Low-temperature data remanence attacks against intrinsic SRAM PUFs

N.A. Anagnostopoulos
Security Engineering Group
TU Darmstadt / CASED
Mornewegstraße 32, 64293
Darmstadt, Hessen, Germany
anagnostopoulos@seceng.informatik.tu-darmstadt.de

Stefan Katzenbeisser
Security Engineering Group
TU Darmstadt / CASED
Mornewegstraße 32, 64293
Darmstadt, Hessen, Germany
katzenbeisser@seceng.informatik.tu-darmstadt.de

Markus Rosenstihl
Institut für Festkörperphysik
TU Darmstadt
Hochschulstraße 6, 64289
Darmstadt, Hessen, Germany
markus.rosenstihl@physik.tu-darmstadt.de

André Schaller
Security Engineering Group
TU Darmstadt / CASED
Mornewegstraße 32, 64293
Darmstadt, Hessen, Germany
schaller@seceng.informatik.tu-darmstadt.de

Sebastian Gabmeyer
Security Engineering Group
TU Darmstadt / CASED
Mornewegstraße 32, 64293
Darmstadt, Hessen, Germany
gabmeyer@seceng.informatik.tu-darmstadt.de

Tolga Arul
Integrated Circuits and Systems Lab
TU Darmstadt / CASED
Mornewegstraße 32, 64293
Darmstadt, Hessen, Germany
tolga.arul@cased.de

ABSTRACT

In this paper, we present the first systematic investigation of data remanence effects on an intrinsic Static Random Access Memory Physical Unclonable Function (SRAM PUF) implemented on a commercial off-the-shelf (COTS) device in a temperature range between -110° C and -40° C. Although previous studies investigated data remanence in SRAMs only at temperatures above -50° C, our experimental results clearly indicate that the extended temperature region we examine has dramatic effects on the security of intrinsic SRAM PUFs. We propose a number of different attacks and experimentally verify that data remanence effects can be exploited successfully to attack intrinsic SRAM PUFs on a COTS device, where the (micro)processor and the SRAM reside on the same die. Our experimental attack writes a bit-string to memory and freezes the device. Due to data remanence effects the attacker-known bit-string remains in memory and is subsequently read out by the bootloader to generate the PUF response. In this way, the attacker is able to construct a forged secret key by manipulating the PUF response. Finally, we also discuss and assess potential countermeasures against the attacks we examine.

Keywords

Data remanence, static random access memory (SRAM), physical unclonable function (PUF), low temperature, attack

1. INTRODUCTION AND MOTIVATION

The use of cryptography usually requires, among others, the identifaction of parties, the secure storage of a secret on a device, and a source of randomness. Physically Unclonable Functions (PUFs) have been shown to provide adequate security mechanisms for cost-efficient identification, key storage, and random number generators on commodity devices [26] and have been implemented in industrial applications already [23, 49]. The security of a PUF is based on the existence of at least one (random but stable) output that is unique per device for some given input. In this case, the input is referred to as a *challenge* and the corresponding output as a *response*, thus together forming a *challenge-response pair* (CRP). The uniqueness of such an output is strongly based on the existence of small, naturally occurring variations and inherent disorders between two identically manufactured devices, which result in different outputs when provided with the same input [39]. In this way, each PUF instance acts as a physical implementation of a one-way function, with its output being a unique inherent secret per device [35]. Based on the number of challenge-response pairs PUFs are classified into *weak* and *strong* PUFs.

In this paper we focus on a specific class of weak PUFs, namely SRAM PUFs, whose response is built by concatenating the start-up state of cells of a Static Random Access Memory (SRAM) device [18]. This raw PUF response is highly unique for an SRAM device, being both random and robust in the sense that only a few cells' start-up values are not stable over time [26]. Of particular interest are *intrinsic* SRAM PUFs, i.e., PUFs that do not require the addition of specialised security hardware but are inherently characterised by the hardware itself. Consequently, they have been proposed as a security anchor in low-cost commodity and legacy devices, providing identification and key generation solutions [12, 16, 17, 19, 20, 29, 30, 31, 33, 36, 37, 47, 50], or a source of randomness [16, 17, 19, 20, 36, 47, 48].

Given their wide range of application we investigate the security provided by intrinsic SRAM PUFs, which—in contrast to the class of strong PUFs—have not been analyzed thoroughly before. For this purpose we study the data remanence effects of SRAM cells at very low temperatures with the intention to attack their identification, key storage, and random number generation abilities. For our study we have used a commercial off-the-shelf (COTS) device with a modern design layout, where the SRAM is placed on the same die as the (micro)processor. Contrastingly, all previous studies examined either standalone SRAM modules [4, 5, 24, 43, 52], FPGA implementations [2, 46], or ASICs [3, 34, 54]. Moreover, our paper is—to the best of our knowledge—the first systematic investigation of low-temperature data remanence effects of SRAM cells in the temperature region between $-110°$ Celsius and $-40°$ Celsius while previous studies have investigated data remanence in SRAMs only at temperatures above $-50°$ Celsius [3, 5, 24, 34, 43, 46, 52, 54]. This extended temperature region, as our study clearly indicates, has dramatic effects on the security of intrinsic SRAM PUFs, which may be exploited by a number of simple attacks that can be easily implemented even without prior expertise.

These attacks utilise the fact that, for intrinsic SRAM PUFs, the memory area used for the construction of the PUF is typically shared with the operating system and the user space programs. Thus, the SRAM can be written to while the system operates. In particular, we show that an attacker, who has compromised a user-space application on the device, can write data to the memory and preserve it through data remanence effects. Essentially, the attack alters the start-up values of the SRAM PUF to a bit-string known to the attacker. We implemented two such attacks and, in addition, provide possible countermeasures to prevent low-temperature data remanence attacks.

Summarizing our contribution, we present in this paper:

(a) the first thorough (attack-focused) study on data remanence of SRAM cells in the temperature range from $-100°$ C to $-40°$ C,

(b) a discussion of attacks on *intrinsic* SRAM PUFs that exploit the observed data remanence effects,

(c) an experimentally verified implementation of such an attack, which successfully recovered the PUF response, and

(d) a proposal for countermeasures that go beyond the current state-of-the-art.

Outline

The remainder of this paper is structured as follows: Section 2 describes the setup and the results of the low-temperature data remanence study of SRAM cells. In Section 3 we explain in detail the proposed attack scenarios, discuss our experimental evaluation of such an attack, and present possible countermeasures. Section 4 discusses the related work on data remanence effects and previously proposed attacks against SRAM PUFs. Finally, Section 5 contains some final remarks on the attacks and their significance regarding the security of intrinsic SRAM PUFs, as well as a few directions of potential future research on related topics.

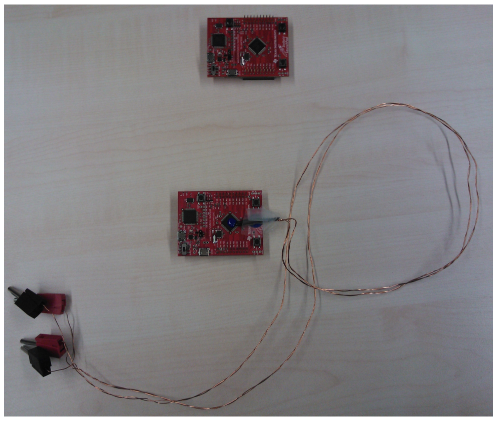

Figure 1: The Stellaris board, above, as it normally appears and, at the bottom, with the PT1000 resistance temperature detector sensor glued to its microprocessor module.

2. LOW-TEMPERATURE DATA REMANENCE STUDY

In this section, we explore the data remanence effects in an intrinsic SRAM PUF, implemented on a commercial off-the-shelf (COTS) device, in the temperature range between $-110°$C and $-40°$C. We have also conducted additional experiments at lower and higher temperatures to consolidate our results regarding the attacks proposed in Section 3.

We chose to investigate the data remenance effects of a Stellaris LM4F120 LaunchPad Evaluation Board (EK-LM4F120XL) produced by Texas Instruments. Stellaris incorporates the SRAM module on the actual die of its microprocessor and its SRAM cells have been shown to exhibit good PUF characteristics [27]. In this regard, our setup is novel as previous studies examined either standalone SRAM modules [4, 5, 24, 43, 52], FPGA implementations [2, 46], or ASICs [3, 34, 54], while our study uses a commercial off-the-shelf (COTS) device with a modern design layout, where the SRAM is placed on the same die as the (micro)processor.

2.1 Study setup

The Stellaris board utilises the ARM Cortex-M4F LM4F120H5QR microcontroller. On the die of the microcontroller resides also a 32 KB single-cycle on-chip SRAM [45], which we used as an inherent on-board SRAM PUF. Although the Stellaris board provides an internal temperature sensor, we also used an external sensor to valudate the temperature readings. The two sensors reported a temperature difference of 1-2°C, which appears plausible as one sensor is inside the module's package and the other was glued on top of it as shown in Fig. 1.

The external temperature sensor consisted of an Agilent 34401 multimeter connected to a PT1000 resistance temperature detector (RTD) in 4-wire setup to eliminate the influence of the contact (lead) resistance on the measurement. The PT1000 temperature sensor was glued on top of the microprocessor's package housing with the thermally conducting epoxy glue "WLK 30" produced by Fischer Elektronik.

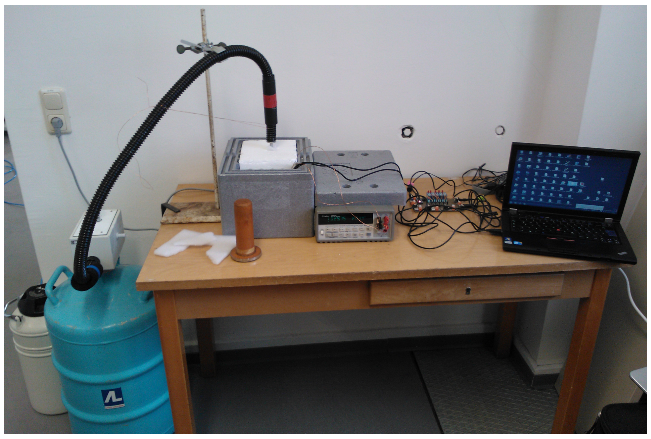

Figure 2: An overview of the experimental setup, showing the cryogenic storage dewar containers filled with liquid nitrogen on the left and the Styropor (polystyrene) box in the middle, with an air tube connecting the heat exchanger on top of the blue dewar container to it. Next to the Styropor isolation box, on its right, is the Agilent 34401 multimeter which is connected to the PT1000 RTD sensor with copper cables. Further to the right is the custom board which controls the device's power supply on-off time and which is connected to the Stellaris board through USB (Universal Serial Bus) cables, and on the rightmost position sits the computer with which we control the custom board and with which we cause the Stellaris SRAM PUF to generate CRPs.

In order to thermally isolate the PUF device, we placed it inside a Styropor (polystyrene) box with feed-throughs for the wires and cooling air. Additionally, in order to overcome the obstacle of very short-term data remanence due to the device being functional and fully grounded [43, 46], we built a custom board which can control the device's power supply on-off time in the region of milliseconds. The experimental setup is shown in Fig. 2.

We also noted that the board remains operational and responsive until the environmental temperature falls below $-120°$C, a temperature well below the usual industrial limit of good operation at around $-40°$C (see the microcontroller's data sheet [45]). Therefore, we chose to cool the board to temperatures around $-110°$C and above using pressurized air flowing through a heat exchanger placed inside a dewar of liquid nitrogen. Additionally, we used liquid nitrogen itself to cool the board below $-110°$C in order to investigate how such temperatures can affect its operation and whether they ensure total data remanence.

Furthermore, we must also note the role of *burn-in* effects on data remanence attacks, i.e., the tendency of values stored for a long time on an SRAM cell to *burn in* and persist on that cell [14, 41]. This means that the longer a value is allowed to burn in, the longer it is expected to persist on that cell due to data remanence. In our experiments, we tried to reduce such effects as much as possible to make sure that an attack can be successful even if the attacker does not have an extended amount of time in order to burn in a convenient pattern on the SRAM cells. In this way, we tried

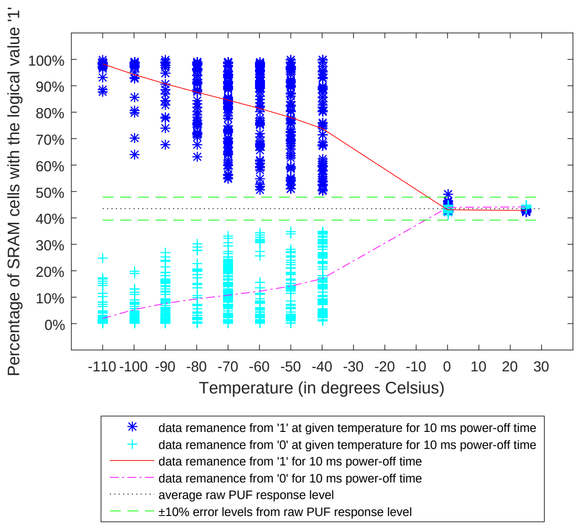

Figure 3: Results reflecting the level of data remanence after writing all the SRAM cells with either logical '1' or logical '0' at different temperatures for 10ms power-off time.

to avoid providing the attacker with an advantage that may not always be possible to gain in reality.

Finally, in order to read the memory values right after start up, we are using a modified bootloader, based on the Texas Instruments Real-Time Operating System (RTOS) for the Stellaris board and its inherent bootloader. Our bootloader reads the initial SRAM values, transmits them to our computer through the UART (Universal Asynchronous Receiver Transmitter) interface carried out through the USB connection and then overwrites all the SRAM cells with a given pattern. Additionally, we also implemented non-privileged code in order to access the internal temperature sensor of the board and be able to compare its readings to those of the external one.

2.2 Results

In order to measure the data remanence of the Stellaris SRAM PUF, we wrote a known pattern to its SRAM, powered off the device for some very short time interval (10 and 20 milliseconds), and subsequently measured how many of the bits corresponded to the previously written pattern. We conducted our experiments by writing both an all-zero and an all-ones pattern to the SRAM cells to determine whether the logical values being written can somehow influence the data remanence effects we observed.

We measured the data remanence of the Stellaris SRAM PUF between $-110°$ and $-40°$C for 10 and 20ms power-off times, as shown in Figs. 3 and 4. As shown in Fig. 3, for a power-off time of 10ms, our results indicate an extremely high level of data remanence at temperatures lower than $-100°$C, even though the board is functioning and thus the SRAM is grounded. Additionally, we note that the data remanence decreases as the temperature increases. At 0°C and above, there is essentially no data remanence, as the results indicate that the cells have returned to their usual initial values after the device was rebooted. We also observed, however, that (a) on average more than 75% of the SRAM cells' contents were preserved even for temperatures

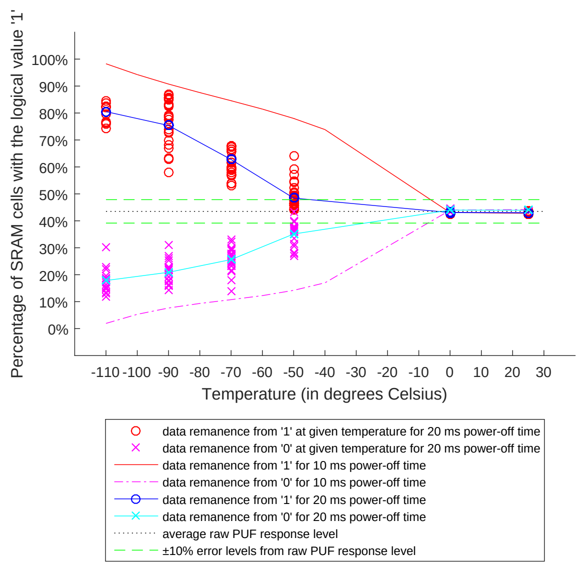

Figure 4: Results comparing the level of data remanence after writing all the SRAM cells with either logical '1' or logical '0' at different temperatures for 20ms power-off time to that observed for 10ms power-off time.

as low as $-40°$C and (b) the variance of the data remanence results increases as the temperature increases.

As shown in Fig. 4, the data remanence levels for the 20ms power-off time were (for all temperature levels) at least 20% lower than those for the 10ms, while the SRAM cells have almost completely returned to their initial values as the temperature approaches $-50°$C and higher. This degradation is more noticeable in Figs. 5 and 6, where the results for 10ms and 20ms power-off times at different temperatures are compared, first for the SRAM cells having been overwritten with '1' (Fig. 5) and then for the cells having been overwritten with '0' (Fig. 6).

We also note that, as shown in Figs. 3 and 4, at very low temperatures, there are no significant differences between the data remanence of logical value '1' and of logical value '0' in the SRAM cells, neither in the case of 10ms power-off time nor in the case of 20ms power-off time. Yet, in the course of our experiments, we noticed that the SRAM is slightly biased towards logical '0', with its normal start-up values being 57% zeros and only 43% ones. This is also indicated by the average raw PUF response level in the different figures concerning the results of our measurements. Therefore, the data remanence of the logical value '1' can be considered to be decreasing slightly faster as the temperature increases in comparison to the data remanence of the logical value '0', as in both cases the curve indicating the relation between data remanence and temperature tends to converge to the normal start-up values, the PUF response. Thus, one observes a quite higher data remanence rate of the logical value '0' at temperatures below $-80°$C compared to the data remanence rate of the logical value '1' at the same temperature levels.

Our measurements were performed using two different Stellaris boards in order to validate our experimental results. We must also note that given the very short power-off

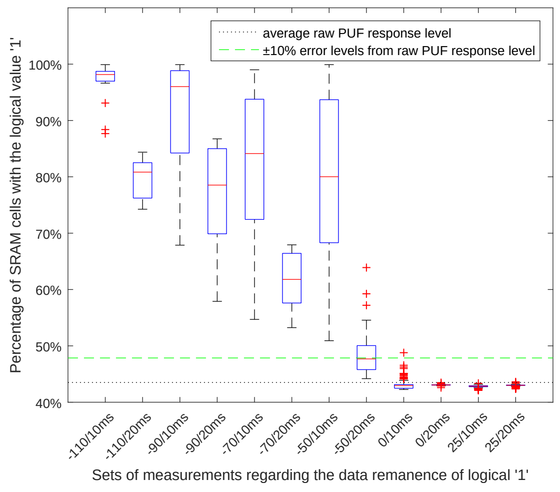

Figure 5: A comparison of measurements for 10ms and 20ms power-off times at different temperatures with data remanence of logical '1'. In the identifier of each set, the first number denotes the temperature at which the measurement took place, and the second the relevant power-off time in milliseconds.

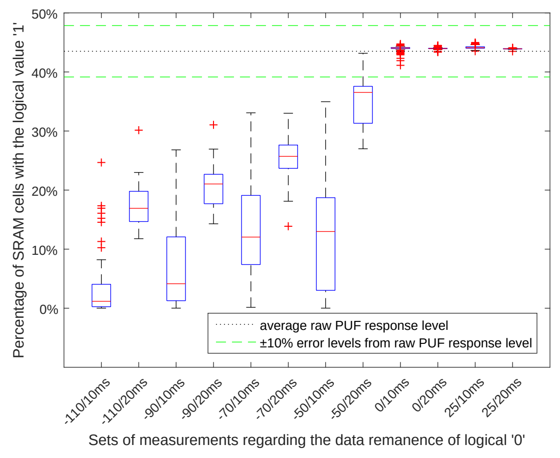

Figure 6: A comparison of measurements for 10ms and 20ms power-off times at different temperatures with data remanence of logical '0'. In the identifier of each set, the first number denotes the temperature at which the measurement took place, and the second the relevant power-off time in milliseconds.

reset time and the very low temperatures at which the Stellaris board had to operate, our code sometimes failed to fully overwrite or read the SRAM. We also note that we achieved better results when the board was pre-frozen close to the temperature at which we wanted to measure, so that the temperature could be subsequently lowered to the actual measurement temperature and remain stable for the whole duration of our measurements.

Additionally, we must stress that all our measurements were contacted at temperatures which may vary $\pm5°$C from the reported temperature, as the temperature was slowly rising. For this reason, we tried to start our measurements at a temperature that was $\sim5°$C lower than the noted measurement temperature and make sure that at no point during the measurements the temperature deviated more than 5°C from its reported value. Furthermore, as the SRAM was functional and thus grounded, we observed almost no data remanence when the power-off time exceeded 20ms at -50°C or 50ms at -110°C.

Finally, we have to mention that we also froze the board to temperatures around -120°C, noting even higher data remanence levels. However, at this temperature, the board becomes unstable and either halts temporarily for some time interval, continuing its execution off where it was left when it becomes operational again, or becomes completely unresponsive for some time, having lost power internally, and subsequently reboots, once its internal power is regained. Therefore, attacks which require a responsive SRAM PUF cannot be performed at this temperature and below. Nevertheless, we have noted a data remanence level ranging between 80% and 98% even after a very extended power-off time at temperatures below -120°C.

3. DATA REMANENCE ATTACKS ON INTRINSIC SRAM PUFS

In this section, we propose a number of low-temperature data remanence attacks against the use of an intrinsic SRAM PUF as an identification mechanism, a secure key storage device or, a random number generator. All attacks proposed make use of the data remanence effects noted above. Our basic scenario regarding the operation of the SRAM PUF is based on the following SRAM PUF use case:

1. When the device reboots, some code segment of the bootloader queries the SRAM for its start-up values, which form the raw SRAM PUF response.
2. Then, at some point, helper data for error correction are acquired from either the device itself or from another device, along with any other information needed.
3. The raw PUF response and the helper data are combined according the implemented protocol to, e.g., generate a secret key.
4. Subsequently, the same bootloader code segment overwrites the SRAM with a specific pattern (could be all logical '0' or '1' or some different pattern), which essentially erases the raw SRAM PUF response from the SRAM cells.

In the rest of this section we sketch some attacks against this setup. All attacks utilise data remanence effects of SRAM cells and can be performed by attackers with little expertise and budget. Additionally, we distinguish between attacks that could only take place before the SRAM has been overwritten (step 4 above), to which we refer as pre-erasure data remanence attacks, and ones that can still be performed after this process takes place, to which we refer as post-erasure data remanence attacks.

3.1 Pre-erasure data remanence attacks

The attacks described in this section target the boot phase in an attempt to prevent the memory from being overwritten by the bootloader. Subsequently, low-temperature data remanence is utilised to preserve and read out the raw PUF response stored in the SRAM cells.

Assuming that the relevant helper data are public, as is often the case, or can somehow be acquired, the attacker can combine them with the values stored in the SRAM to construct the same key as the one now produced by the device.

We can distinguish two different cases of attacks, depending on the actual type and level of access the attacker has to the SRAM PUF device.

In the first case, the attacker has access only to non-privileged code which can read the SRAM contents, while in a second case, the attacker has physical access to the SRAM PUF device and can also query it and receive meaningful responses. As the first case essentially tries to gain access, after a reboot, to the data of a previous session (the PUF response) preserved through low-temperature data remanence, it can be seen as a cold boot attack.

In the case where an attacker cannot immediately modify the bootloader of the SRAM PUF device into revealing its raw PUF responses, an attacker would need to find a way to jump, before the memory is overwritten, to some non-privileged attacker code which would allow the PUF response to be captured. However, this solution may require access to at least the binary source code of the bootloader and having it reverse engineered, unless the attacker finds a way to skip the memory overwriting instructions and continue execution of the bootloader until unprivileged user space code can be executed.

We suggest that this can be achieved by freezing the device before the memory overwriting instructions are executed, in such a way that execution continues and the device remains responsive but the overwriting instructions have essentially no effect. However, we were unable to experimentally verify this. Such an attack can be feasible either if the SRAM module is not bundled in the same packaging as the processor, and thus can be independently frozen while the processor continues execution normally, or if well-targeted *cryomicroprobing* is feasible, i.e., the targeted freezing of specific elements of the device, such as registers and/or other logical circuitry.

The second case is based on freezing the device before its SRAM is overwritten and probing into it to extract the preserved raw PUF response. This approach was also mentioned by Samyde et al [40], albeit without referring to a specific way of probing into the SRAM. This can potentially be achieved either by *cryomicroprobing* or by the techniques proposed by Nedospasov et al. [32] regarding invasive PUF analysis that employs electrical stimulation by heat or laser. We believe that, if practiced correctly, either technique could have minimal effects on the state of nearby frozen SRAM cells, while successfuly unfreezing specific cells and reading them out, until the whole SRAM PUF response has been acquired.

In order for any of the aforementioned attacks to be successful, the attacker needs to time the bootloader and, through trial and error, estimate when the bootloader overwrites the SRAM.

3.2 Post-erasure data remanence attacks

A different category of attacks can take place even if the SRAM has already been overwritten. In contrast to the attacks outlined in Section 3.1 that target the read out of the actual PUF response, these attacks rather aim to "program" the PUF response. Therefore, these attacks are less complicated to carry out.

As a first scenario, we consider an attacker who can store a chosen set of values in the SRAM, e.g., by exploiting a software vulnerability that allows the injection of code which overwrites it with a specific pattern during runtime, and who then can freeze the device. Due to data remanence effects, when the system recovers from being frozen and reboots, the initial values of the SRAM cells will not be a random raw PUF response, but rather a somewhat noisy version of the chosen set of values that the attacker had stored on the SRAM. As in the previous category of attacks, we again assume that the attacker has access to the helper data and can combine them with a captured raw PUF response in order to reproduce legitimate keys. Alternatively, an attacker could also replace the helper data stored in the device with data that fit the forged response in order to facilitate the generation of a legitimate key.

In another similar scenario, the attacker can induce errors on the SRAM in a similar fashion as described by Oren et al. [34] and Zeitouni et al. [54], using temperature in conjunction with the power-off (reset) time to produce a range of responses partially matching the legitimate one. Then, by applying differential fault analysis, the attacker can recreate the legitimate PUF response. The main advantage of our approach is that by controlling the temperature one can quite precisely control the amount of faults induced in the SRAM cells and thus easily get responses on which the differential fault analysis attack can be applied successfuly.

In these two ways, the attacker can successfuly either alter the SRAM PUF response in order to produce forged keys, which can subsequently authenticate some malware as legitimate code, or recreate the legitimate response of the PUF in order to authenticate to third parties, provided that the related helper data are public or at least accessible to the attacker. Finally, if one takes into account that such helper data may correct errors accounting for up to 5-10% of the overall SRAM PUF response size, which is the natural variation in raw PUF responses [26, 19, 12], we expect that these attacks are already successful if data remanence effects allow the attacker to keep at least 90-95% of the written memory content. We also note that these attacks do not require any changes to the bootloader.

3.3 Experimental results and discussion of the success potential of the proposed attacks

In order to test whether the data remanence effects described in Section 2 enable the attacks mentioned in Section 3, we ran several experiments using the experimental setup described in Section 2.1 and additional code we implemented. In particular, we produced non-privileged attacker code which reads the SRAM values and then copies them to a different memory segment or transmits them to our computer through the UART interface. In this way, we want to test whether there is a way to bypass the memory overwriting instructions of the bootloader and jump into this user space attacker code by freezing the board, and thus implement one of the attacks discussed in Section 3.1. In our case, the execution of this code is triggered when a specific button of the board is pushed.

Finally, we have also implemented non-privileged attacker code which can be used to write on the SRAM, in order to implement data remanence attacks requiring new data being written before the device is frozen, as proposed in Section 3.2. Moreover, as we set the pattern with which the bootloader overwrites the SRAM on our own, we were able to easily recognise whether the SRAM had been overwritten or not, in order to properly time our experiments.

3.3.1 Examining the potential for pre-erasure attacks

Concerning the attacks happening before the SRAM is overwritten, described in Section 3.1, we were unfortunately not able to freeze the board into disregarding the overwriting of the SRAM instructions run by the bootloader and then continuing normal execution of the operating system and our non-privileged attacker code. We therefore leave it on future research to investigate ways to make this attack practical. The only missing piece seems to be finding a way to freeze only the SRAM, without disrupting the microprocessor and its execution sequence.

Nevertheless, an attacker can freeze the board below $-120°$C, causing it to lose power and stop operating before the SRAM has been overwritten, and then probe into it with one of the techniques we have already mentioned in Section 3.1 while it is kept frozen and out of operation, as we have noted a data remanence level ranging between 80% and 98% even after a very extended power-off time at temperatures below $-120°$C. As the data remanence levels observed are high, we expect such an attack to be successful. Nevertheless, the actual implementation of such an invasive attack remains the subject of future research. In this case, helper data can again be used to correct a significant amount of errors and lead to a successful attack, if the amount of errors is kept low by reducing the power-off time.

3.3.2 Testing out a post-erasure attack

Regarding the data remanence attacks described in Section 3.2, we tested the rate of their success by conducting the following experiment. We first constructed helper data which would match a selected secret key that the attacker wants to produce and the response that the attacker wants to enforce the SRAM PUF to produce, by having overwritten the SRAM with it (either all '0' or all '1' in our experiments).To this end, we combined the PUF response an attacker would like to obtain (either all '0' or all '1' in our experiments) with a secret key, which is a pre-selected bitstring of a particular size, through a fuzzy extractor scheme [7, 8, 9, 15] in order to construct the relevant helper data.

Then, we combined these helper data with the raw responses produced by the SRAM PUF after the device has been reset at a particular temperature through the selected fuzzy extractor scheme, in order to test if the selected secret key could be successfully reconstructed by each recovered response at a given temperature. By using these key enrolment and key reconstruction phases, we were able to determine the rate of successful key reconstruction from the recovered raw PUF responses after a particular reset time at a given temperature.

The error correction of our fuzzy extractor scheme is based on a simple repetition code and the Golay $(23, 12, 7)_2$ error correction code (perfect binary Golay code), a proven error correction scheme for SRAM PUFs [15, 42], which can correct errors accounting for up to $\sim$10% of a recovered raw response of our SRAM PUF, if they are uniformly distributed. Unfortunately, this proved not to be the case with our recovered responses, and although the error correction code

Table 1: Rates of successfully reconstructing the selected key from recovered raw SRAM PUF responses after a 10ms reset time and after a 20ms reset time.

pattern	10ms power-off time									
written	$-110°$C	$-100°$C	$-90°$C	$-80°$C	$-70°$C	$-60°$C	$-50°$C	$-40°$C	0°C	25°C
all '1'	90%	∼88%	60%	∼32%	<20%	<20%	∼11%	∼8%	≡0%	≡0%
all '0'	∼93%	∼92%	∼64%	∼63%	∼51%	<50%	∼38%	∼28%	≡0%	≡0%

pattern	20ms power-off time			
written	$-110°$C	$-90°$C	$-70°$C	$-50°$C
all '1'	<10%	<10%	≡0%	≡0%
all '0'	<10%	<10%	≡0%	≡0%

used always corrects at every temperature errors accounting for up to at least 5%, the exact correction threshold per raw response depends both on the amount of errors contained in it and on their actual distribution. Nevertheless, we have successfully tested our scheme using both 128-bit and 512-bit keys. Our results for reset times of 10ms and 20ms for the temperature region between $-110°$C and $-40°$C are presented in Table 1.

Based on the results for a reset time of 10ms, we can note that an attacker can enforce the generation of an own key based on the data remanence of the values having been stored on the SRAM, in 90% of all cases for temperatures below $-100°$C and in more than 60% for temperatures below $-90°$C. Additionally, we note a discrepancy in success rates of logical '0' and '1', which can be attributed to the initial SRAM cell values being slightly biased towards logical '0'. We therefore note that an attacker is more successful when storing the logical value towards which the SRAM PUF is more biased. We must note that in case a pattern of all '0' has been stored, successful key reconstruction rates are close to 50% even for temperatures close to $-50°$C and below. We therefore can note the importance of data remanence in SRAM PUFs at temperatures below $-50°$C.

Based on the same results, we can also conclude that an attack based on differential fault analysis in a similar fashion as described by Oren et al. [34] and Zeitouni et al. [54], using temperature in conjunction with the power-off (reset) time to produce a range of responses partially matching the legitimate one, seems plausible, especially for temperatures below $-100°$C, where the amount of faults induced appears to be quite small and controllable. By controlling the temperature more precisely, one can quite precisely also control the amount of faults induced in the SRAM cells and thus easily get responses on which the differential fault analysis attack can be applied successfuly. Nevertheless, the actual realisation of such an attack is the subject of future research.

Finally, we also note that for a reset time of 20ms, the successful key reconstruction rate falls below 10% even for a temperature around $-110°$C and is definitively equivalent to 0% for temperatures around $-70°$C and above.

3.4 A discussion of potential countermeasures

A variety of countermeasures against data remanence have been proposed in the relevant literature. However, solutions which alter the memory design architecture by adding extra circuits and components [14, 25, 51], cannot be considered as efficient, or even feasible, as they also cause extra manufacturing costs which can usually be high enough to not allow such solutions to enter mass production. Furthermore, solutions which are based on a different kind of memory, other than an SRAM, such as the ones proposed by Zhang et al. [55], not only suffer from high manufacturing costs, but also obviously exclude the existence of an SRAM PUF, which is the target of the data remanence attacks which we examine.

Furthermore, some more conventional countermeasures include the addition of wire meshes and other physical defences, the obfuscation or encryption of the memory, the restriction of access to privileged components, such as the bootloader, and the overwriting or erasure of the memory [6, 11, 14, 28]. However, these countermeasures can only make data remanence attacks harder to perform, but not completely prevent them, as there are ways to bypass them. For example, we have proven that some attacks can be successful even if the SRAM is overwritten. Moreover, such countermeasures would also require additional resources and costs.

Huffmire et al. [22] have even proposed physically destructing the device as a means of preventing data remanence attacks. Destructing the device or significantly altering it cannot serve as an efficient way of protection, especially in the case of SRAM PUFs, where the SRAM has to remain fully functional in order to serve as a PUF, and degaussing it would probably significantly alter its PUF characteristics and, thus, its response. Therefore, such countermeasures can only serve as a last resort in order to prevent other parties gaining access to the device and its secret at all costs.

Defining a specific minimum power-off time for the device [5, 10], can also be a potential countermeasure. However, in order to implement such a countermeasure, the device would have to never lose power completely, as the system would need some way of constantly determining the time and its on-off state. In such a case, an attacker could bypass such an extra system by also powering it off or disrupting its operation and/or registers, or even, cool down the device to an even lower temperature where the duration of the data remanence effect exceeds the set minimum power-off time.

Another interesting countermeasure against data remanence attacks is ensuring that the SRAM used as a PUF cannot be accessed by non-privileged software. For example, if instead of an intrinsic PUF, a dedicated SRAM is used as a PUF, an attacker may not use a software vulnerability in order to overwrite or, even read, its contents. Nevertheless, even in this case, efficient attacks based on low-temperature data remanence may still exist, such as freezing the SRAM PUF and probing it in order to extract its response. Such attacks, however, are left to future research.

Additionally, the use of temperature sensors to detect abrupt changes in the temperature and take adequate action in order to protect the device can potentially work [11], if the device remains operational. Nevertheless, an attacker could bypass such sensors by powering off the system or disrupting them, especially if they were not mounted on the SRAM itself. In either case, however, the effectiveness of this countermeasure in the case of low-temperature data remanence attacks against SRAM PUFs is a potential subject of future research.

Its effectiveness is highly dependent on what happens when a deep surge in the temperature is detected; in case only the memory is overwritten, our attacks would still succeed. In case the system is powered off, some of our attacks can still be successful, if we can power it back on quickly enough. Finally, in case the SRAM is somehow disabled permanently, that would destroy the SRAM PUF device, thus effectively achieving at least an availability attack. We must also note that not all devices have an internal temperature sensor and it addition will also inevitably increase the manufacturing costs of the device.

4. RELATED WORK

Unlike Dynamic Random Access Memories (DRAMs), which tend to be external modules on a system, SRAMs are usually incorporated in the same module as the system's (micro)processor. Thus, DRAMs tend to be more vulnerable to physical and side-channel attacks, while SRAM modules are far less easily accessible, providing much fewer opportunities for successful exploitation.

As far as cold data remanence attacks are considered, a well-known attack targeting DRAM modules is based on successfuly freezing the module and removing it from the targeted system, in order to gain access to its contents [14, 11]. Such an attack, however, may not be feasible against an SRAM, which physically resides in the same module as the system's (micro)processor. Thus, removing such a module and gaining access to it are a much harder tasks to achieve. Especially, since the SRAM is being used as a PUF, it cannot be replaced without notice by a structurally identical module.

4.1 Data remanence effects in SRAMs

A number of different publications consider the effects of data remanence in SRAM cells. Most notably, Skorobogatov [43] discusses the effects of low temperature on data retention on a number of different SRAM models produced between 1987 and 2000. However, unlike our study, Skorobogatov only examines SRAM in its conventional role as a memory module, and not as a PUF, and only provides data for measurements at temperatures above -50° Celsius.

Jiao et al. [24] studied the effects of data remanence on SRAM modules produced between 1989 and 2004, showing that data remanence significantly increases as the temperature decreases. Their study proves that while the electric current in the memory cells decreases as the temperature decreases, the data retention rate of the different boards is independent of this fact. Furthermore, the authors also show that even on newer SRAM modules data remanence exists for a significant amount of time as the temperature decreases. Still, one must keep in mind that also this article refers to standalone SRAM modules and not to those incorporated in the packaging of the processor. However, the authors did not consider SRAMs as PUFs and only examined data retention for temperatures above -30° Celsius.

As already mentioned, SRAMs are no longer a separate module, but rather incorporated in the same package as the processor, and thus their removal or replacement on the system is not a feasible attack strategy. A potential solution to this problem is proposed by Samyde et al. [40], who attempted a *cryomicroprobing* attack that freezes and subsequently probes individual cells of the SRAM.

Inspired by Skorobogatov, Tuan et al. [46] examined the data remanence effects observed on different SRAM cells contained on an FPGA for temperatures above -40° C, illustrating once again a significant degree of data retention in low temperatures. They also observed that modern SRAM cells have a bias towards a specific value, an effect that forms the basis for their ability to serve as a PUF, with a slightly higher tendency towards logical '0' than logical '1'.

Chen et al. [4] examined the effects of circuit degradation on data remanence in an FPGA, while also identifying a potential way to tackle this issue. Their article exposes the potential effect of circuit degradation caused by aging on the data remanence effects exhibited by SRAM cells. Additionally, Saxena and Voris [41] explored the role of data remanence in SRAM modules regarding the use of SRAMs as random number generators on RFID tags.

Finally, Cakir et al. [3] compared the data remanence of an SRAM to that of a DRAM for temperatures between -40° C and 85° C. They showed that the SRAM exhibits a much higher degree of data remanence, while the data remanence of both devices seems to be growing logarithmically as the temperature decreases and quite linearly as the power-off time increases.

4.2 Data remanence attacks on SRAM PUFs

Oren et al. [34] have proposed an attack against an ASIC implementation of SRAM PUFs based on data remanence decay. The attack utilises both data remanence and decay in order to induce faults in a state where all the SRAM cells have already been overwritten with a specific known value, and thus the nominal start-up values are not known. The device is powered off repeatedly, allowing each time a larger amount of different cells to decay and thus revert to their nominal start-up values when the system is again powered on. Therefore, the faults induced each time correspond to legitimate bits of the SRAM PUF response. Finally, a differential fault analysis technique, first proposed by Biham and Shamir [1], is applied on the collected SRAM responses, in order to recover more and more legitimate bits, until the full PUF response has been correctly recreated. Such an approach would, of course, require the attacker to be able to query the SRAM PUF multiple times, each time getting a meaningful response based on the faulty values of the SRAM cells. Additionally, all tests were performed at room temperature ($\sim$25° C) and the authors acknowledge that powering off the SRAM long enough would move the SRAM in a state where all of its cells have turned to their nominal start-up values, when the device is again powered on, thus essentially preventing the differential fault analysis of the responses.

As it is therefore essential to be able to control the exact data retention rate of the SRAM, Zeitouni et al. [54] built on the initial publication by Oren et al. by suggesting a voltage control system, which provides much better results than the time control system proposed in the initial form of the publication regarding the control of the data retention rate of the SRAM. As [54] is essentially an extension of the article by Oren et al., one must note that while the tests using the time control system are taken from the initial publication, having been performed at room temperature (∼25° C), the tests using the new voltage control system were performed inside a refrigerator at temperatures between 2.7° C and 7.6° C. Therefore, and as the authors clearly acknowledge the effects of low temperatures to data remanence, one could question whether the two suggested control approaches can be truly compared regarding their effectiveness. Note that in Section 2 of this paper, we provide insights into this question by proving the efficiency of time-based control of data remanence in SRAM PUFs in conjunction to really low temperatures.

Furthermore, Wild and Güneysu [52] examined the use of SRAM-based block memories (BRAMs) in Xilinx FPGAs as a basis for SRAM PUFs. In their article, the authors also examine the effect of data remanence on their suggested PUF, albeit for relative high temperatures, around 50° C. As they note, there seems to be little to no data remanence at such temperatures. One could note that data remanence, the tendency of an SRAM cell to keep its logical value, and bit flipping, the tendency of an SRAM cell to change its value to the opposite logical one, are inversely related. Therefore, while low temperatures enhance data remanence, high temperatures tend to support bit flips.

While potential countermeasures against data remanence attacks are discussed in detail in Section 3.4, we must also note here a few of the proposed ideas of preventing data retention attacks against SRAM PUFs. For example, Zhang et al. [55] suggested that using a different kind of memory, namely Spin-Transfer Torque Magnetoresistive Random Access Memory (STT MRAM), could successfuly prevent data remanence attacks against memory-based PUFs, while Wenjing et al. [51] suggested securing the SRAM's power supply or adding circuitry to overwrite the SRAM in order to prevent such attacks. Finally, Kai et al. [25] also proposed quite exquisite changes in the design of SRAM cells as a countermeasure against data remanence attacks. Nevertheless, one has to also note that such solutions would come at an increased manufacturing cost and thus may not be always feasible and cost-efficient.

Additionally, Claes et al. [5] compare the data remanence of SRAM and FF (Flip Flop) PUFs at temperatures between -40° C and 20° C, in order to determine a suggested reset time for the two different types of PUFs at the two different temperatures such that data remanence effects affecting the PUF operation are avoided. However, as the device is powered off in order for the reset to take place, it is hard to prevent attackers from reducing the power-off time, i.e. the reset, to their own liking. Additionally, an attacker could subject the PUFs to much lower temperatures in order to ensure an extended data remanence time and overcome any relevant delay penalty imposed on the reset time. Therefore, the increase of the reset time may not constitute a truly effective countermeasure in order to prevent data remanence attacks.

Moreover, Holcomb et al. [21] proposed the use of relative data remanence effects of different SRAM cells for the construction of a PUF, while Xu et al. [53] improved the technique suggested in the original article. However, this idea would require data remanence effects to last an adequate amount of time in order for them to be measurable by the relevant circuitry, which may not always be the case. Interestingly, the original paper by Holcomb et al. also states that the cells which exhibit a strong data retention effect tend to be the ones with highly reliable power-up values, i.e. the more stable bits of the SRAM PUF response. Additionally, Tehranipoor et al. [44] suggested the use of the data remanence in a DRAM, this time for the implementation of a DRAM PUF.

Furthermore, Rahmati et al. [38] even proposed that data remanence could be used for timekeeping purposes and both articles regarding data retention decay attacks by Oren et al. [34] and Zeitouni et al. [54] suggest that the original system proposed by Rahmati could be further simplified, with its complexity being reduced from linear to logarithmic time.

Finally, Gutmann [13] provides an overview of the data remanence effects in a wide range of different semiconductor devices, including SRAMs, while also discussing the actual physical phenomena behind data remanence.

5. CONCLUSION AND FUTURE WORK

We have examined and discussed the results of data remanence effects on an intrinsic SRAM PUF implementation on a commercial off-the-shelf (COTS) device for temperatures ranging between -110° C and -40° C, while also proposing a number of easily implementable attack scenarios based on these effects. We have subsequently shown that low-temperature data remanence attacks appear to be feasible and successful against intrinsic SRAM PUFs, which can, for example, serve as secure key storage. Our results clearly indicate that the success of the described attacks is highly dependent upon the power-off time and the environmental temperature settings that an attacker can achieve. We have additionally noted that, because the SRAM is operational and serves as a PUF, and thus is also grounded, it exhibits only very short-term data remanence.

We have also discussed and compared a variety of different countermeasures and noted the potential existence of efficient countermeasures against the described attacks, suggesting that further research is required in this field. For example, we need to investigate whether the use of temperature sensors could actually constitute an effective countermeasure. We can therefore conclude that additional further research is required in order to better assess the exact degree to which low-temperature data remanence attacks may affect the role of SRAM PUFs as an adequate security mechanism.

Finally, it is also noteworthy that most of our attacks consider intrinsic SRAM PUFs only, which re-use the memory being used by the system and its applications as a PUF. It is worth also mentioning that a dedicated SRAM PUF, where the memory is only being used as a PUF and not as a regular memory component, would be much less affected by the attacks we propose, but nevertheless could still be frozen and probed by an attacker in order to extract a legitimate PUF response out of it. Such attacks against dedicated SRAM PUF modules should also be the subject of future research.

Acknowledgments

This work has been co-funded by the DFG as part of project P3 within the CRC 1119 CROSSING.

Additionally, the authors would like to thank Prof. Dr. Michael Vogel and the "Molecular Dynamics and Condensed Matter" group of the Institute for Condensed Matter Physics of TU Darmstadt for their significant help and collaboration that made our experiments possible.

We would also like to thank mrs. Huifang Jiao for providing us with her very interesting 2006 article [24], which although is in Chinese, can now be easily translated by modern technology.

6. REFERENCES

[1] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97*, pages 513–525. Springer, 1997.

[2] A. Braeken, S. Kubera, F. Trouillez, A. Touhafi, N. Mentens, and J. Vliegen. Secure FPGA technologies and techniques. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 560–563. IEEE, 2009.

[3] C. Cakir, M. Bhargava, and K. Mai. 6T SRAM and 3T DRAM data retention and remanence characterization in 65nm bulk CMOS. In *Custom Integrated Circuits Conference (CICC), 2012 IEEE*, pages 1–4. IEEE, 2012.

[4] H.-W. Chen, S. Srinivasan, Y. Xie, and V. Narayanan. Impact of Circuit Degradation on FPGA Design Security. In *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, pages 230–235. IEEE, 2011.

[5] M. Claes, V. van der Leest, and A. Braeken. Comparison of SRAM and FF PUF in 65nm technology. In *Information Security Technology for Applications*, pages 47–64. Springer, 2011.

[6] P. Colp, J. Zhang, J. Gleeson, S. Suneja, E. de Lara, H. Raj, S. Saroiu, and A. Wolman. Protecting data on smartphones and tablets from memory attacks. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 177–189. ACM, 2015.

[7] M. Cortez, G. Roelofs, S. Hamdioui, and G. Di Natale. Testing PUF-based secure key storage circuits. In *Proceedings of the conference on Design, Automation & Test in Europe*, page 194. European Design and Automation Association, 2014.

[8] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

[9] 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 2004*, pages 523–540. Springer, 2004.

[10] S. Eiroa, J. Castro, M. C. Martinez-Rodriguez, E. Tena, P. Brox, and I. Baturone. Reducing bit flipping problems in SRAM physical unclonable functions for chip identification. In *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, pages 392–395. IEEE, 2012.

[11] M. Gruhn and T. Müller. On the practicability of cold boot attacks. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 390–397. IEEE, 2013.

[12] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. *FPGA intrinsic PUFs and their use for IP protection*. Springer, 2007.

[13] P. Gutmann. Data remanence in semiconductor devices. In *Proceedings of the 10th conference on USENIX Security Symposium-Volume 10*, page 4. USENIX Association, 2001.

[14] 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*, 52(5):91–98, 2009.

[15] H. Handschuh. Hardware-Anchored Security Based on SRAM PUFs, Part 1. *IEEE Security & Privacy*, (3):80–83, 2012.

[16] H. Handschuh. Hardware-Anchored Security Based on SRAM PUFs, Part 2. *IEEE Security & Privacy*, (4):80–81, 2012.

[17] H. Handschuh, G.-J. Schrijen, and P. Tuyls. Hardware intrinsic security from physically unclonable functions. In *Towards Hardware-Intrinsic Security*, pages 39–53. Springer, 2010.

[18] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[19] D. E. Holcomb, W. P. Burleson, and K. Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *Computers, IEEE Transactions on*, 58(9):1198–1210, 2009.

[20] D. E. Holcomb, W. P. Burleson, K. Fu, et al. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security*, volume 7, 2007.

[21] 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*, pages 165–179. Springer, 2012.

[22] T. Huffmire, R. Kastner, et al. Threats and Challenges in reconfigurable hardware security. International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'08), 2008.

[23] Intrinsic ID. Physically unclonable functions (PUF). https://www.intrinsic-id.com/physical-unclonable-functions/physical-unclonable-functions/. Accessed: 2016-02-29.

[24] H. Jiao, X. Zhang, X. Jia, et al. The characteristic study of data remanence of SRAM (in Chinese). *Research & Progress of SSE*, 26(4):536, 2006.

[25] Y. Kai, Z. Xuecheng, Y. Guoyi, and W. Weixu. Security strategy of powered-off SRAM for resisting physical attack to data remanence. *Journal of Semiconductors*, 30(9):095010, 2009.

[26] S. Katzenbeisser, Ü. 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 2012*, pages 283–301. Springer, 2012.

[27] F. Kohnhäuser, A. Schaller, and S. Katzenbeisser. PUF-Based Software Protection for Low-End Embedded Devices. In *Trust and Trustworthy*

Computing, pages 3–21. Springer, 2015.

[28] O. Kömmerling and M. G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. *Smartcard*, 99:9–20, 1999.

[29] P. A. Layman, S. Chaudhry, J. G. Norman, and J. R. Thomson. Electronic fingerprinting of semiconductor integrated circuits, May 2004. US Patent 6,738,294.

[30] R. Maes and I. Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer, 2010.

[31] J. G. Merchan, S. S. Kumar, P. T. Tuyls, and G. J. Schrijen. Identification of devices using physically unclonable functions, Aug. 2008. US Patent App. 12/674,367.

[32] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit. Invasive PUF analysis. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on*, pages 30–38. IEEE, 2013.

[33] S. Okumura, S. Yoshimoto, H. Kawaguchi, and M. Yoshimoto. A 128-bit chip identification generating scheme exploiting SRAM bitcells with failure rate of 4.45× 10- 19. In *ESSCIRC (ESSCIRC), 2011 Proceedings of the*, pages 527–530. IEEE, 2011.

[34] Y. Oren, A.-R. Sadeghi, and C. Wachsmann. On the effectiveness of the remanence decay side-channel to clone memory-based PUFs. In *Cryptographic Hardware and Embedded Systems-CHES 2013*, pages 107–125. Springer, 2013.

[35] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.

[36] J. Petit, C. Bosch, M. Feiri, and F. Kargl. On the potential of PUF for pseudonym generation in vehicular networks. In *Vehicular Networking Conference (VNC), 2012 IEEE*, pages 94–100. IEEE, 2012.

[37] M. Platonov, J. Hlavác, and R. Lórencz. Using Power-Up SRAM State of Atmel ATmega1284P Microcontrollers as Physical Unclonable Function for Key Generation and Chip Identification. *Information Security Journal: A Global Perspective*, 22(5-6):244–250, 2013.

[38] 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 *Proceedings of the 21st USENIX conference on Security symposium*, pages 36–36. USENIX Association, 2012.

[39] U. Rührmair and D. E. Holcomb. PUFs at a glance. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6. IEEE, 2014.

[40] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater. On a new way to read data from memory. In *Security in Storage Workshop, 2002. Proceedings. First International IEEE*, pages 65–69. IEEE, 2002.

[41] N. Saxena and J. Voris. Data remanence effects on memory-based entropy collection for RFID systems. *International Journal of Information Security*, 10(4):213–222, 2011.

[42] A. Schaller, T. Arul, V. van der Leest, and S. Katzenbeisser. Lightweight anti-counterfeiting solution for low-end commodity hardware using inherent PUFs. In *Trust and Trustworthy Computing*, pages 83–100. Springer, 2014.

[43] S. Skorobogatov. Low temperature data remanence in static RAM. *University of Cambridge Computer Laborary Technical Report*, 536:11, 2002.

[44] F. Tehranipoor, N. Karimina, K. Xiao, and J. Chandy. DRAM-based intrinsic physical unclonable functions for system level security. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 15–20. ACM, 2015.

[45] Texas Instruments. *Stellaris®LM4F120H5QR Microcontroller data sheet*, Feb. 2013.

[46] T. Tuan, T. Strader, and S. Trimberger. Analysis of data remanence in a 90nm FPGA. In *Custom Integrated Circuits Conference, 2007. CICC'07. IEEE*, pages 93–96. IEEE, 2007.

[47] V. van der Leest and P. Tuyls. Anti-counterfeiting with hardware intrinsic security. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, pages 1137–1142. IEEE, 2013.

[48] V. van der Leest, E. van der Sluis, G.-J. Schrijen, P. Tuyls, and H. Handschuh. Efficient Implementation of True Random Number Generator Based on SRAM PUFs. In *Cryptography and Security: From Theory to Applications*, pages 300–318. Springer, 2012.

[49] Verayo Inc. Technology. http://verayo.com/tech.php. Accessed: 2016-02-29.

[50] I. Verbauwhede and R. Maes. Physically unclonable functions: manufacturing variability as an unclonable device identifier. In *Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI*, pages 455–460. ACM, 2011.

[51] K. Wenjing, Y. Kai, Y. Guoyi, and Z. Xuecheng. Novel security strategies for SRAM in powered-off state to resist physical attack. In *Integrated Circuits, ISIC'09. Proceedings of the 2009 12th International Symposium on*, pages 298–301. IEEE, 2009.

[52] A. Wild and T. Güneysu. Enabling SRAM-PUFs on Xilinx FPGAs. In *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, pages 1–4. IEEE, 2014.

[53] X. Xu, A. Rahmati, D. E. Holcomb, K. Fu, and W. Burleson. Reliable Physical Unclonable Functions Using Data Retention Voltage of SRAM Cells. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 34(6):903–914, 2015.

[54] S. Zeitouni, Y. Oren, C. Wachsmann, P. Koeberl, and A.-R. Sadeghi. Remanence Decay Side-Channel: The PUF Case. *IEEE Transactions on Information Forensics and Security*, 11(6):1106–1116, 2016.

[55] L. Zhang, X. Fong, C.-H. Chang, Z. H. Kong, and K. Roy. Optimizating Emerging Nonvolatile Memories for Dual-Mode Applications: Data Storage and Key Generator. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 34(7):1176–1187, 2015.