Physical Vulnerabilities of Physically Unclonable Functions

Clemens Helfmeier, Christian Boit
Semiconductor Devices,
Dept. of High-Frequency and Semiconductor System Tech.
Technische Universität Berlin,
Berlin, Germany
{clemens.helfmeier, christian.boit}@tu-berlin.de

Dmitry Nedospasov, Shahin Tajik, Jean-Pierre Seifert
Security in Telecommunications,
Dept. of Software Eng. and Theoretical Computer Science
Technische Universität Berlin,
Berlin, Germany
{dmitry, shahin, jpseifert}@sec.t-labs.tu-berlin.de

***Abstract*—In recent years one of the most popular areas of research in hardware security has been Physically Unclonable Functions (PUF). PUFs provide primitives for implementing tamper detection, encryption and device fingerprinting. One particularly common application is replacing Non-volatile Memory (NVM) as key storage in embedded devices like smart cards and secure microcontrollers. Though a wide array of PUF have been demonstrated in the academic literature, vendors have only begun to roll out PUFs in their end-user products. Moreover, the improvement to overall system security provided by PUFs is still the subject of much debate. This work reviews the state of the art of PUFs in general, and as a replacement for key storage in particular. We review also techniques and methodologies which make the physical response characterization and physical/digital cloning of PUFs possible.**

I. Introduction

Physically Unclonable Functions (PUF) are a novel approach to secure integrated circuits (IC). They are initially introduced by Pappu et. al [1] and the first hardware implementations were realized by Gassend et. al [2]. Those early implementations led to additional criteria for future implementations [3], [4]. Yet even in light of this criteria, the plethora of PUF implementations make them extremely difficult to classify and categorize, and therefore, different classifications exists in literature which take different PUF characteristics into account. As a result, evaluating the level of security provided by a given PUF implementation is extremely difficult.

In this work we focus on physical resiliency of systems implementing PUFs, and hence, we review the necessary criteria for such applications. Several hardware vendors have begun to roll out products featuring PUF implementations as a replacement to NVM key storage [5], [6], and in both cases the SRAM PUF [7] was chosen for the implementation. In general vendors focus on two particular characteristics provided by PUFs: 1) tamper evidence, and 2) insusceptibility to physical readout. Based on these criteria, we investigate several implementations and compare their deserved and supplied properties. Subsequently we can identify inherent weakness in the current crop of PUF implementations.

The rest of the paper is organized as follows. In section II we summarize required background information for the concept of PUFs and the SRAM PUF in particular. Section III discusses different aspects of current SRAM PUF attacks and points out the flow of events for the attacks to succeed. In the Discussion, section IV, the usefulness of SRAM PUFs is estimated and finally, in section V, we conclude the paper.

II. Background

State-of-the art invasive attacks have long demonstrated that decrypted data can be extracted directly from the device, independent of the hardware encryption function used [8]. In these attacks deciphered data is recovered by targeting buses located after the hardware decryption function. Moreover, such attacks can target any and all registers on the device including the registers storing the PUF response [9]. However, the inherent tamper-evidence in PUFs should ensure that additional attack sensors are not required. Nevertheless, attacks mitigating multiple physical sensors have also been demonstrated [10]. Though such devices did not contain a PUF, they demonstrate how difficult it is to reliably implement tamper evidence in a circuit.

Though many applications for PUFs have been proposed, products in the field today primarily utilize PUFs in the key storage role. In fact, physically linking a device's encryption function to one or more physical parameters of the device was introduced before the PUFs as we know them today had been presented [11]. Though PUFs can be realized by measuring physical device parameters with sensors [3], the vast majority of PUF implementations instead generate a unique digital response. The general assumption is that the device will still exhibit sufficient tamper-evidence to prevent attacks. Such schemes allow the manufacturer to produce per device encryption keys that are difficult to read out for the attacker. The remaining implementations can loosely be categorized as PUFs based on the settling state of memories and PUFs based on the intrinsic timing variations on the device.

A. Timing Based PUFs

Timing based PUFs include, among others, ring oscillator PUFs and arbiter PUFs [7]. These implementations evaluate the intrinsic timing behavior of logic cells inside the circuit. All timing based PUFs require additional dedicated circuitry for their realization. This makes retrofitting existing implementations impossible. Ring oscillator PUFs are large in physical size, making them unfit for economic reasons. Arbiter PUFs are relatively small due to their simple circuitry, however even this amount of overhead has made such PUFs unattractive as compared to SRAM PUFs.

Moreover, timing based PUFs are susceptible to modeling attacks [12]. Such attacks require collecting a large set of Challenge Response Pairs (CRPs) to model the intrinsic device behavior. These attacks, commonly referred to as digital clones, do not attack the physical behavior of the device, but merely utilize a digital model of the device behavior. Such attacks assume that the attacker is unimpeded in obtaining CRPs from the hardware. However, in a real-world scenario, depending on the implementation, obtaining a sufficient amount of CRPs may be impossible.

In the following, we will generally focus on memory-based PUFs and SRAM PUFs in particular. SRAM PUFs are significantly more economical in scenarios where a static key is generated from the PUF response and used for decryption.

B. Settling State Based PUFs

Settling state based PUFs utilize bistable circuits such as registers, flip-flops or SRAM cells [7]. The most dense implementation is the SRAM PUFs as no additional circuitry is required apart from the bistable circuit and the addressing logic. Also, SRAM is readily available in nearly all digital ICs and can be retrofitted for this purpose. SRAM PUFs also have very predictable performance across all production and operating conditions [13]. This also makes SRAM PUFs particularly attractive for manufacturers, hence their prevalence.

An SRAM cell is the connection of two inverters into a loop plus two transistors for read and write operations. Figure 1 shows the schematic for a single SRAM cell. The two inverters consist of transistors P1, N1 and P2, N2, the read and write transistors are S1, S2. Though the transistors are placed relatively close together, their characteristics can differ significantly. For example, many implementations place the drains of the two inverters close together with the corresponding source diffusions far apart. This results in opposite physical directions of transistor layout and thus can have significant influence on the characteristics of the inverters. As the SRAM cell during startup of VDD exhibits an undetermined state, these different characteristics of the two inverters bias the resulting (stable) state after startup. The individual SRAM cell will fall into either of it's two stable states depending on the bias and store the value accordingly. The SRAM PUF uses those startup values to derive a secret key for encryption.

As the response of each individual SRAM bit is relatively independent of it's neighbors, modeling attacks are not possible in SRAM. Instead, SRAM PUFs can only be characterized directly by extracting the SRAM data contents after startup. An overview of potential characterization techniques for SRAM PUFs are presented in section III.

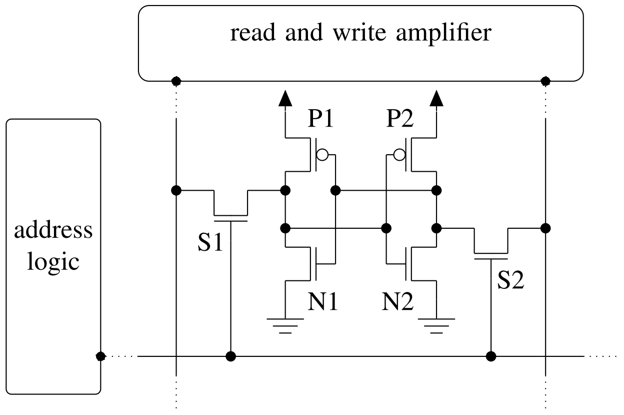


Fig. 1: 6T SRAM cell circuit.

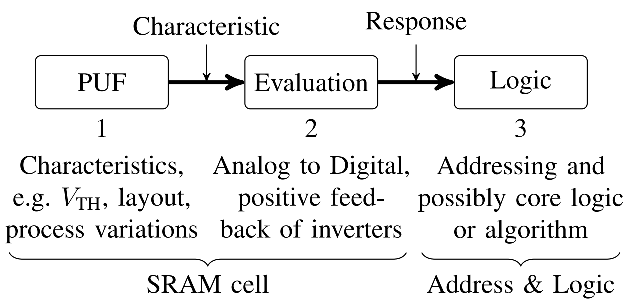
III. SRAM PUF Attacks

When the PUF response is utilized in a memory encryption scheme, the attacker may find ways to circumvent the encryption completely. For encrypted data to be processed, it must first be deciphered in the device's core [10], [8]. In this scenario, the cryptographic implementation only provides a means of obfuscating the data from the attacker. Hence, the cryptographic implementation is irrelevant, as is the PUF response and derived key material.

To successfully prevent an attacker to gain knowledge of the unencrypted memory contents, the PUF must be tamper evident. This means, that the invasive attacks must alter the PUF characteristic in such a way that the key material derived from the PUF response is lost unrecoverable. This is an important requirement since additional attack detectors or hardware sensors implemented on an IC could easily be thwarted by an invasive attacker [10], [14].

From a physical point of view, a cryptosystem containing a PUF could be described as a three stage device. Figure 2 shows the three stages of the *PUF*, the *Evaluation* and the *Core* parts. The PUF part contains the actual instance dependent characteristics. In an SRAM PUF system, this would be equivalent to the individual inverters drive strengths and threshold voltages. This characteristic is then used by the evaluation part to derive a response information. Again, with the SRAM PUF, this equals electrical feedback loop which derives the PUF response (bit state) from the different threshold voltages or drive strengths. The third part is the core logic device which processes the data afterwards in order to derive key material or even encrypt and decrypt memory. In current literature, the combination of the first two modules, the PUF and the evaluation, are referred to as making up the PUF.

An SRAM PUF could be attacked in various ways. To circumvent a PUF implementation an attacker may choose to either: (1) identify the characteristic PUF behavior for modeling, emulation or cloning, (2) retrieve the full PUF response for modeling, emulation or cloning, or (3) circumvent the PUF completely and extract the necessary data elsewhere on the device.



Characteristic
Response
PUF
Evaluation
Logic
1
2
3
Characteristics, e.g. V_{TH}, layout, process variations
Analog to Digital, positive feed-back of inverters
Addressing and possibly core logic or algorithm
SRAM cell
Address & Logic

Fig. 2: A general PUF system divided into three parts for attack discussion. Note that the often discussed helper data would be integrated into the core logic. The second row depicts the individual attack scenarios, the third row specifies the mapping of the modules to SRAM PUFs.

Recovering the PUF response itself implies recovering the quantification provided by the evaluation circuit, whereas extracting the characteristic behavior may be more complicated. In the case of SRAM PUFs, the PUF are the startup values where as the characteristic behavior are the relative biases that lead to these values. Thus, when evaluating the logical state of an SRAM cell, the PUF response is analyzed as in [9]. Whereas when the individual drive strengths are evaluated, for example based on the likeliness of either state as conducted in [15], the characteristics is analyzed.

To make use of the PUF characteristics, the evaluation and core logic must be understood by an attacker. Instead, an attacker could ignore the PUF and extract the unencrypted data directly from the device This eliminates much of the reverse-engineering that might otherwise be necessary. Thus, PUF implementations should consider such attack vectors as well.

Vendors of secure IC's have already identify that the characteristic behavior must be defined by surrounding environment as well [16]. Unfortunately, SRAM PUFs are extremely resilient to changes in the surrounding environment. Within an SRAM PUF implementation the characteristic behavior is stored inside the transistors of the SRAM cells. The interconnects, feedback loops and corresponding addressing and circuitry make up the evaluation circuit. Recent works suggest, that tampering with the device does not significantly change SRAM PUF responses. For example, thinning the device to allow for access to the individual to SRAM cells can remove most of the silicon prior to the PUF changing it's behavior [15].

The SRAM must also be integrated into the system that surrounds it. Attacking the surrounding circuitry, will also not change the intrinsic behavior of the PUF. As a result by disabling the logic core, it is possible to recover the PUF response without altering the PUF's behavior [9]. In this scenario the full PUF response can be recovered by extracting the values stored within the SRAM after startup, see Figure 3. Physical read-out of coating PUFs for example would not be possible as any physical alterations to the device would alter the charateristic behavior [3]. Hence an attacker would be limited solely to less invasive characterization techniques.

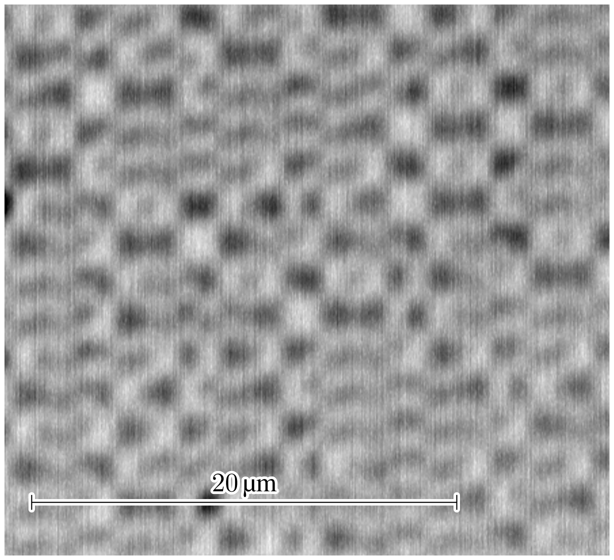


Fig. 3: SRAM contents of a PoC PUF implementation after laser irradiation on an MSP430 microcontroller with a 180 nm process. The laser wavelength used for imaging was 1300 nm. Due to the small feature sizes of the device, the layout of the SRAM cells is no longer visible in the image. Instead the irregular structures represent the logical values stored within the device, allowing for the full PUF response to be recovered.

IV. Discussion

As compared to delay-based PUFs, memory PUFs generally have individual evaluation circuits. In the case of SRAM, each individual SRAM cell stores a single bit of the full PUF response. Moreover, the full PUF response is generated at startup and stored within the SRAM array. This makes memory PUFs particularly susceptible to physical read-out utilizing laser stimulation [17], [9], as well as emission analysis [18], [15]. Moreover, SRAM PUFs are particularly resilient to influence from external sources. They are especially resilient to temperature variations [13]. Additionally as much as 99% of the device thickness can be removed without influencing the PUF response [15].

However, attacks can also target the circuitry surrounding the PUF. In a system that utilizes PUFs, the registers storing the PUF response can be readout using fully-invasive micro-probing attacks [10], [8]. The resiliency of such PUF implementations to influence from external sources ensures that such attacks will be feasible. Furthermore, such attacks would instead target the core of a secure device where data is a processed in its decrypted form. Hence attackers can circumvent the PUF completely without the risk of changing the physical response. As such the PUF adds only a layer of obfuscation to the fully-invasive attacker.

Because PUFs must be integrated into a system as a whole, the security of the entire system must be considered. Solutions utilizing PUF mechanisms today lack sufficient mechanisms to prevent physical tampering with the device. This makes most PUF implementations to date susceptible to physical attacks and read-out. To mitigate this threat the PUF response must be intrinsically tied to the integrity of the device. Sensor based PUF solutions offer a potential solution to this problem [3]. As a result, by utilizing the PUF response to generate the encryption keys for the on-die encryption, any alteration to the integrity of the device would alter the key, rendering the device inoperative [11].

V. Conclusion

When it comes to physical attacks, the threat arises from insufficient integrity checks on the device. One of the most interesting properties of PUF, is tamper evidence. However, this is generally assumed and seldom verified. Moreover, resiliency to physical attacks is at odds with the vendor's goal of producing robust circuits. Any physical alterations to a secure circuit should ideally render the device completely inoperative. In reality, SRAM PUF implementations boast their resiliency to influence from external parameters [13]. Optimizing for robustness against environmental parameters ensures that the circuit becomes less tamper evident. As such most current implmenetations lack sufficient protection against physical attacks.

VI. Acknowledgements

This research was supported by the Helmholtz Research School on Security Technologies. We would like to thank DCG Systems for continuous support on our OptiFIB system and Hamamatsu KK Japan and Germany for support on our PHEMOS system.

References

[1] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[2] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 148–160, 2002.

[3] P. Tuyls, G.-J. Schrijen, B. Škorić, J. Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, ser. Lecture Notes in Computer Science, L. Goubin and M. Matsui, Eds. Springer Berlin Heidelberg, 2006, vol. 4249, pp. 369–383.

[4] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Controlled physical random functions," in *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*. IEEE, 2002, pp. 149–160.

[5] Microsemi, "Microsemi and Intrinsic-ID Deliver Integrated Security Solutions for Government Applications," http://investor.microsemi.com/releasedetail.cfm?ReleaseID=731250, Jan. 2013.

[6] NXP Semiconductors N.V, "NXP strengthens SmartMX2 security chips with PUF anti-cloning technology," http://www.nxp.com/news/press-releases/2013/02/nxp-strengthens-smartmx2-security-chips-with-puf-anti-cloning-technology.html, Feb. 2013.

[7] A.-R. Sadeghi and D. Naccache, Eds., *Towards Hardware-Intrinsic Security: Foundations and Practice*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

[8] O. Kömmerling and M. Kuhn, "Design principles for tamper-resistant security processors," *USENIX Workshop on Smartcard Technology, Chicago, IL (10–11 May 1999) http://www. cl. cam. ac. uk/Research/Security/tamper*, 1999.

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

[10] C. Tarnovsky, "Hacking the smartcard chip," in *Blackhat DC 2010*, Arlington, VA, Feb. 2010.

[11] O. Kömmerling and F. Kömmerling, "Anti tamper encapsulation for an integrated circuit," Patent US 20 010 033 012 A1, 2001.

[12] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *CCS '10: Proceedings of the 17th ACM conference on Computer and communications security*. ACM Request Permissions, Oct. 2010.

[13] R. van den Berg, B. Skoric, and V. van der Leest, "Bias-based modeling and entropy analysis of PUFs," in *TrustED '13: Proceedings of the 3rd international workshop on Trustworthy embedded devices*. ACM Request Permissions, Nov. 2013.

[14] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. Krissler, C. Boit, and J.-P. Seifert, "Breaking and entering through the silicon," in *CCS '13: Proceedings of the 20th ACM conference on Computer and communications security*, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1145/2508859.2516717

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

[16] NXP Semiconductors N.V, "PUF - physical unclonable functions – protecting next-generation smart card ics with sram based pufs," http://www.nxp.com/documents/other/75017366.pdf, Feb. 2013.

[17] 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*, ser. SISW '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 65–.

[18] D. Nedospasov, J.-P. Seifert, A. Schlosser, and S. Orlic, "Functional integrated circuit analysis," in *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, 2012, pp. 102–107.