faster. Since neighboring ring oscillators may affect each other (e.g., by electromagnetic induction), we consider the potential dependency between the frequencies of neighboring oscillators and assume that the frequency of spatially distant oscillators is independent. Thus, we compute Equations 1 and 2 for $W'(i, j) = \big(Y_{i-2,j}, Y_{i-1,j}, Y_{i+1,j}, Y_{i+2}\big)$.

Arbiter PUFs. Arbiter PUFs measure the delay difference of two delay lines that are configured by the PUF challenge. The individual delays caused by the switches and their connections are additive, which implies that the PUF response y to a challenge x can be computed if a sufficient number of responses to challenges that are close to x are known. Hence, we compute Equations 1 and 2 for $W'(x) = \{y' \leftarrow \mathsf{PUF}(x') | x' \in \mathcal{X}',\ \mathsf{dist}(x, x') \leq k\}$, which corresponds to the

[5] Note that this corresponds to the game-based security definition of unpredictability by Armknecht et al. [1], which formalizes the difficulty of predicting Y in case the PUF responses in W are known.

[6] SRAM cells are typically arranged in a matrix, where all cells in a row are connected by a word line and all cells in a column are connected by a bit line.

worst case where the adversary knows responses to challenges that differ in at most k bits from the challenge whose response he must guess. Specifically, we use $\mathcal{X}$ consisting of 200 randomly chosen challenges and $k = 1$.

Computing the Entropy. To compute the entropy and min-entropy (Equations 1 and 2) for each test case E_q, we first estimate $\Pr\left[x = Y(x), w = W(x)\right]$ for each $x \in \mathcal{X}'$ by dividing the number of observations of each tuple (x, w) in database $\mathtt{DB}_q$ by the size of the sample challenge set $\mathcal{X}'$. Further, to compute $\Pr\left[x = Y(x) \middle| w = W(x)\right] = \Pr\left[x = Y(x), w = W(x)\right] / \Pr\left[w = W(x)\right]$, we estimated $\Pr\left[w = W(x)\right]$ by dividing the number of observations of each tuple $\big(Y(x), w = W(x)\big)$ in database $\mathtt{DB}_i$ by the size of $\mathcal{X}'$. Eventually, we computed Equations 1 and 2.

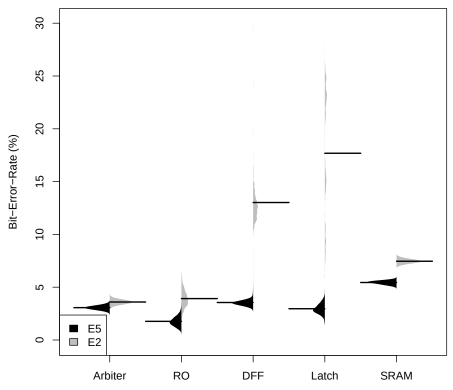
5 Evaluation and Results

We applied the evaluation methodology in Section 4 to all PUF instances in all ASICs. Most of our results are illustrated using *bean plots* [12] that allow an intuitive visualization of empirical probability distributions (Figures 2 to 5). Each bean shows two distributions, smoothed by a Gaussian kernel to give the impression of a continuous distribution, together with their means indicated by black bars. The distribution in black on the left side typically corresponds to data collected under normal PUF operating conditions, while the one in gray on the right side corresponds to some other test case in Table 2 and 3.

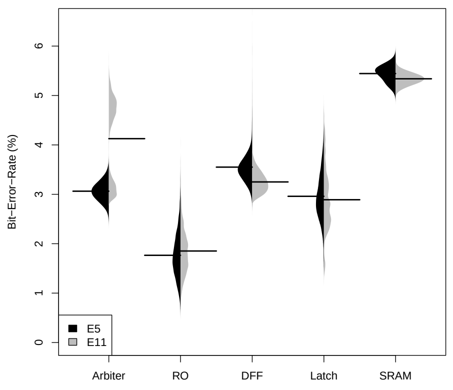
Due to space restrictions, we illustrate only the most important results and provide a detailed discussion in the full version of this paper [13].

5.1 Robustness Results

We computed the bit error rate (BER) under varying environmental conditions (Table 3). Our results show that all arbiter, ring oscillator and SRAM PUF instances have a very similar BER, while there is a big variability in the BERs of the flip-flop and latch PUF instances (Figure 2). Further, the BER of the arbiter, ring oscillator and SRAM PUF instances is below 10% for all test cases, which can be handled by common error correction schemes, such as fuzzy extractors [3]. The BER of most PUFs depends on the operating temperature (Figure 2a): Compared to $+25\,^{\circ}\mathrm{C}$ (test case E_5), at $-40\,^{\circ}\mathrm{C}$ (test case E_2) the BER of the flip-flop and latch PUF increases significantly, while the BER of the ring oscillator and SRAM PUF increases only slightly and the BER of the arbiter PUF hardly changes. A similar behavior of the BERs can be observed at $+85\,^{\circ}\mathrm{C}$ (test case E_8). All PUFs in all ASICs turned out to be robust against variations of their supply voltages. Compared to nominal operating conditions (test case E_5), the distributions of the BERs only slightly increase when varying the supply voltage by 10% (test case E_4 and E_6). The arbiter PUF exhibits a significantly increased BER when operated in a noisy working environment (test case E_{11}; Figure 2b) while there is no significant change of the BER of all other PUFs. Hereby, we



(a) Bit error rates at $+25\,^{\circ}\mathrm{C}$ (test case E_5, black) and at $-40\,^{\circ}\mathrm{C}$ (test case E_2, gray)



(b) Bit error rates with active core off (test case E_5, black) and active core on (test case E_{11}, gray)

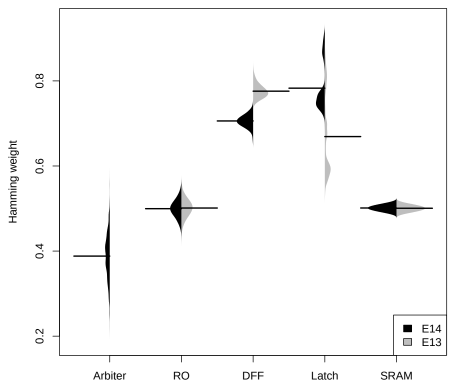
Fig. 2: Distribution of the bit error rate (BER) in percent over all PUF instances at different ambient temperatures and noise levels. The two peaks of the BER distribution of the arbiter PUF show that those arbiter PUFs that are spatially close to the active core are more affected than those farther away.

observed that the BER of arbiter PUF instances that are spatially close to the active core significantly changes, while those that are farther away are not directly affected.

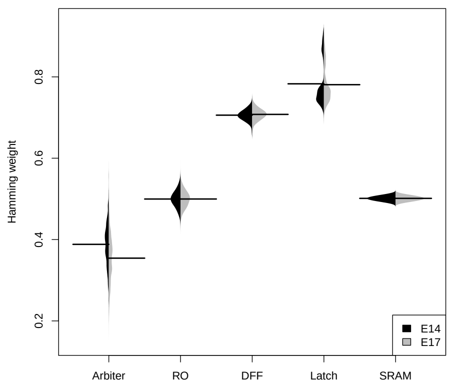
5.2 Unpredictability Results

In this section, we present the results of our unpredictability analysis. Due to the time-limited access to the climate chamber, the data required to analyze the unpredictability of the arbiter PUF at $-40\,^{\circ}\mathrm{C}$ and at $+85\,^{\circ}\mathrm{C}$ is not available. However, we show the results for normal operating conditions and different supply voltages.

Hamming Weights. To get a first indication of randomness in the PUFs, we computed the Hamming weight of their responses as described in Section 4.2. Our results show that ring oscillator and SRAM PUF responses are close to the ideal Hamming weight of 0.5, independent of the operating conditions (Figure 3), which indicates that their responses may be random. The Hamming weight of the flip-flop PUF and latch PUF responses strongly depends on the ambient temperature (Figure 3a) and is clearly biased. Supply voltage variations (test cases E_{16} and E_{17}) have no significant impact on the Hamming weight of the responses of any of the PUF instances in the ASIC (Figure 3b).



(a) Hamming weight at +25 °C (test case E_{14}, black) and −40 °C (test case E_{13}, gray)



(b) Hamming weight at 1.20 V (nominal voltage, test case E_{14}, black) and 1.32 V (+10% overvoltage, test case E_{17}, gray)

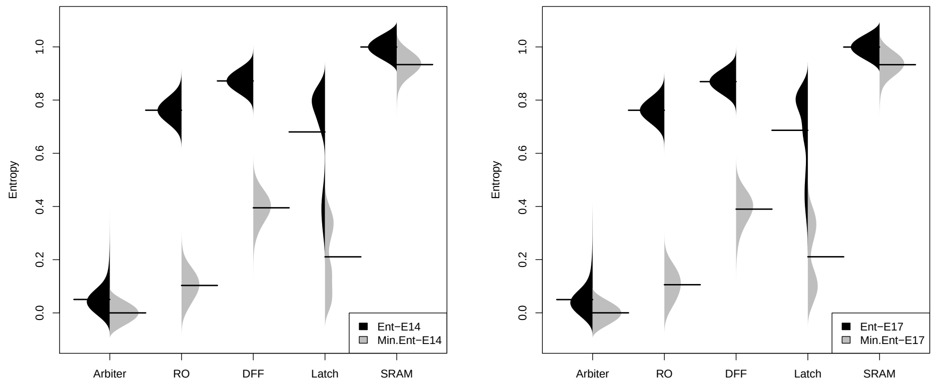
Fig. 3: Distribution of the Hamming weight over all PUF instances at different ambient temperatures and supply voltage levels. The two peaks of the Hamming weight distribution of the latch PUF may come from the fact that one of the four latch PUF instances on each ASIC is implemented in a separate power domain.

CTW Compression. The context-tree weighting (CTW) compression test gives a good indication of the upper bound of the entropy of PUF responses. The higher the compression rate, the lower the entropy of the PUF. The results of this test (Table 4) confirm the Hamming weight test results: The compression rate of the ring oscillator and SRAM PUF responses is invariant for all test cases; the compression rates of the flip-flop and latch PUF responses do not change for different supply voltages (test case E_{16} and E_{17}), but vary with the ambient temperature (test cases E_{13}, E_{14} and E_{15}). The compression rate of the SRAM PUF responses strongly indicates that these responses are uniformly random, while there seem to be some dependencies in the responses generated by all other PUFs.

Entropy Estimation. The results of the entropy estimation described in Section 4.2 confirm the results of all previous tests and provide more insights into the entropy and min-entropy of the PUF responses (Figure 4). The entropy of responses corresponding to neighboring arbiter PUF challenges is remarkably low, which confirms the high prediction rate of emulation attacks against arbiter PUFs reported in literature [26]. The entropy and min-entropy of the ring oscillator and SRAM PUF responses is invariant to temperature (test cases E_{13}, E_{14} and E_{15}) and supply voltage (test case E_{16} and E_{17}) variations. Moreover, the entropy and min-entropy of flip-flop and latch PUFs vary with the operating

Table 4: CTW compression results.

Test	**Size of PUF response after CTW compression in percent**				
Case	Arbiter	Ring-Oscillator	Flip-Flop	Latch	SRAM
E_{13}	—	0.77	0.77	0.84	1.00
E_{14}	0.51	0.77	0.87	0.70	1.00
E_{15}	—	0.77	0.98	0.53	1.00
E_{16}	0.53	0.77	0.88	0.69	1.00
E_{17}	0.49	0.77	0.87	0.71	1.00



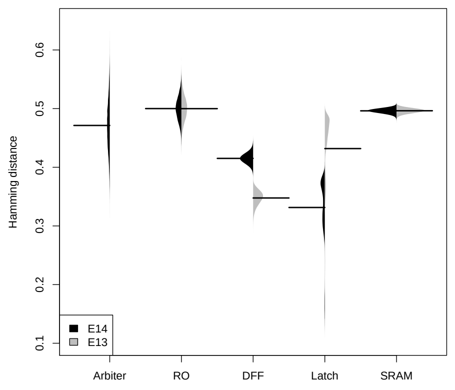
(a) Entropy (black) and min-entropy (gray) at +25 °C (test case E_{14})

(b) Entropy (black) and min-entropy (gray) at 1.32 V (+10%, test case E_{17})

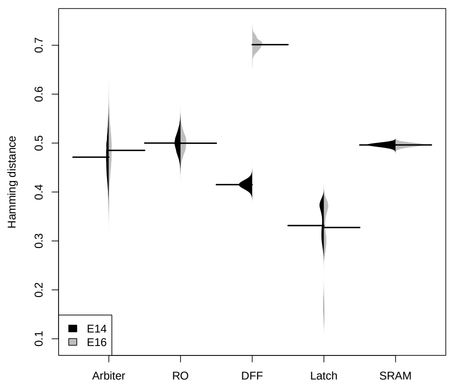
Fig. 4: Distribution of the entropy (black) and min-entropy (gray) over all PUF instances at different ambient temperatures and supply voltage levels.

temperature (test cases E_{13}, E_{14} and E_{15}) and are constant for different supply voltages (test case E_{16} and E_{17}).

Hamming Distances. The Hamming distance test (Section 4.2) gives an indication of whether the responses generated by different PUF instances to the same challenge are independent. Our results show that, independent of the ambient temperature (test cases E_{13}, E_{14} and E_{15}) and supply voltage (test cases E_{16} and E_{17}), the responses of different ring oscillator and SRAM PUF instances have the ideal Hamming distance of 0.5, while there seem to be dependencies between the responses generated by different arbiter PUF instances to the same challenge (Figure 5). The Hamming distance of the responses of the flip-flop PUFs changes for different temperatures and supply voltages. At +85 °C (test case E_{15}) the Hamming distance of the flip-flop PUF is ideal, while it is biased towards zero at −40 °C (test case E_{13}). Moreover, at 1.08 V (−10% undervoltage, test case E_{16}) we observed a bias of the Hamming distance towards one, while



(a) Hamming distance at +25 °C (test case E_{14}, black) and at −40 °C (test case E_{13}, gray)



(b) Hamming distance at 1.20 V (nominal voltage, test case E_{14}, black) and at 1.08 V (−10% undervoltage, test case E_{16}, gray)

Fig. 5: Distribution of the Hamming distance over all PUF instances at different ambient temperatures and supply voltage levels.

the Hamming distance at 1.32 V (+10% overvoltage, test case E_{17}) is similar to the distribution at nominal operating conditions (test case E_{14}). The Hamming distance of the responses of the latch PUFs are biased towards zero and invariant for different supply voltages.

5.3 Discussion

Our results show that arbiter, ring oscillator and SRAM PUFs are more robust to temperature variations than the latch and flip-flop PUFs. This could be due to the dual nature of these PUFs, i.e., the two delay paths, two ring oscillators, and the symmetrical structure of the SRAM cells, respectively. As discussed in Section 3, we do not have access to the internal circuit diagrams and layout of the standard cells provided by TSMC and thus can only speculate about the transistor schematics of the flip-flops and latches. Standard cell libraries typically use implementations based on transmission gates, which are more compact than static latches or flip-flops with a dual structure and there is no duality or symmetry in these transistor schematics. Further, the results of the Hamming weight and Hamming distance tests indicate that the unpredictability of PUFs with a dual structure are less affected by temperature variations.

The entropy of the arbiter PUF is remarkably low, which can be explained by the linear structure of this PUF. Note that in the arbiter PUF implementation, two signals travel along two delay paths and finally arrive at an arbiter (Section 3). In case the delay difference δ_t of the two paths is greater than the setup-time t_{setup} plus the hold time t_{hold} of the arbiter, the PUF response will be correctly generated according to which signal arrives first. However, in case $\delta_t < t_{\text{setup}} + t_{\text{hold}}$, the arbiter will be in the metastable state and the PUF response will depend on the bias of the arbiter caused by manufacturing process and/or layout variations of the arbiter and the noise in the circuit. A limited number of simulations (with 20 PUFs for 3 challenges) including extracted post layout parasitics were performed before the tape-out of the ASIC to estimate this effect.

Since the arbiter PUF design is based on delay accumulation, it is very susceptible to emulation attacks [26]. An example illustrating this fact is the case where two challenges differ in only the last bit. In this case, signals will travel along the same paths through 63 delay elements, and only in the last element the paths will be different. If the attacker knows the outcome for one challenge, he can guess the outcome of the other one with high probability, which might explain the low entropy and min-entropy of the arbiter PUFs.

5.4 Summary

The arbiter PUF responses have a very low entropy and their use in applications with strict unclonability and unpredictability requirements should be carefully considered. Further, the arbiter PUFs are susceptible to changes of their supply voltage and to environmental noise, which significantly increases the bit error rate of the PUF. However, the bit error rate stays within acceptable bounds and can be compensated by existing error correction mechanisms.

The flip-flop and latch PUFs are susceptible to temperature variations, which have a significant effect on the bit error rate and the unpredictability of the PUF responses. Hence, flip-flop and latch PUFs should not be used in an environment, where the adversary can lower the ambient temperature of the PUF, reducing the entropy of the PUF responses.

The SRAM and ring oscillator PUFs achieve almost all desired properties of a PUF: The bit error rate does not change significantly under different operating conditions, the entropy of the PUF responses is high and the responses generated by different PUF instances seem to be independent. However, the ring oscillator exhibits a low min-entropy, which might be problematic in some applications.

6 Conclusion

We performed the first large-scale analysis of the five most popular PUF types (arbiter, ring oscillator, SRAM, flip-flop and latch PUFs) implemented in ASIC. Our analysis is based on PUF data obtained from 96 ASICs, each housing several PUF instances. Our results allow for the first time a fair comparison of these PUFs. In this context, we presented an evaluation methodology for the empirical assessment of the robustness and unpredictability properties of PUFs that are fundamental in most applications of PUFs.

Our results show that the SRAM and ring oscillator PUFs seem to achieve all desired properties of a PUF. However, the arbiter PUFs have a very low entropy and the entropy of the flip-flop and latch PUFs is susceptible to temperature variations. Hence, the suitability of these PUFs for security-critical applications, such as authentication or key generation must be carefully considered.

Future work includes the analysis of stronger PUF constructions and the development of entropy estimation methodologies that also include potential dependencies between different PUF instances.

Acknowledgement. We thank all our partners Intel, Intrinsic ID, KU Leuven, and Sirrix AG, who developed the ASIC and evaluation board. Further, we thank Intrinsic ID for providing us the raw PUF data for the test cases at $-40\,^{\circ}\mathrm{C}$ and $+85\,^{\circ}\mathrm{C}$. Moreover, we thank Vincent van der Leest (Intrinsic ID), Roel Maes (KU Leuven) and our anonymous reviewers for their helpful comments. This work has been supported by the European Commission under grant agreement ICT-2007-238811 UNIQUE.

References

1. Armknecht, F., Maes, R., Sadeghi, A.R., Standaert, F.X., Wachsmann, C.: A formal foundation for the security features of physical functions. In: IEEE Symposium on Security and Privacy (SSP). pp. 397–412. IEEE Computer Society (May 2011)
2. Armknecht, F., Maes, R., Sadeghi, A.R., Sunar, B., Tuyls, P.: Memory Leakage-Resilient encryption based on physically unclonable functions. In: Advances in Cryptology (ASIACRYPT). LNCS, vol. 5912, pp. 685–702. Springer Berlin/Heidelberg (2009)
3. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Advances in Cryptology (EUROCRYPT). LNCS, vol. 3027, pp. 523–540. Springer Berlin/Heidelberg (2004)
4. Eichhorn, I., Koeberl, P., van der Leest, V.: Logically reconfigurable PUFs: Memory-based secure key storage. In: ACM Workshop on Scalable Trusted Computing (ACM STC). pp. 59–64. ACM, New York, NY, USA (2011)
5. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled physical random functions. In: Computer Security Applications Conference (ACSAC). pp. 149–160. IEEE (2002)
6. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: ACM Conference on Computer and Communications Security (ACM CCS). pp. 148–160. ACM, New York, NY, USA (2002)
7. Guajardo, J., Kumar, S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Cryptographic Hardware and Embedded Systems (CHES). LNCS, vol. 4727, pp. 63–80. Springer Berlin/Heidelberg (2007)
8. Hammouri, G., Dana, A., Sunar, B.: CDs have fingerprints too. In: Cryptographic Hardware and Embedded Systems (CHES). LNCS, vol. 5747, pp. 348–362. Springer Berlin/Heidelberg (2009)
9. Holcomb, D.E., Burleson, W.P., Fu, K.: Power-Up SRAM state as an identifying fingerprint and source of true random numbers. IEEE Transactions on Computers 58(9), 1198–1210 (2009)

10. Ignatenko, T., Schrijen, G.J., Škorić, B., Tuyls, P., Willems, F.: Estimating the Secrecy-Rate of physical unclonable functions with the Context-Tree weighting method. In: IEEE International Symposium on Information Theory (ISIT). pp. 499–503. IEEE (Jul 2006)
11. Intrinsic ID: Product webpage. http://www.intrinsic-id.com/products.htm (November 2011)
12. Kampstra, P.: Beanplot: A boxplot alternative for visual comparison of distributions. Journal of Statistical Software 28(1), 1–9 (Oct 2008)
13. Katzenbeisser, S., Ünal Kocabaş, Rožić, V., Sadeghi, A.R., Verbauwhede, I., Wachsmann, C.: PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon. Cryptology ePrint Archive, To appear (2012)
14. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: Extended abstract: The butterfly PUF protecting IP on every FPGA. In: Workshop on Hardware-Oriented Security (HOST). pp. 67–70. IEEE (Jun 2008)
15. Lee, J.W., Lim, D., Gassend, B., Suh, E.G., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Symposium on VLSI Circuits. pp. 176–179. IEEE (Jun 2004)
16. van der Leest, V., Schrijen, G.J., Handschuh, H., Tuyls, P.: Hardware intrinsic security from D flip-flops. In: ACM Workshop on Scalable Trusted Computing (ACM STC). pp. 53–62. ACM, New York, NY, USA (2010)
17. Lim, D., Lee, J.W., Gassend, B., Suh, E.G., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 13(10), 1200–1205 (Oct 2005)
18. Lin, L., Holcomb, D., Krishnappa, D.K., Shabadi, P., Burleson, W.: Low-power sub-threshold design of secure physical unclonable functions. In: International Symposium on Low-Power Electronics and Design (ISLPED). pp. 43–48. IEEE (Aug 2010)
19. Maes, R., Tuyls, P., Verbauwhede, I.: Intrinsic PUFs from flip-flops on reconfigurable devices (Nov 2008)
20. Maes, R., Verbauwhede, I.: Physically unclonable functions: A study on the state of the art and future research directions. In: Towards Hardware-Intrinsic Security. pp. 3–37. Information Security and Cryptography, Springer Berlin/Heidelberg (2010)
21. Maiti, A., Casarona, J., McHale, L., Schaumont, P.: A large scale characterization of RO-PUF. In: International Symposium on Hardware-Oriented Security and Trust (HOST). pp. 94–99. IEEE (Jun 2010)
22. Majzoobi, M., Koushanfar, F., Potkonjak, M.: Testing techniques for hardware security. In: International Test Conference (ITC). pp. 1–10. IEEE (Oct 2008)
23. Marsaglia, G.: The Marsaglia random number CDROM including the diehard battery of tests of randomness. http://www.stat.fsu.edu/pub/diehard/
24. Öztürk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: International Conference on Pervasive Computing and Communications (PerCom). pp. 170–178. IEEE, Washington, DC, USA (Mar 2008)
25. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical One-Way functions. Science 297(5589), 2026–2030 (Sep 2002)
26. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: ACM Conference on Computer and Communications Security (ACM CCS). pp. 237–249. ACM, New York, NY, USA (2010)
27. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite

for random and pseudorandom number generators for cryptographic applications. Special Publication 800-22 Revision 1a, NIST (April 2010)
28. Sadeghi, A.R., Visconti, I., Wachsmann, C.: Enhancing RFID security and privacy by physically unclonable functions. In: Towards Hardware-Intrinsic Security. pp. 281–305. Information Security and Cryptography, Springer Berlin/Heidelberg (Nov 2010)
29. Schulz, S., Sadeghi, A.R., Wachsmann, C.: Short paper: Lightweight remote attestation using physical functions. In: Proceedings of the Fourth ACM Conference on Wireless Network Security (ACM WiSec). pp. 109–114. ACM, New York, NY, USA (2011)
30. Škorić, B., Maubach, S., Kevenaar, T., Tuyls, P.: Information-theoretic analysis of capacitive physical unclonable functions. Journal of Applied Physics 100(2) (Jul 2006)
31. Su, Y., Holleman, J., Otis, B.P.: A digital 1.6 pJ/bit chip identification circuit using process variations. IEEE Journal of Solid-State Circuits 43(1), 69–77 (Jan 2008)
32. Suh, E.G., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: ACM/IEEE Design Automation Conference (DAC). pp. 9–14. IEEE (Jun 2007)
33. Tuyls, P., Batina, L.: RFID-tags for anti-counterfeiting. In: Topics in Cryptology (CT-RSA). LNCS, vol. 3860, pp. 115–131. Springer Berlin/Heidelberg (Feb 2006)
34. Tuyls, P., Škorić, B., Ignatenko, T., Willems, F., Schrijen, G.J.: Entropy estimation for optical PUFs based on Context-Tree weighting methods security with noisy data. In: Security with Noisy Data. pp. 217–233. Springer London (2007)
35. Tuyls, P., Škorić, B., Stallinga, S., Akkermans, A.H.M., Ophey, W.: Information-Theoretic security analysis of physical uncloneable functions. In: Financial Cryptography and Data Security (FC). LNCS, vol. 3570, p. 578. Springer Berlin/Heidelberg (2005)
36. Verayo, Inc.: Product webpage. http://www.verayo.com/product/products.html (November 2011)
37. Škorić, B., Tuyls, P., Ophey, W.: Robust key extraction from physical uncloneable functions. In: Applied Cryptography and Network Security (ACNS). LNCS, vol. 3531, pp. 99–135. Springer Berlin/Heidelberg (2005)
38. Willems, F.: CTW website. http://www.ele.tue.nl/ctw/
39. Willems, F.M.J., Shtarkov, Y.M., Tjalkens, T.J.: The context-tree weighting method: basic properties. IEEE Transactions on Information Theory 41(3), 653–664 (May 1995)