Remanence Decay Side-Channel: The PUF Case

Shaza Zeitouni, Yossef Oren, Christian Wachsmann, Patrick Koeberl, Ahmad-Reza Sadeghi

***Abstract*—We present a side-channel attack based on remanence decay in volatile memory and show how it can be exploited effectively to launch a non-invasive cloning attack against SRAM PUFs — an important class of PUFs typically proposed as lightweight security primitives which use existing memory on the underlying device. We validate our approach using SRAM PUFs instantiated on two 65nm CMOS devices. We discuss countermeasures against our attack and propose the constructive use of remanence decay to improve the cloning-resistance of SRAM PUFs.**

Moreover, as a further contribution of independent interest, we show how to use our evaluation results to significantly improve the performance of the recently proposed TARDIS scheme, which is based on remanence decay in SRAM memory and used as a time-keeping mechanism for low-power clockless devices.

I. Introduction

Physically Unclonable Functions (PUFs) have been an attractive research area and are increasingly proposed as building blocks in cryptographic protocols and security architectures. One major class of PUFs and the focus of this paper are memory-based PUFs [1], [2], [3], [4], [5], [6]. These PUFs are commonly proposed as an alternative to secure non-volatile storage and are used in a variety of anti-counterfeiting mechanisms and authentication schemes [7], [8], [1], [9], [10], [11], [12]. Today, PUF-based security products are already on the market, mainly targeting IP-protection and anti-counterfeiting applications as well as Radio-Frequency Identification (RFID) systems [13], [14], [15].

Memory-based PUFs are arrays of volatile memory elements, such as SRAM cells [1], [5], flip-flops [2], [6], or latches [3], [4]. These elements are typically bi-stable circuits with two stable states corresponding to a logical zero and one. By controlling the voltage level at the inputs of the elements, they enter either one of the two states. Due to the bi-stability, the elements retain their states as long as they are supplied with power. Memory-based PUFs exploit the following phenomenon: When powering up such an element without applying any voltage at the bit-line input, its state mainly depends on the physical characteristics of the underlying transistors. Due to uncontrollable manufacturing variations, these characteristics are unique for each physical instantiation of the element. Hence, the state of all memory elements, after powering the memory without without applying any voltage at the bit-lines, can be used as a unique identifier, known as *'PUF response'*, of the device containing the memory. However, since the response of a memory-based PUF could be read out and copied to another device, protecting the PUF response against unauthorized accesses is considered a fundamental requirement of memory-based PUF implementations, which implies mimimally the presence of some security mechanism to prevent unauthorized accesses to the PUF response.

Memory-based PUFs are considered to be cost-effective since it is possible to use the already existing memory of the device as PUFs [8], [1], [11], [16], [17], [18], [19]. However, in this case the memory is also used to store data of some other components in the device and will be overwritten at some point of time. In particular, volatile memory is typically initialized, i.e., overwritten with a known bit pattern, usually all zeros or ones, before it is used for data storage. Further, although volatile memory loses the data it stores when it is powered off, the data are not immediately lost but rather decay slowly over time [20], [21]. Hence, it is very likely that any data written to the memory of a memory-based PUF may affect the PUF response when removing or reducing the power supply for short time periods. Although this effect has been discussed in the literature [22], [23], [5], [24], [25], [26], it has never been used to attack memory-based PUFs. A preliminary and shorter version of this work has been published at [27]. This is an extended version that includes new evaluation results of voltage-based attacks.

Contribution. We present the first side-channel attack based on the remanence decay in volatile

Y. Oren is affiliated with Ben-Gurion University of the Negev, Beer-Sheva, Israel. E-mail: yos@bgu.ac.il.

A.-R. Sadeghi is affiliated with Technische Universität Darmstadt, Germany. E-mail: ahmad.sadeghi@trust.cased.de.

P. Koeberl, C. Wachsmann and S. Zeitouni are affiliated with Intel Collaborative Research Institute for Secure Computing (Intel CRI-SC), Darmstadt, Germany. E- mail: patrick.koeberl@intel.com, christian.wachsmann@trust.cased.de and shaza.zeitouni@trust.cased.de.

memory and show how it can be exploited for a non-invasive cloning attack against SRAM-based PUFs. To the best of our knowledge this is the first cloning attack on memory-based PUFs based on remanence decay. In particular, our contributions are as follows:

First cloning attack on SRAM PUFs using remanence decay side channels. Our attack recovers the secret response of a memory-based PUF in applications where the underlying memory is overwritten with a known value after the PUF response has been read. This attack can be applied to all memory-based PUF systems that share the PUF memory with some other functionality, which is often suggested in the literature to allow cost-effective PUF implementations [8], [1], [11], [16], [17], [18], [19]. We show that the attack is successful against small memory-based PUFs even when using common lab equipment. The requirements of the attack are that the adversary can control the supply voltage of the device containing the PUF and that the PUF memory is initialized with a known value before it is used as a data storage, which is typically the case.

Experimental validation of the attack. We validate the feasibility of our attack using SRAM PUFs instantiated on two 65nm CMOS devices, and suggest several improvements to increase the performance of our attack.

Constructive use of remanence decay. We propose using remanence decay as a source of side-channel information to enhance the cloning-resistance of SRAM PUFs. Cloning such a PUF would require emulating the remanence decay behavior, which increases the cost of cloning or even render it uneconomical.

Improved TARDIS time-keeping mechanism. As a contribution of independent interest, we propose a time-memory tradeoff to dramatically reduce the complexity of the recently proposed TARDIS [21] time-keeping mechanism for clockless devices from linear to logarithmic time, enhancing its applicability to many practical scenarios; we further propose a simplified version of TARDIS which satisfies the main functional requirements of the TARDIS mechanism while requiring only constant time and minimal non-volatile storage.

Outline. We introduce our notation, the system and the adversary model in Section II. The attack is described in Section III and its experimental validation is presented in Section IV. A practical instantiation of our attack is shown in Section V. We discuss the impact and improvements of the attack in Section VI and make suggestions on the constructive use of remanence decay, including the improved TARDIS algorithm, in Section VII. We give an overview of the related work in Section VIII and finally conclude in Section IX.

II. Model and Preliminaries

We consider devices that contain memory-based PUFs, where the underlying memory can be overwritten with a known value after the PUF response has been read. This typically happens when the PUF memory is also used for data storage by some other functionality in the device, which is a common approach to cost-effective implementations of memory-based PUFs [8], [1], [11], [16], [17], [18], [19].

Initial state. Volatile memory is typically initialized, i.e., overwritten with a specific bit pattern (usually all zeroes or ones), before it is used as a data storage. We denote this pattern as the *initial state* of the memory.

Definition 1 (Initial state)**.** *The* initial state *of the memory is the matrix $\vec{M}_{\text{init}}$ representing the data that is written to the memory before it is used as a data storage.*

Start-up state. Observe that, typically the data stored in the volatile memory are not immediately lost when the power to the memory is removed or decreased but decay slowly over time [20], [21]. Hence, when powered off only for a short time, the memory may still hold some of the data that have been written to it before the power-cycle. We capture this aspect by introducing the notion of the *start-up state*.

Definition 2 (Start-up state)**.** *Let v_{nom} be the nominal supply voltage of the memory V_{dd}.*
Consider the following experiment:

1) *Set supply voltage of memory elements to 0 V for time t*
2) *Set supply voltage of memory elements to v_{nom}*
3) *Read the states of all memory elements and store them in a matrix $\vec{M}_t$*

We say that $\vec{M}_t$ is the start-up state *of the memory with respect to time t.*
Further, consider the following experiment:

1) *Set supply voltage of memory elements to $v < v_{\text{nom}}$ for constant time τ*
2) *Set supply voltage of memory elements to v_{nom}*
3) *Read the states of all memory elements and store them in a matrix $\vec{M}_v$*

We say that $\vec{M}_v$ is the start-up state *of the memory with respect to voltage v and a constant time τ.*

PUF state. The response of a memory-based PUF corresponds to the start-up state of the underlying memory, where the memory has been powered off long enough that any data previously stored in it have decayed. We capture this aspect by introducing the notion of the *PUF state* of a memory.

Definition 3 (PUF state)**.** *Let t_∞ indicates the time required for previously stored data to completely decay from the memory. We denote the start-up state $\vec{M}_{t_\infty}$ as the* PUF state $\vec{M}_{\text{PUF}}$ *of the memory, i.e., $\vec{M}_{\text{PUF}} := \vec{M}_{t_\infty}$.*

Observe that, in the case where the memory has been powered off only for a short time before it is used as a PUF, the PUF response may be distorted by the data previously stored in the memory.

Device behavior. At some point while the device is running, it reads the start-up state of its memory and uses it as the PUF response in some computation. In many applications the result of this computation can be observed from outside the device. For instance, in PUF-based (authentication) protocols [8], [11], [12], the device receives some query Q and responds with a message X that depends on the PUF response. In these schemes, the response of the memory-based PUF is typically used to derive a cryptographic secret that is used to compute X. However, the device behavior is not limited to challenge-response protocols. In the extreme case X could be only one single bit of information, e.g., indicating whether the correct PUF response was extracted from the memory or not. For instance, in PUF-based IP protection schemes [1], [9], [10], the device refuses to boot in the case where the PUF response is incorrect, which can be observed by the adversary. We capture this aspect by introducing the notion of *device behavior*.

Definition 4 (Device behavior)**.** *Let $\vec{M}$ be the start-up state of the device memory with respect to some time t or voltage v (Definition 2). Further, let Q be some query that can be sent to the device. We denote with $X = \mathsf{Dev}(\vec{M}, Q)$ the response to Q of the device using the start-up state $\vec{M}$. The algorithm* Dev *describes the* behavior *of the device with respect to Q and $\vec{M}$.*

Assumptions and adversary model. Following the common adversary model of memory PUFs [8], [1], [11], [16], [17], [18], [19], we assume that the adversary $\mathcal{A}$ cannot simply read the plain PUF response from the underlying memory. This means that $\mathcal{A}$ does *not* know the start-up state $\vec{M}$ (Definition 2) with respect to any time t or voltage v and, in particular, $\mathcal{A}$ does not know the PUF state $\vec{M}_{\text{PUF}}$ (Definition 3). Further, we assume that all algorithms implemented in the device are known to $\mathcal{A}$ (Kerckhoffs' principle). This means that $\mathcal{A}$ could compute $X = \mathsf{Dev}(\vec{M}, Q)$ if he knew $\vec{M}$ and Q. Moreover, $\mathcal{A}$ knows the initial state $\vec{M}_{\text{init}}$ (Definition 1) that is part of the algorithms used by the device. Furthermore, we assume that $\mathcal{A}$ can observe the device behavior (Definition 4) and that $\mathcal{A}$ can control the time t the memory is powered off before it is used as a PUF as well as the supply voltage v of the memory. This means that $\mathcal{A}$ can send some query Q to the device and observe its reaction X that depends on the device's start-up state $\vec{M}$.

III. Cloning Attack Using Remanence Decay

The high level idea and approach of our attack is to recover the PUF response in a device that overwrites the SRAM of the PUF with some data that are known to the adversary $\mathcal{A}$ (cf. Section II). The attack principle is similar to Biham-Shamir attack [28] to extract a secret key stored in some device (e.g., a smart card).

The Biham-Shamir attack consists of two phases. In the first phase, $\mathcal{A}$ collects a sequence of ciphertexts, each encrypting the same plaintext with a slightly different key. In more detail, $\mathcal{A}$ requests the device to encrypt the plaintext and, after he received the corresponding ciphertext, he injects a fault into the device that sets one bit of the key to a known value. $\mathcal{A}$ repeats this step until all bits in the key are set to a value $\mathcal{A}$ knows. In the second phase of the attack, $\mathcal{A}$ iteratively recovers the secret key of the device, starting from the last ciphertext that has been generated by the device using the key known to $\mathcal{A}$. $\mathcal{A}$ performs an exhaustive search for each key used by the device to generate each ciphertext collected in the first phase. Since the keys of two consecutive ciphertexts differ in at most one single bit and the value of this bit is known to $\mathcal{A}$, this exhaustive search is linear in the bit-length of the key. This way, $\mathcal{A}$ can recover the secret key of the device with a total effort quadratic in the bit-length of the key.

Similarly, we aim at extracting the secret PUF state from a device containing an SRAM PUF. Similar to the Biham-Shamir attack, we iteratively collect a series of device responses to the same query, each generated using a different start-up state. In each iteration, we send the query to the device, record its response (that depends on the start-up state), and then inject a fault to change some bits in the start-up state. The fault injection is performed by carefully controlling the amount of remanence decay undergone by the SRAM, e.g., by increasing the time the device is powered off between two iterations,

or by reducing the device's supply voltage for a fixed time period. Due to the different remanence decay exhibited by each SRAM cell, for any given power-off period or reduction of voltage supply, some SRAM cells will lose the known value of the initial state and revert back to their unknown PUF state, some will retain their initial state and some will exhibit *metastable* behaviour by taking a random state. Hence, in contrast to the Biham-Shamir attack, the number of bits k that are different in the start-up states used in two consecutive iterations is typically larger than one bit. However, as we show in Section IV, k has an upper bound that highly depends on the exhaustive method or the computational power and a lower bound imposed by the accuracy of the equipment used to control the remanence decay.

In the second phase of the attack, we iteratively recover the PUF state. A trivial approach would be to perform a simple exhaustive search for all cells that have reverted to their PUF state in the start-up states of two consecutive iterations of phase one. However, while this approach works for small values of k, it is inefficient for large values of k. In Section VI we discuss several approaches to reduce the value of k by improving the test setup and to reduce the complexity of the search for the changed bit positions. Before we describe our attack in detail, we explain the underlying requirements and building blocks.

Controlling the remanence decay. An essential requirement for our attack is that the adversary $\mathcal{A}$ can precisely control the remanence decay in the SRAM. There are two approaches how this can be achieved. The *voltage-based* approach directly decreases the supply voltage to the chip for a certain amount of time τ, while the *time-based* approach sets the supply voltage of the chip to 0 V for a precisely-measured amount of time t. In general, the time-based approach is easier to use since it only requires a precise timer to trigger the voltage drop, while the voltage-based approach requires an expensive precision DC power source. Next, we present results for both approaches.

Data remanence experiment. One major building block of our attack is the data remanence experiment where the adversary $\mathcal{A}$ observes how the remanence decay affects the behavior of the device containing the PUF.

Definition 5 (Data remanence experiment)**.** *Consider a device that overwrites the memory used by the PUF with some known data. Let v_{nom} be the nominal supply voltage of the device. Let $\vec{M}_{\text{PUF}}$ (Definition 3) be the PUF state and $\vec{M}_{\text{init}}$ (Definition 1) be the initial state of the device memory. Further, let* Dev *(Definition 4) be the algorithm describing the device behavior with respect to some start-up state $\vec{M}_t$ or $\vec{M}_v$ (Definition 2).*
The time-based data remanence experiment $X = \mathsf{DRE}(\vec{M}_{\text{init}}, t, Q)$ *is as follows:*

1) *Set the memory content of the device to $\vec{M}_{\text{init}}$*
2) *Temporarily set the supply voltage of the device to 0 V for time t and then set it back to v_{nom}*
3) *Send the query Q to the device and observe its response $X = \mathsf{Dev}(\vec{M}_t, Q)$*

Further, we define the voltage-based data remanence experiment $X = \mathsf{DRE}(\vec{M}_{\text{init}}, v, Q)$ *as follows:*

1) *Set the memory content of the device to $\vec{M}_{\text{init}}$*
2) *Temporarily set the supply voltage of the device to $v < v_{\text{nom}}$ for a constant time τ and then set it back to v_{nom}*
3) *Send the query Q to the device and observe its response $X = \mathsf{Dev}(\vec{M}_v, Q)$*

Finder algorithm. Another building block of our attack is the finder algorithm, which recovers the PUF state based on the device behavior observed in a series of data remanence experiments.

Definition 6 (Finder algorithm)**.** *Let $\vec{M}_{i+1}$ and $\vec{M}_i$ be two start-up states (Definition 2) that consist of n bits and that differ in at most $k < n$ bits, i.e., the Hamming distance $\mathsf{dist}(\vec{M}_i, \vec{M}_{i+1}) \leq k$. Further, let $X_{i+1} = \mathsf{Dev}(\vec{M}_{i+1}, Q)$ for some arbitrary device query Q. A* finder algorithm *is a probabilistic polynomial time algorithm $\mathsf{Finder}(\vec{M}_i, Q, X_{i+1})$ that returns $\vec{M}_{i+1}$.*

The finder is most efficient when $\mathsf{dist}(\vec{M}_i, \vec{M}_{i+1})$ is minimal, ideally one. In this case, Finder can recover an unknown n-bit start-up state $\vec{M}_{i+1}$ from $\vec{M}_i$ and X_{i+1} by performing a simple exhaustive search with linear complexity in n. However, $\mathsf{dist}(\vec{M}_i, \vec{M}_{i+1})$ is typically larger than one since multiple SRAM cells may have similar remanence decay behavior, decay at the same time/voltage, while other SRAM cells may be metastable (i.e., take a random value) [29], [21], [30], [31]. In the worst case, where up to k bits have changed in a start-up state with n bits, a trivial finder performing an exhaustive search may require up to $\sum_{\ell=1}^{k} \binom{n}{\ell}$. Observe that n typically is a fixed system parameter while k strongly depends on the quality of the equipment used for controlling the remanence decay in the SRAM. As we discuss in Section VI, the adversary can reduce k significantly by using more accurate equipment and he may also use a Finder algorithm that is more efficient than the trivial approach.

Algorithm 1 Extracting the PUF state of an SRAM PUF-enabled device (time-based approach)

Consider a device that uses the same SRAM for the PUF and some other functionality. Let $\vec{M}_{\text{init}}$ be the initial state (Definition 1) and t_∞ be the decay time (cf. Definition 3) of the device memory. Further, let Δt be the difference between the power-off times used in two consecutive time-based DRE experiments (cf. Definition 5). Further, let i and f be indices. The attack works as follows:

1) Fix an arbitrary device query Q
2) Record $X_{\text{PUF}} = \mathsf{DRE}(\vec{M}_{\text{init}}, t_\infty, Q)$
3) Set $i \leftarrow 0$ and $t_0 = 0$
4) Repeat:
   a) Set $i \leftarrow i + 1$
   b) Set $t_i = t_{i-1} + \Delta t$
   c) Record $X_i = \mathsf{DRE}(\vec{M}_{\text{init}}, t_i, Q)$
   d) Stop when $X_i = X_{\text{PUF}}$ and set $f = i$
5) Set $i \leftarrow 0$ and $\vec{M}_{t_0} = \vec{M}_{\text{init}}$
6) Repeat:
   a) Set $i \leftarrow i + 1$
   b) Compute $\vec{M}_{t_i} = \mathsf{Finder}(\vec{M}_{t_{i-1}}, Q, X_i)$
   c) Stop when $i = f$
7) Return M_{t_f}

Details of the attack. The attack is detailed in Algorithm 1 on the example of the time-based approach and works as follows. The adversary $\mathcal{A}$ chooses an arbitrary device query Q (Step 1) and records the response X_{PUF} generated by the device using the PUF state $\vec{M}_{\text{PUF}}$ (Step 2). Then, $\mathcal{A}$ performs a series of time-based DRE experiments (Definition 5) where he slightly increases the power-off time t_i used in each experiment (Steps 3 and 4).[1] This way, $\mathcal{A}$ obtains a sequence of device responses $X_1, \ldots, X_f$ to the same query Q generated by the device using the start-up states $\vec{M}_{t_1}, \ldots, \vec{M}_{t_f}$, respectively, where $\mathsf{dist}(\vec{M}_{t_i}, \vec{M}_{t_{i+1}})$ for all $1 \leq i \leq (f-1)$ is upper bounded by some value k. Observe that $\vec{M}_{t_0} = \vec{M}_{\text{init}}$ is the initial state (Definition 1) and $\vec{M}_{t_f} = \vec{M}_{\text{PUF}}$ is the PUF state (Definition 3) of the SRAM. Next, $\mathcal{A}$ uses the Finder algorithm (Definition 6) to iteratively recover $\vec{M}_{\text{PUF}}$ from the device responses observed in Steps 3 to 4. Specifically, starting from the known initial state $\vec{M}_{t_0} = \vec{M}_{\text{init}}$, the adversary iteratively recovers each $\vec{M}_{t_{i+1}}$ from $\vec{M}_{t_i}$ and X_{i+1} until $\mathcal{A}$ arrives at the PUF state $\vec{M}_{t_f} = \vec{M}_{\text{PUF}}$.

[1] An adversary using the voltage-based approach would gradually lower the supply voltage (for a fixed amount of time) instead of increasing the power-off time.

Theorem 1 (Success of the attack)**.** *The attack in Algorithm 1 successfully recovers the PUF state $\vec{M}_{\text{PUF}}$. The worst case complexity of the attack when using a trivial* Finder *algorithm (Definition 6) is $f \cdot \sum_{\ell=1}^{k} \binom{n}{\ell}$, where f is the number of* DRE *experiments (cf. Definition 5), n is the size of the SRAM, and k is the maximum Hamming distance of the start-up states $\vec{M}_{t_i}$ and $\vec{M}_{t_{i+1}}$ used by the device in two consecutive* DRE *experiments for all $1 \leq i \leq (f-1)$.*

Note that the complexity of the attack strongly depends on the value of k, which highly depends on the accuracy of the equipment and method used to control the remanence decay in the SRAM. Typical values are $k = 0.1469 \cdot n$ for the time-based approach and $k = 0.1004 \cdot n$ for the voltage-based approach (cf. Section IV). Moreover, in our experiments for the time-based approach we observed a decay time of $t_\infty = 2{,}000\ \mu s$ and used $\Delta t = 1\ \mu s$, resulting in $f = \lceil 2{,}000\ \mu s / 1\ \mu s \rceil = 2{,}000$. For the *voltage-based* approach we observed remanence decay for supply voltages between $0.4\ V$ and $0\ V$ and used $\Delta v = 2\ mV$, resulting in $f = \lceil 400\ mV / 2\ mV \rceil = 200$.

Proof of Theorem 1: It follows from Definition 5 that $X_{\text{PUF}} = \mathsf{Dev}(\vec{M}_{t_\infty}, Q)$ and from Definition 3 that $\vec{M}_{t_\infty} = \vec{M}_{\text{PUF}}$. Hence, in Step 2, X_{PUF} is the response of the device using the PUF state. Furthermore, it follows from Definition 5 that $X_i = \mathsf{Dev}(\vec{M}_{t_i}, Q)$ in Step 4(c). Hence, after Step 5 we have obtained a sequence of device responses $X_0, \ldots, X_f$ that correspond to the memory states $\vec{M}_{t_0}, \ldots, \vec{M}_{t_f}$. Due to the different decay times of the individual SRAM cells and the metastability in the SRAM, two memory states $\vec{M}_{t_i}$ and $\vec{M}_{t_{i+1}}$ differ in at most $k < n$ bits. Hence, $\mathsf{dist}(\vec{M}_{t_i}, \vec{M}_{t_{i+1}}) \leq k$ and it follows from Definition 6 that $\mathsf{Finder}(\vec{M}_{t_{i-1}}, Q, X_{i-1}) = \vec{M}_{t_i}$ in Step 6(b). By definition it holds that $\vec{M}_{t_0} = \vec{M}_{\text{init}}$ and by induction over i it follows that $\vec{M}_{t_f} = \vec{M}_{\text{PUF}}$ in Step 7.

It remains to show the complexity of the attack. In the worst case, Finder performs an exhaustive search over all $\sum_{\ell=1}^{k} \binom{n}{\ell}$ possible positions of the up to k bits in which the n-bit state $\vec{M}_{t_{i+1}}$ may differ from $\vec{M}_{t_i}$. This means that in the worst case Finder must verify $\sum_{\ell=1}^{k} \binom{n}{\ell}$ guesses to find the correct memory state $\vec{M}_{t_i}$ in each of the f iterations of Step 6(b). This leads to an overall attack complexity of $f \cdot \sum_{\ell=1}^{k} \binom{n}{\ell}$, which finishes the proof. ∎

IV. Experimental Validation of the Attack

In order to reduce the complexity of the attack, it is required that only a small number k of SRAM cells in two consecutive DRE experiments change their

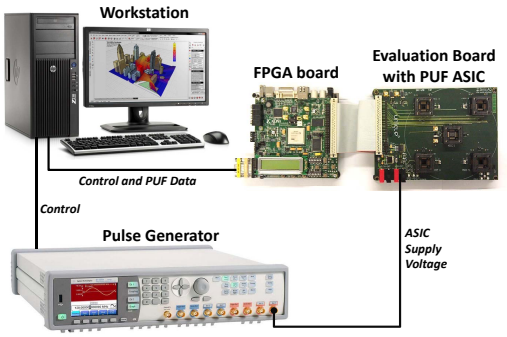

Figure 1: Test setup for the time-based approach using an Agilent 81150 pulse generator

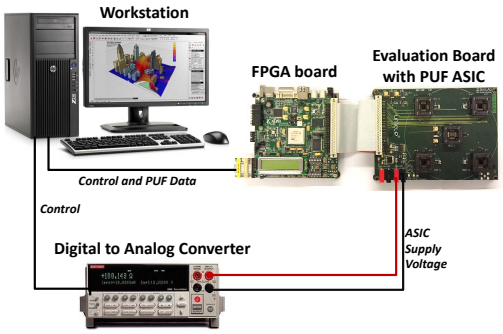

Figure 2: Test setup for the voltage-based approach using a Keithley 2400 sourcemeter

states during the transition from the known (initial) state to the final PUF state. This number k is mainly controlled by two factors: (1) the accuracy of the equipment used to control the remanence decay of the memory during the attack and (2) the number of metastable SRAM cells, i.e., those that take random states. In this section, we investigate the impact of both factors on the remanence decay in the SRAM PUFs implemented in two 65 nm CMOS ASICs. Our evaluation uses both the time-based and the voltage-based approach to control the remanence decay.

Test setup. Our analysis is based on data obtained from two ASICs that have been manufactured in TSMC 65 nm CMOS technology within an Europractice multi-project wafer run. The ASIC has been designed within the UNIQUE[2] research project. Each ASIC implements four different SRAM PUF instances, each using 8 kBytes of SRAM. The test setup consists of an ASIC evaluation board, a Xilinx Virtex 5 FPGA board, a power supply; either an Agilent 81150 pulse/function/arbitrary pulse generator for the time-based approach or a Keithley 2400 general-purpose sourcemeter for the voltage-based approach, and a workstation (Figures 1 and 2). The evaluation board allows controlling the ASIC supply voltage using an external power supply. In each experiment, we wrote a pre-determined bit pattern (i.e., all ones) to the SRAM, used the pulse generator or sourcemeter to deliver a temporary voltage drop with precisely controlled width and amplitude and finally read back the memory contents of the SRAM. The rated accuracy of the Agilent 81150 pulse generator has a temporal resolution of 5 ns and an amplitude resolution of 25 mV. The Keithley 2400 sourcemeter has a basic accuracy of 50 μV.

To accelerate and simplify the remanence decay process, we did not place any decoupling capacitors between the pulse generator/sourcemeter's output and the ASIC's supply voltage input, as shown in [21], the effect of such a capacitor in the time-based approach is an increase in the time between the power-down event and the beginning of remanence decay. In the voltage-based approach, the capacitor increases the time required for the ASIC's power input to converge to the required value, again resulting in a slowdown of the remanence decay process. The interaction with the evaluation board and the ASICs is performed by the FPGA, which is connected to a workstation that controls the PUF evaluation and the pulse generator or the sourcemeter. Further, the workstation is used to process and store the data obtained from the ASICs. All tests with the Agilent 81150 pulse generator were performed at room temperature (approx. 25°C) in an air conditioned laboratory. The tests with the Keithley sourcemeter were performed in a refrigerator at temperatures between 2.7° and 7.6°C, However, in the time-based approach, we wanted to capture the effect of power-off time on data remanence without controlling the ambient temperature, as it is already known that some attacks use low temperatures to decelerate data remanence of SRAM cells [20], [32].

Chip-scale modeling. The purpose of this experiment was to observe and to reproduce the decay behaviour reported in [21] and to gauge its stability and reproducibility for the SRAM PUF for the time-based and the voltage-based approach.

[2] http://www.unique-project.eu/

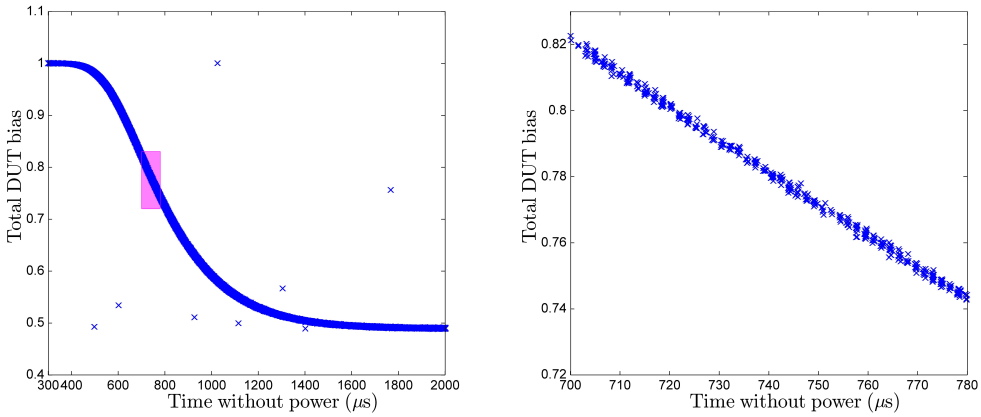

Figure 3: Chip-scale view of remanence decay (time-based approach)

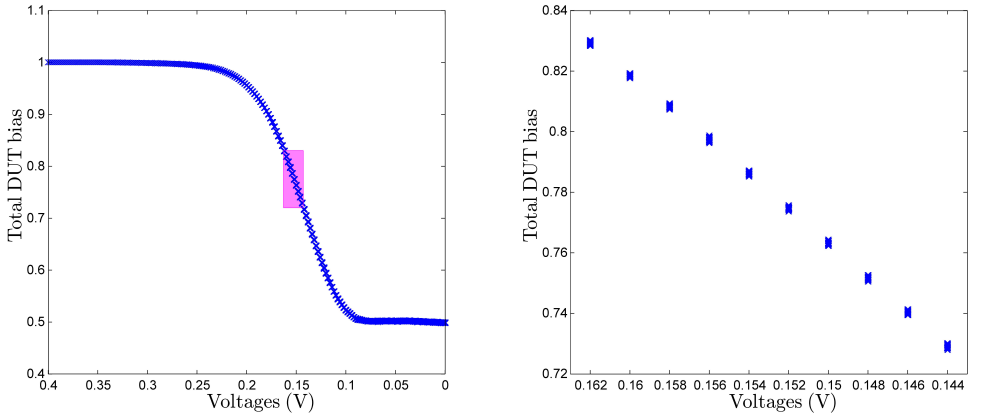

Figure 4: Chip-scale view of remanence decay (voltage-based approach)

Time-based approach. A series of 10,000 data remanence experiments with an initial state $\vec{M}_{\text{init}}$ consisting of only ones was performed. Each experiment was repeated 10 times with 1,000 different power-off times t between 300 μs and 2,000 μs. During the power-off time the supply voltage was set to 0 V. After each experiment we measured for each SRAM cell the probability that it still stores the value we wrote to it before the power cycle. We call this probability the *bias* of the cell.

Voltage-based approach. We performed 30 series of data remanence experiments with an initial state $\vec{M}_{\text{init}}$ consisting of only ones. Each series consists of 201 experiments, where the voltage was dropped by 2 mV for 1 ms starting at 0.4 V, as the experiments show no decay to zero in the range between 1.2 V and 0.4 V, and then set back to the default supply voltage of 1.2 V. For each experiment, we measured the probability that each SRAM cell preserves the value written to it before the power cycle.

Chip-scale results. Our results are depicted in Figure 3 and Figure 4. The graphs on the right represent zoomed-in portions of the graphs on the left. In both figures, the y-axis corresponds to the mean bias over all SRAM cells, while the x-axis corresponds to the total *time* the ASIC was without power in Figure 3 or to the *voltages* applied to the ASIC Figure 4. Each cross in the graphs corresponds to a single experiment. As shown in the left graph in Figure 3, the average bias over all SRAM cells decays very reliably from 1 to the expected 0.5 [30], [31] during the course of 2 ms, while the left graph in Figure 4 exhibits the average bias over all SRAM cells decaying from 1 to 0.5 in the range of 0.4 V to 0 V. The results are also compatible with the findings in [33], [21], i.e., the typical values of DRV fall in the range between 80 mV and 250 mV.

As the detailed views on the right of Figure 3

and Figure 4 show that, there is a small variation in the measured bias between identical experiments, which was either due to the physical limitations of our test setup or due to those SRAM cells exhibiting metastability.

Bit-scale modeling. The next experiment investigates whether the individual SRAM cells have different transition times and transition voltages, which is required in our attack. With the *transition time* (resp. *transition voltage*) of an SRAM cell we mean the point in time (resp. voltage level) where the cell loses the value that has been written to it and reverts to its PUF state. Based on the results of the previous experiment, we estimated the bias of each SRAM cell over time.

Bit-scale results. Figure 5 and Figure 6 display 2-D contour plots of the cell-level behaviour of the SRAM PUF. Again, the graphs on the right represent zoomed-in portions of the graphs on the left. Each horizontal row in the graph corresponds to the bias of a single SRAM cell, selected out of 1000 representative cells whose final bias were close to zero. We only selected cells with final bias close to zero since the cells with a final bias close to one will not show any decay behavior in our experiment where we wrote a logical one to all memory cells before the power cycle. For the purpose of legibility, the cells were sorted in the graph by their transition time/voltage. The left and right gray lines on the graphs correspond to times in Figure 5 and voltages in Figure 6, when the bias of each bit is one and zero, respectively. The black line corresponds to the time/voltage when the bias of each bit is 0.5.

As shown in Figure 5, each individual SRAM cell has a different remanence decay time surrounded by a short period of metastability in which the cell may enter both states. The median metastability period measured was 30 μs and the worst-case metastability rate was 14%. Figure 6 shows each individual SRAM cell has a different remanence decay voltage with a voltage band on either side exhibiting metastability. The median metastability voltage was 4 mV and the worst-case metastability rate was 10%, which is 28% lower than the time-based result. In general, the maximum size of a PUF that can be attacked using our methodology is limited by the metastability, as we discuss further in Section VI.

A detailed look at the evolution of the bias of a single bit over time and voltage is shown in Figure 7 and Figure 8 respectively, and the small spikes which can be noted in the otherwise monotonic plots are probably the result of noise encountered when measuring the bit in its metastable state.

Cross-device comparison. Next we investigated whether the transition times and transition voltages of the SRAM cells in one device allow one to infer some information on the transition times and transition voltages of the SRAM cells in another device. A second goal of this experiment was to get a first impression of whether the transition times and transition voltages in SRAM cells could be used to identify individual SRAM chips, an idea we discuss in Section VI. In this experiment, we measured the bias over time and the transition times and transition voltages of each SRAM cell in two ASICs. Again, we considered only cells whose PUF state is zero.

Cross-device results. The results for the time-based approach as well as voltage-based approach are shown in Figure 9 and Figure 10 respectively. Each cross in the graphs corresponds to the bias of a single SRAM cell. In both figures, the x-coordinate of each point is the transition time/voltage of the SRAM cell on the first ASIC, while the y-coordinate is the transition time/voltage of the same SRAM cell on the second ASIC. As Figure 9 shows, the transition times of the two ASICs are virtually uncorrelated, which we confirmed by computing the normalized cross correlation ρ of both data sets, which is $\rho = 0.002$. Our results are in line with the findings by Holcomb et al. [25] who also suggest using the remanence decay behaviour as a source of unique information to identify individual devices. Figure 10 also confirms the same conclusion, i.e. the transition voltages of the two ASICs are uncorrelated, which is confirmed by the normalized cross correlation ρ of both data sets $\rho = 0.0012$.

Time-Based vs. Voltage-Based Attacks. The evaluation results in the previous sections confirm results in the literature [33], [21] and show that the voltage-based approach is less sensitive to temperature variations, making it potentially more effective in an attack than the time-based approach. Our results are summarized in Table I, which shows that using the voltage-based approach results in a significantly lower metastability rate than using the time-based approach. This means that a voltage-based attack may still be effective in situations where the time-based attack fails. An interesting observation is that the set of metastable SRAM cells in voltage-based approach shows 28% improvement over time-based approach, which indicates that most of the inaccuracies in our experiments are due to the limitations of our test setup and not due the physical properties of the SRAM PUF itself.

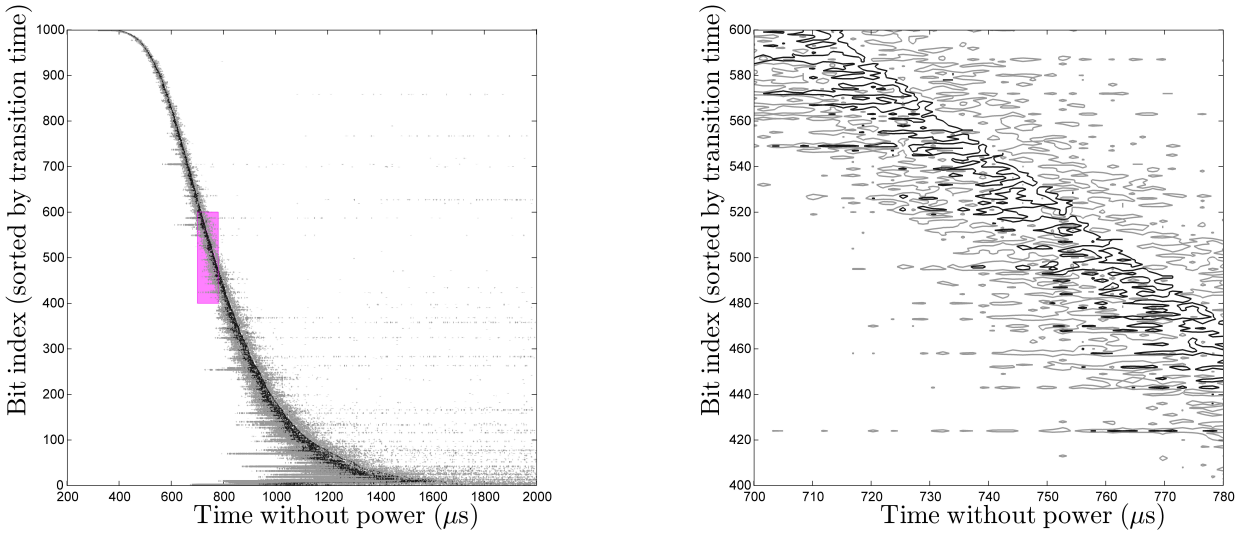

Figure 5: Bit-scale view of remanence decay (time-based approach)

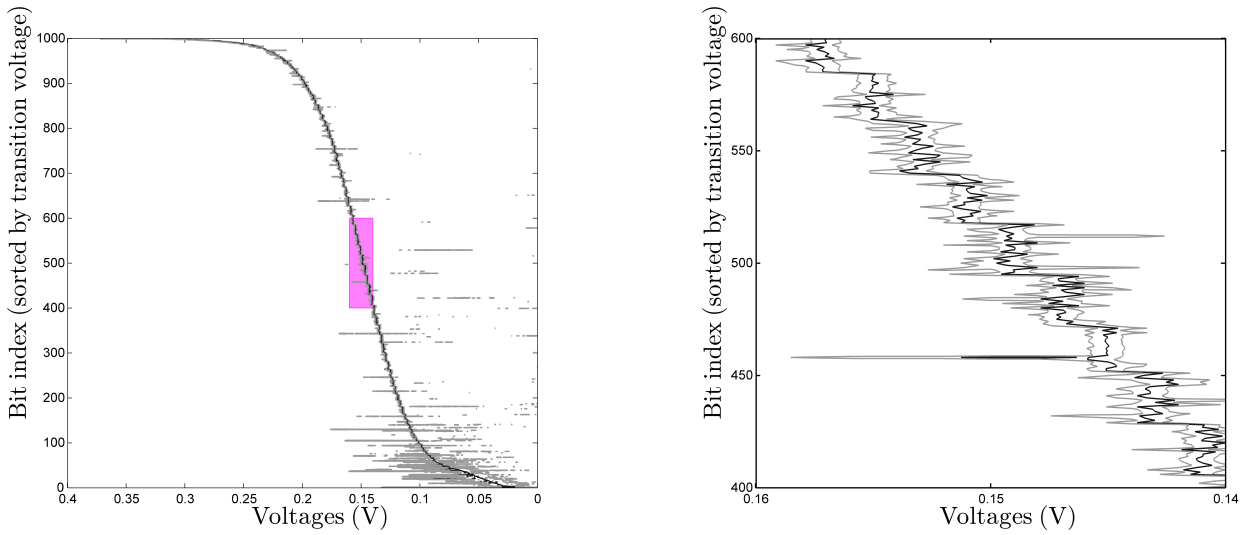

Figure 6: Bit-scale view of remanence decay (voltage-based approach)

V. Effectiveness of the attack in practical settings

To investigate the effectiveness of our attack in a practical setting, we created a standard implementation of an SRAM PUF-based authentication scheme. This scheme uses a standard secret-key-based challenge-response protocol and derives the underlying key from the PUF response using a basic repetition code [34].[3] In more detail, during the enrollment of the device, the memory addresses of those 128 SRAM bytes whose PUF state is highly biased (i.e., that have a Hamming weight of 0, 1, 7, or 8) are stored as the public helper data, each representing

[3]We omit the linear encoding used in [34] and the privacy amplification typically used in PUF-based key storage since it has no effect on our attack.

Table I: Comparison of Voltage-Based and Time-Based Remanence

Remanence control	Voltage-based	Time-based
Bits stable at 1	44.82%	43.45%
Bits stable at 0	45.14%	41.86%
Metastability rate (worst case)	10.04%	14.69%

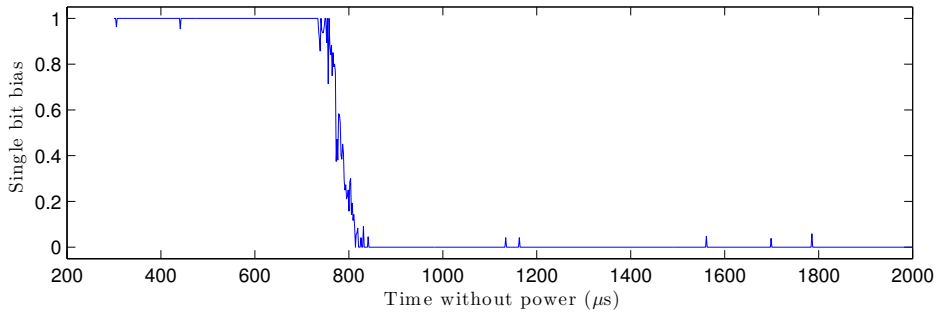

Figure 7: Close-up look at a single bit for the time-based approach

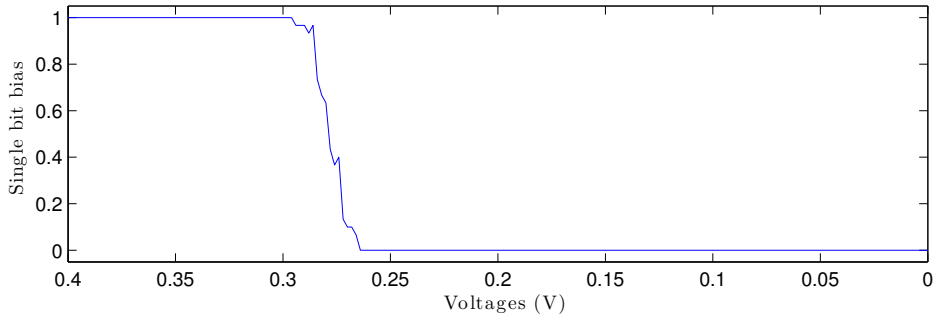

Figure 8: Close-up look at a single bit for the voltage-based approach

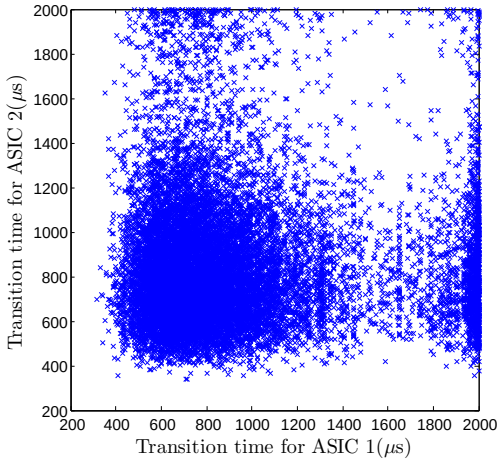

Figure 9: Correlation between the transition time in two different devices

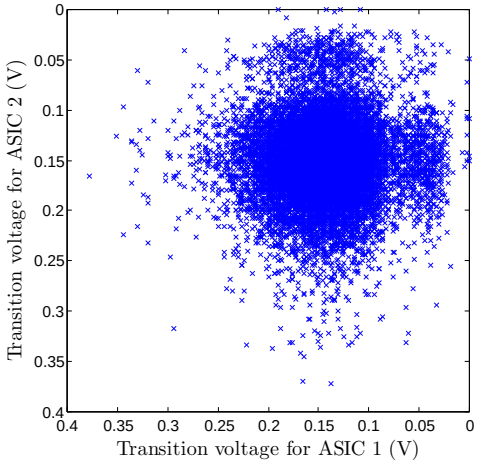

Figure 10: Correlation between the transition voltage in two different devices

one bit of the secret key stored in the PUF. The key is reconstructed from the PUF as follows. The 128 SRAM bytes whose addresses are stored in the helper data are read from the SRAM and the value of each bit in the key is set as the result of a simple majority voting over all bits in the respective byte to ensure that the 128 bits key derived from the 128 highly-biased SRAM bytes are stable. Even though the bits participating in the majority voting (and their associated transition times/voltages) are spatially averaged, there is still a single point of time/voltage for each output bit where a majority of its constituent bits transit to the PUF state. The bit-flips before this transition period are absorbed

by the error-correcting code and can be ignored by the attacker. This is a simple example of the general behaviour of error correcting codes, where the output symbol changes only after a sufficient number of errors has accumulated. The resulting secret key K is then used in the secret key-based challenge-response protocol, i.e. $X = \mathsf{MAC}_K(Q)$, where MAC is a Message Authentication Code.

The attack is as in Definition III. However, we use an optimized Finder algorithm (Definition 6) that only searches for key candidates with a Hamming distance less than 10 bits from the previous key, which significantly improves the performance of the attack compared to the trivial Finder described in Definition 6.

The overall running time of the attack is estimated as $2^{53.6}$ MAC operations. Considering that modern CPUs can perform 2^{31} AES operations per second [35], [36], the total cost of the attack on an AES-based MAC is $2^{22.6}$ CPU-seconds, or approximately two CPU-months. The attack can easily be parallelized by testing multiple key candidates simultaneously, making the attack even more practical for moderately-funded adversaries.

VI. Impact of Our Attack and Countermeasures

Impact. Our results in Section IV show that by carefully controlling the power-off times or the supply voltage of the SRAM PUF, one can reliably control the number of metastable bits as required by the attack described in Section III. This means that, even if we use the trivial finder algorithm discussed in Definition 6, common lab equipment and the less effective time-based approach to control the remanence decay in the SRAM, we can recover a 216-bit SRAM PUF derived key by making at most 2^{64} calls to the Dev algorithm (cf. Definition 4). Using the voltage-based approach with the same finder algorithm and equipment as in the time-based approach, we can extract the response of a 315-bit SRAM PUF derived key in the same time. Further, our results in Section V show that, depending on the post-processing of the PUF responses, our attack can also be applied to systems using larger PUFs. Hence, it is problematic to overwrite the memory of an SRAM PUF with a known value, which, however, is required when the PUF memory is also used for other purposes, as suggested in many prior works [8], [1], [11], [16], [17], [18], [19]. This particularly holds for resource-constrained devices with only small amounts of SRAM, such as RFIDs or medical implants [8], [11], [16], where SRAM PUFs without shared memory are impractical.

Improving the attack. One approach to lower the complexity of our attack is using more accurate equipment that allows a very precise control of the remanence decay in the SRAM using the voltage-based approach, which limits the number of metastable bits and the complexity of the finder algorithm (cf. Definition 6).

Furthermore, several optimizations of the finder algorithm are possible: The order in which the individual SRAM cells transition from their initial state to their PUF state is different for the time-based and the voltage-based approach (cf. Section V). Further, in some scenarios the adversary may be able to control the initial state of the SRAM. This results in four different ways to observe the decay behavior of each SRAM cell and allows the adversary to choose the way with the lowest metastability rate for his attack, which can significantly reduce the complexity of the naïve finder algorithm (cf. Definition 6).

Another approach to improve the complexity of the finder algorithm is to take advantage of the algorithms used by the device to process the PUF responses (cf. Section V). These algorithms typically include an error correction mechanism such as a fuzzy extractor [37], which helps to maintain the consistency of the PUF response to the same challenge under different environmental variations affecting the underlying physical object. Due to this error correction the device response changes only when the error correction mechanism fails, assuming that the fuzzy extractor generates some key, a wrong key, for any arbitrary *state* $\vec{M}$ that differs from $\vec{M}_{\mathrm{PUF}}$ with more t bits, which fuzzy extractor can correct. Hence, the finder algorithm needs to consider only one single candidate of each codeword class. This can either be done explicitly by considering the structure of the error correcting code or by casting the problem as an optimization problem and using an optimizer [38].

Countermeasures. There are several countermeasures that prevent our attack by breaking the underlying assumptions but that are impractical in low-resource scenarios such as RFIDs and sensors [8], [11], [16]. One approach to prevent the attack described in Section III is using an additional memory that can only be accessed by the PUF. However, this contradicts the idea of using the existing memory of the device and significantly increases implementation costs. Another approach is to wait until any value stored in the memory has decayed before reading the PUF response. However, this requires the device to have some notion of time and increases the boot time, which can be problematic in some applications. Further, the attack can be prevented by designing the

algorithms processing the PUF response such that the device behavior for different start-up states is indistinguishable by the adversary. However, this might imply the use of complex cryptographic primitives for authentication schemes that exceed the capabilities of *resource-constrained* devices for which PUFs with shared memory have been proposed [8], [11], [16].

VII. Constructive Use of Remanence Decay

Device authentication. The remanence decay behavior can be used to authenticate an SRAM to some verifier. Specifically, using the same approach as in our attack, a verifier could force the SRAM into a partially reverted state by, e.g., writing some value to the SRAM and then powering the device off for a carefully controlled amount of time. Since the verifier knows the (secret) PUF state of the SRAM and the decay behavior of the genuine device, he can determine the partially reverted SRAM state and check whether it matches the expected state of the SRAM to be authenticated. Care must be taken that this additional functionality does not expose the device to our attack, for example by requiring that the verifier successfully authenticates to the device before being allowed to write to the SRAM.

Note that, it is much more difficult to clone such an SRAM PUF since the clone must emulate the SRAM decay behavior, which requires the clone to contain a time-keeping mechanism that raises its costs. Our results suggest that for an SRAM of size n bits there are $\log(n!)$ bits of entropy encoded in the *order* at which individual SRAM cells revert to their PUF state. However, further evaluations are needed to assess the practicality of this approach, in particular the temperature-dependency and the effect of aging on the decay behavior must be investigated.

Improving the TARDIS time-keeping algorithm. The use of SRAM remanence decay has recently been proposed as a time-keeping mechanism for clockless low-power devices, such as passive RFID tags [21]. This mechanism, called TARDIS, allows a clockless device to estimate how much time has passed since its last power-down and aims to impede oracle attacks. TARDIS consists of two main elements: the Init algorithm which sets all SRAM cells to a fixed value (all ones) and the Decay algorithm which determines how long the device has been without power based on the number of ones that are still stored in the SRAM. Observe that the Init algorithm requires a one to be written to each cell of the SRAM, while the Decay algorithm must read the value of each cell while the device is booting. These two operations consume a non-negligible amount of power and add 15.2 ms to the start-up time of the device.

Our observations on the behaviour of remanence decay can be used to dramatically improve the performance of the TARDIS system. As our results show, the transition time of each bit is uniquely determined by its individual DRV. By profiling the SRAM in an offline phase, we can thus determine the *order* in which the SRAM cells return to their PUF state and store this ordering in the non-volatile memory of the device. Now, if we observe that a certain group of bits has reverted to its PUF state, we immediately know that all bits which have a lower transition time have also returned to their PUF state. Similarly, if we observe that a certain group of bits is still in its initial state, we immediately know that all bits that have a longer transition time are also still in their initial state. Knowing this ordering, we can replace the linear-time Decay algorithm of [21] with the well known binary search algorithm that takes logarithmic time. To deal with metastability, the algorithm should sample not only one but a group of bits for each transition time period.

If the device needs to detect only whether or not the *entire* SRAM has returned to its PUF state, another improvement is possible that dramatically decreases the running time of both the Init and the Decay algorithms from linear time to constant time. In this case, both algorithms need only to access those SRAM cells that are known to be the *last* to revert to the PUF state.

Since most of the applications described in [21] can be adapted to use this improvement, our results enhance the applicability of the TARDIS system to practical scenarios. We stress that the SRAM used by the TARDIS scheme cannot be used as an SRAM PUF since its content is well-known in this case.

VIII. Related Work

While the impact of remanence decay on the randomness that can be extracted from SRAM cells and the reliability of SRAM PUFs has been discussed in the literature [22], [23], [5], [24], [25], it has never been used as a side channel to attack SRAM PUFs. In fact, some papers investigate side channel attacks in the context of PUFs, mainly focusing on the side channel leakage of the algorithms processing the PUF response [39], [40] or proposing combinations of side channels attacks on PUFs with modeling or fault injection attacks [41], [42], [43]. The impact of environmental changes on the repeatability of PUF response has been introduced as a source of fault injection attack on arbiter and RO PUFs in [44], and current-based PUFs in [45], while the same impact

has been evaluated in [30], [31] for memory-based PUFs, however no results on fault injection attacks have been reported. In contrast, to the best of our knowledge, we present the first cloning attack that injects faults into the SRAM PUF and uses the data remanence effect in SRAM as a side channel to recover the (secret) PUF response.

It has been shown that SRAM PUFs can be emulated and physically cloned. By physically inspecting the SRAM hardware the adversary learns information that helps emulating the PUF [46]. Further, it has been shown that after learning the response of an SRAM PUF p_1, a focussed ion beam (FIB) can be used to modify the circuits of the SRAM cells of another SRAM PUF p_2 so that p_2 shows a very similar challenge/response behavior as p_1 [47].

Data remanence in DRAM has been used to extract security-sensitive data from the random access memory of PCs and workstations [20]. While these attacks aim to recover some data that has been written to an unprotected memory, the goal of our attack is to recover the start-up pattern of an SRAM PUF that is typically protected by some kind of access control mechanism.

IX. Conclusion

We demonstrated a simple non-invasive cloning attack on SRAM PUFs using remanence decay as a side-channel and validated its feasibility on 8 KBytes SRAM PUFs instantiated on two 65 nm CMOS devices. Our attack and evaluation is general and can be optimized for concrete systems. Our evaluation results show that even without optimizations, attacks on small SRAM PUFs are feasible using common lab equipment. We discussed countermeasures against our attack and suggest using remanence decay to improve the cloning-resistance of SRAM PUFs. As a contribution of independent interest, we showed how our evaluation results can be used to improve the performance of TARDIS [21], a recently proposed time-keeping mechanism for clockless devices.

We investigated both the time-based approach and the voltage-based approach to control the data remanence decay in the SRAM. Our results show that the voltage-based approach is more promising than the time-based approach. Directions for future work include the design of non-trivial finder algorithms that, e.g., exploit the properties of the algorithms used by the device processing the PUF response.

Acknowledgements

We thank Ünal Kocabaş for preparing the lab experiments in the first phase of this work. The development and manufacturing of the PUF ASIC used in this work has been supported by the European Commission under grant agreement ICT-2007-238811 UNIQUE. This work has been co-funded by the DFG as part of project P3 within the CRC 1119 CROSSING.

References

[1] J. Guajardo, S. S. KuMar., G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," in *Field Programmable Logic and Applications (FPL)*. IEEE, 2007, pp. 189–195.

[2] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic PUFs from flip-flops on reconfigurable devices," in *Benelux Workshop on Information and System Security*, 2008.

[3] Y. Su, J. Holleman, and B. P. Otis, "A digital 1.6 pJ/bit chip identification circuit using process variations," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 69–77, 2008.

[4] S. S. KuMar., J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *Workshop on Hardware-Oriented Security (HOST)*. IEEE, June 2008, pp. 67–70.

[5] D. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2009.

[6] V. van der Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls, "Hardware intrinsic security from D flip-flops," in *ACM Workshop on Scalable Trusted Computing (ACM STC)*. ACM, 2010, pp. 53–62.

[7] D. Lim, J. W. Lee, B. Gassend, E. G. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[8] P. Tuyls and L. Batina, "RFID-tags for anti-counterfeiting," in *Topics in Cryptology (CT-RSA)*, ser. LNCS, vol. 3860. Springer, 2006, pp. 115–131.

[9] J. Guajardo, S. S. KuMar., G.-J. Schrijen, and P. Tuyls, "Brand and IP protection with physical unclonable functions," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2008, pp. 3186–3189.

[10] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, 2010.

[11] A.-R. Sadeghi, I. Visconti, and C. Wachsmann, "Enhancing RFID security and privacy by physically unclonable functions," in *Towards Hardware-Intrinsic Security*, ser. Information Security and Cryptography. Springer, 2010, pp. 281–305.

[12] I. Eichhorn, P. Koeberl, and V. van der Leest, "Logically reconfigurable PUFs: Memory-based secure key storage," in *ACM Workshop on Scalable Trusted Computing (ACM STC)*. ACM, 2011, pp. 59–64.

[13] Verayo Inc, "Product webpage," http://www.verayo.com/product/products.html, 2013.

[14] Intrinsic ID, "Product webpage," http://www.intrinsic-id.com/products.htm, 2013.

[15] NXP Semiconductors N.V., "NXP strengthens SmartMX2 security chips with PUF anti-cloning technology," 2013.

[16] J. Guajardo, M. Asim, and M. Petković, "Towards reliable remote healthcare applications using combined fuzzy extraction," in *Towards Hardware-Intrinsic Security*, ser. Information Security and Cryptography. Springer, 2010, pp. 387–407.

[17] S. Kardas, M. S. Kiraz, M. A. Bingol, and H. Demirci, "A novel RFID distance bounding protocol based on physically unclonable functions," in *Radio Frequency Identification: Security and Privacy Issues (RFIDSec)*, ser. LNCS. Springer, June 2011.

[18] P. Koeberl, J. Li, A. RaJan., C. Vishik, and W. Wu, "A practical device authentication scheme using SRAM PUFs," in *Conference on Trust and Trustworthy Computing (TRUST)*, ser. LNCS, vol. 6740. Springer, June 2011, pp. 63–77.

[19] P. Koeberl, J. Li, R. Maes, A. RaJan., C. Vishik, and M. Wójcik, "Evaluation of a PUF device authentication scheme on a discrete 0.13μm SRAM," in *International Conference on Trusted Systems (INTRUST)*, ser. LNCS, vol. 7222. Springer, 2012, pp. 271–288.

[20] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest we remember: Cold-boot attacks on encryption keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, May 2009.

[21] A. Rahmati, M. Salajegheh, D. Holcomb, J. Sorber, W. P. Burleson, and K. Fu, "TARDIS: Time and remanence decay in SRAM to implement secure protocols on embedded devices without clocks," in *USENIX Security Symposium*. USENIX Association, 2012, pp. 36–52.

[22] C. Tokunaga, D. Blaauw, and T. Mudge, "True random number generator with a metastability-based quality control," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 78–85, 2008.

[23] N. Saxena and J. Voris, "We can remember it for you wholesale: Implications of data remanence on the use of RAM for true random number generation on RFID tags (RFIDSec 2009)," http://arxiv.org/abs/0907.1256, July 2009.

[24] G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. de Groot, V. van der Leest, G.-J. Schrijen, M. van Hulst, and P. Tuyls, "Evaluation of 90nm 6T-SRAM as physical unclonable function for secure key generation in wireless sensor nodes," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 567–570.

[25] D. E. Holcomb, A. Rahmati, M. Salajegheh, W. P. Burleson, and K. Fu, "DRV-fingerprinting: Using data retention voltage of SRAM cells for chip identification," in *Radio Frequency Identification. Security and Privacy Issues*, ser. LNCS, J.-H. Hoepman and I. Verbauwhede, Eds., vol. 7739. Springer, 2013, pp. 165–179.

[26] C.-H. Chen, K. Bowman, C. Augustine, Z. Zhang, and J. Tschanz, "Minimum supply voltage for sequential logic circuits in a 22nm technology," in *Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on*, Sept 2013, pp. 181–186.

[27] Y. Oren, A.-R. Sadeghi, and C. Wachsmann, "On the effectiveness of the remanence decay side-channel to clone memory-based PUFs," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS. Springer, 2013, vol. 8086, pp. 107–125. To appear.

[28] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology (CRYPTO)*, ser. LNCS, vol. 1294. Springer, 1997, pp. 513–525.

[29] D. Holcomb, W. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Workshop on RFID Security (RFIDSec)*, July 2007.

[30] M. Bhargava, C. Cakir, and K. Mai, "Comparison of bi-stable and delay-based physical unclonable functions from measurements in 65nm bulk CMOS," in *Custom Integrated Circuits Conference (CICC)*. IEEE, 2012, pp. 1–4.

[31] S. Katzenbeisser, U. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, vol. 7428. Springer, 2012, pp. 283–301.

[32] S. Skorobogatov, "Low temperature data remanence in static RAM," http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-536.pdf, Tech. Rep., June 2002.

[33] H. Qin, Y. Cao, D. Markovic, A. Vladimirescu, and J. Rabaey, "SRAM leakage suppression by minimizing standby supply voltage," in *International Symposium on Quality Electronic Design (ISQED)*, vol. 0. IEEE, 2004, pp. 55–60.

[34] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGAs," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, vol. 5154. Berlin, Heidelberg: Springer, July 2008, pp. 181–197.

[35] Calomel.org, "Aes-ni ssl performance study," https://calomel.org/aesni_ssl_performance.html, 2015.

[36] Intel, "Aes-ni performance enhancements: Hytrust datacontrol case study," https://software.intel.com/en-us/articles/intel-aes-ni-performance-enhancements-hytrust-datacontrol-case-study, 2015.

[37] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology (EUROCRYPT)*, ser. LNCS, vol. 3027. Springer, May 2004, pp. 523–540.

[38] Y. Oren, M. Renauld, F.-X. Standaert, and A. Wool, "Algebraic Side-Channel attacks beyond the Hamming weight leakage model," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. LNCS, E. Prouff and P. Schaumont, Eds., vol. 7428. Springer, 2012, pp. 140–154.

[39] D. Karakoyunlu and B. Sunar, "Differential template attacks on PUF enabled cryptographic devices," in *Workshop on Information Forensics and Security (WIFS)*. IEEE, 2010, pp. 1–6.

[40] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-channel analysis of PUFs and fuzzy extractors," in *Trust and Trustworthy Computing (TRUST)*, ser. LNCS, vol. 6740. Springer, June 2011, pp. 33–47.

[41] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong pufs," *Cryptology ePrint Archive, Report 2013/632*, 2013. [Online]. Available: http://eprint.iacr.org/2013/632

[42] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson, "Efficient power and timing side channels for physical unclonable functions," in *Cryptographic Hardware and Embedded Systems*, vol. 8731. Springer Berlin Heidelberg, 2014, pp. 476–492.

[43] G. T. Becker and R. Kumar, "Active and passive side-channel attacks on delay based puf designs," http://eprint.iacr.org/2014/287, 2014.

[44] J. Delvaux and I. Verbauwhede, "Fault injection modeling attacks on 65 nm arbiter and ro sum pufs via environmental changes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 61, pp. 1701–1713, June 2014.

[45] R. Kumar and W. Burleson, "Hybrid modeling attacks on current-based pufs," in *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, Oct. 2014, pp. 493–496.

[46] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive PUF analysis," in *Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Aug. 2013, pp. 30–38.

[47] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *Hardware-Oriented Security and Trust (HOST)*, June 2013, pp. 1–6.