DRAM PUFs in Commodity Devices

Wenjie Xiong, André Schaller, Nikolaos Athanasios Anagnostopoulos, Muhammad Umair Saleem, Sebastian Gabmeyer, Stefan Katzenbeisser, Jakub Szefer

***Abstract*—A Physically Unclonable Function (PUF) is a unique and stable physical characteristic of a piece of hardware, which emerges due to variations in the fabrication processes. PUFs have been shown to be a promising cryptographic primitive for key storage, hardware-based device authentication, and identification. This paper shows how to realize the recently proposed decay-based intrinsic DRAM PUFs in commercial off-the-shelf systems. A key advantage of the new PUF is that it can be queried during run-time of a Linux system, and requires no hardware modifications, nor use of FPGAs.**

I. Introduction

The continued miniaturization and cost reduction of system-on-chip devices has enabled the creation of ubiquitous smart devices. However, the proliferation of such smart devices creates new security vulnerabilities, as devices such as smartphones, smart appliances, and sensors often collect critical or private information. If these devices lack the implementation of sufficient security mechanisms, such sensitive information can be manipulated or leaked. Critical challenges in securing these devices are how to provide robust device authentication and identification mechanisms, and means to store long-term cryptographic keys in a secure manner that minimizes the chances of their illegitimate access.

A classic approach to device identification is to embed cryptographic keys in each device by burning them in at manufacturing time. However, this solution comes with potential pitfalls, such as increased production complexity as well as rather limited protection against key extraction attempts. In order to address these issues, researchers have proposed Physically Unclonable Functions (PUFs) [1]. PUFs leverage the unique behavior of a device due to manufacturing variations as a hardware-based fingerprint. A PUF instance is extremely difficult to replicate, even by the manufacturer. Hence, PUFs have been proposed as cryptographic building blocks in security primitives and protocols for: authentication and identification, hardware-software binding, remote attestation, and secret key storage [1], [2]. So far, most types of PUFs in digital electronic systems (such as arbiter PUFs [1]) require the addition of dedicated circuits to the device, and thus, increase manufacturing costs and hardware complexity. Consequently, there is great interest in the so-called intrinsic PUFs [2], which are PUFs that are already inherent to a device.

W. Xiong and J. Szefer are with Yale University, New Haven, CT, USA. E-mails: wenjie.xiong@aya.yale.edu, jakub.szefer@yale.edu

A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, and S. Katzenbeisser have been with Technical University of Darmstadt, Darmstadt, Hessen, Germany. E-mails: andre@andreschaller.de, umairsaleem-chaudhry@gmail.com, sebastian@gabmeyer.net

N. A. Anagnostopoulos, and S. Katzenbeisser are currently with University of Passau, Passau, Bayern, Germany. E-mails: {nikolaos.anagnostopoulos, stefan.katzenbeisser}@uni-passau.de

Intrinsic PUFs are considered an attractive low-cost security primitive, as they use standard hardware that can be found in commercial off-the-shelf devices, without requiring any hardware modifications or additions. The most prominent example of intrinsic PUFs are PUFs based on Static Random-Access Memories (SRAMs) [2], [3], which draw their characteristics from the startup values of bi-stable SRAM memory cells. Although SRAM PUFs are known to have good PUF characteristics, the PUF measurements must be extracted during a very early boot stage (before the SRAM is used). Consequently, the derived key can only be used at this time, or must be saved to a different memory region, which may cause security problems. Alternatively, a dedicated IC chip with an SRAM can be added to the system, which can be accessed at run-time, but this requires hardware changes. Without adding extra hardware, an error-based SRAM PUF can be accessed at run-time as well [4], but to query the PUF, still, hardware modification is required to change the supply voltage.

An alternative to the SRAM PUFs, are DRAM PUFs presented in this paper, which were designed based on our existing work [5]. We show how to extract DRAM PUFs from commercial systems, requiring no special hardware modifications nor FPGA setup, and to provide a practical solution to query DRAM PUFs during run-time on a Linux system. Our decay-based DRAM PUFs allow for repeated accesses, which overcomes the limitation of previous intrinsic memory-based PUFs that were available at device startup only. Furthermore, our PUF can be accessed at run-time without hardware changes. Moreover, the capacity of DRAM is magnitudes larger than SRAM, allowing to use many more bits in order to derive larger cryptographic key material, or to segment DRAM into several logical PUFs. DRAM is an excellent candidate for an intrinsic PUF as DRAM is an integral part of many of today's commodity platforms and can be found in many "smart" devices, such as smartphones or smart thermostats. Recent use of embedded DRAM [6] in low-cost microprocessors will further increase the availability of DRAM as part of mobile and embedded computing platforms. Our work on run-time accessible DRAM PUFs has already served and will continue to serve as a catalyst for higher security, especially in low-end IoT devices currently utilized in sensors or other smart devices, for example in health care, home automation, transportation, or energy grids.

II. Extracting DRAM PUFs from Commodity Devices

A. DRAM Decay Characteristics

In a DRAM cell, a single data bit is stored in a capacitor and can be accessed through a transistor, as shown in Figure 1 (i).

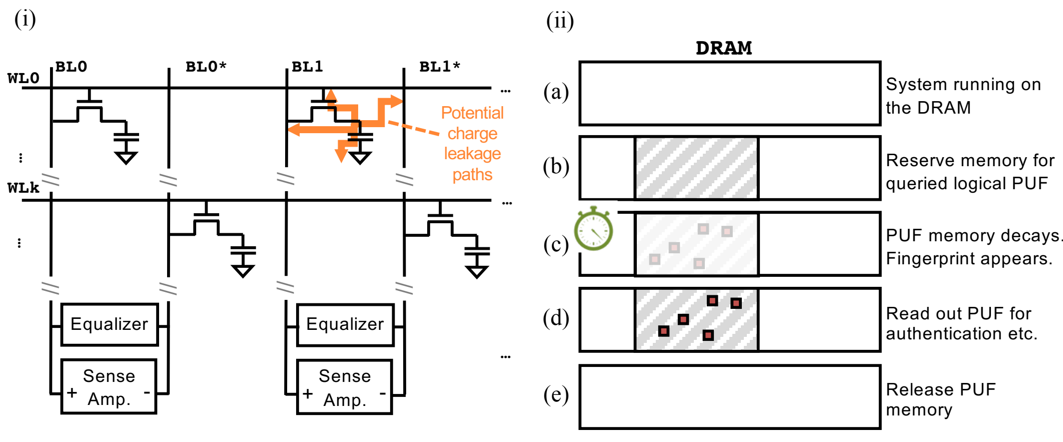
(i)
WL0
BL0
BL0*
BL1
BL1*
Potential charge leakage paths
WLk
Equalizer
Equalizer
Sense Amp.
Sense Amp.
+
-
(ii)
DRAM
(a)
System running on the DRAM
(b)
Reserve memory for queried logical PUF
(c)
PUF memory decays. Fingerprint appears.
(d)
Read out PUF for authentication etc.
(e)
Release PUF memory

Fig. 1. (i) A single DRAM cell consists of a capacitor and a transistor, connected to a word-line (WL) and a bit-line (BL or BL*); arrows indicate leakage paths for dissipation of charges that lead to PUF behavior. (ii) The five steps required for run-time access of a DRAM PUF. Only during steps (b) – (d) the memory associated with the PUF is not usable for other processes.

DRAM cells are grouped in arrays, where each row of the array is connected to a horizontal word-line. Cells in the same column are connected to a bit-line. All bit-lines are coupled to sense-amplifiers that amplify small voltages on bit-lines to levels such that they can be interpreted as logical zeros or ones. In order to access a row, all the bit-lines are *precharged* to half the supply voltage, $V_{DD}/2$; subsequently the word-line is *enabled*, connecting every capacitor in that row with its bit-line. The sense amplifier will then drive the bit-line to V_{DD} or 0V as the digital readout value, depending on the charge on the capacitor.

DRAM cells require periodic refresh of the stored charges, as otherwise the capacitors lose their charge over time, which is referred to as DRAM cell decay or leakage. The hardware memory controller takes care of periodic refresh, whose interval is defined by the vendor and is usually 32ms or 64ms. Without this periodic refresh, some of the cells will slowly decay to 0, while others decay to 1, depending on whether they are the so-called true cells or anti-cells, respectively. Because of the manufacturing variations among DRAM cells, some cells decay faster than others, which can be exploited as basis for the PUF.

B. Run-time DRAM PUF Access on Commodity Devices

Our process to extract a DRAM PUF measurement at run-time in a commodity system using Linux is summarized in Figure 1 (ii). The starting point (a) comprises the DRAM module being configured for ordinary use, where the memory controller periodically refreshes all of the cells' content. In the next step (b), the PUF memory region defined by starting address ($addr$) and size ($size$) is reserved, e.g., using memory ballooning (in Section II-B). Furthermore, the refresh for the PUF region is disabled and the initialization value ($inital$) is written to the region. Next (c), for a given decay time (t), the memory region containing the PUF is not accessed to let the cells decay. After the decay time has expired (d), the memory content is read in order to extract the PUF measurement. At the end (e), normal memory operations are restored and the memory region is made available to the operating system (OS) again. As the decay time and the positions of the flipped bits are unique for individual DRAM regions, the "pattern" of flipped bits for a given decay time t can serve as the PUF response.

We implemented and tested our DRAM PUF construction on two popular platforms, the PandaBoard ES Revision B3 and the Intel Galileo Gen 2. The PandaBoard houses a TI OMAP 4460 System-on-Chip (SoC) module that implements 1GB of DDR2 memory from ELPIDA in a Package-on-Package (PoP) configuration, which operates at 1.2V. The Intel Galileo has an Intel Quark SoC X1000 SoC with two 128MB DDR3 chips from Micron, operating at 1.5V. The two physical DRAM modules are accessed in parallel and located on the same PCB as the processor.

We implemented two different approaches to query the PUF. The first approach uses a modified firmware in order to obtain PUF measurements during the boot phase. Second, we implemented a kernel module-based solution that enables PUF queries during the run-time of a Linux operating system. The kernel module is a proof-of-concept of the run-time accessibility of the proposed DRAM PUF.

Deactivating DRAM refresh for PUF access during device operation is a non-trivial task: when DRAM refresh cycles are disabled, critical data (such as data belonging to the OS or user-space programs) will start to decay and the system will crash. In our experiments, the Intel Galileo board with Yocto Linux crashes about a minute after DRAM refresh is disabled. Therefore, we present a customized solution which allows us to refresh critical code, but leaves PUF areas untouched. This solution is based on two techniques dubbed *selective DRAM refresh* and *memory ballooning*. The former allows for selectively refreshing the memory regions occupied by the OS and other critical applications so that they run normally and do not crash. Memory ballooning, on the other hand, safely reserves the memory region that corresponds to a logical PUF without corrupting critical data and also protects the memory region from OS and user-space programs accesses, to let the cells decay during PUF measurement.

Selective DRAM Refresh. On some devices, such as the

TABLE I
METRICS OF LOGICAL PUF INSTANCES MEASURED AT DIFFERENT DECAY TIMES.

decay time	device family	min. J_{intra}	max. J_{inter}	fractional entropy H_t/N	avg. decay rate	max. fractional intra-HD	min. fractional inter-HD
120s	PandaBoard	0.4634	0.0102	0.0271	0.0041	0.0045	0.0038
	IntelGalileo	0.7712	0.0038	0.0062	0.0009	0.0003	0.0012
180s	PandaBoard	0.4382	0.0168	0.0754	0.0102	0.0083	0.0139
	IntelGalileo	0.8361	0.0044	0.0169	0.0024	0.0005	0.0032
240s	PandaBoard	0.4087	0.0258	0.0893	0.0159	0.0101	0.0244
	IntelGalileo	0.6261	0.0049	0.0250	0.0041	0.0020	0.0057
300s	PandaBoard	0.4222	0.0405	0.1478	0.0202	0.0123	0.0238
	IntelGalileo	0.7944	0.0055	0.0353	0.0061	0.0013	0.0080
360s	PandaBoard	0.3484	0.0342	0.1440	0.0234	0.0206	0.0279
	IntelGalileo	0.8276	0.0072	0.0541	0.0093	0.0022	0.0124

PandaBoard, DRAM consists of several physical modules or logical segments, where the refresh of each module/segment can be controlled individually. In this case, the PUF can be allocated in a different memory segment from the OS and user-space programs. When querying the PUF, only the refresh of the segment holding the PUF is deactivated, while the other segments remain functional.

On other devices, e.g., the Intel Galileo, the refresh rate can only be controlled at the granularity of the entire DRAM. Refresh at segment granularity is not possible. However, memory rows can be refreshed implicitly once they are accessed due to a read or a write operation. When a word line is selected, as a result of a memory access, the sense amplifier drives the bit-lines to either the full supply voltage V_{DD} or back down to 0V, depending on the value that was in the cell. In this way, the capacitor charge is restored to the value it had before the charges leaked. Using the above principle, even if the refresh of the whole memory is disabled, a set of memory rows can be refreshed by issuing a read to one word in each of the rows.

Ballooning System Memory. Memory ballooning is a mechanism for reserving a portion of the memory so as to prevent the memory region from being used by the kernel or any application. This approach allows to specify the physical address ($addr$) and size ($size$) of the memory region that will be reserved for the PUF. Once PUF memory is "ballooned", DRAM refresh is disabled, and selective refresh is enabled for the non-PUF memory region. After PUF access is finished, the balloon can be deflated and the memory can be restored to normal use.

C. Security Assumptions

DRAM PUFs differ from classic memory-based PUFs, as they can be evaluated during run-time. Disabling and enabling DRAM refresh includes writing to hardware registers, a task which can only be performed by the kernel. Furthermore, accessing the memory dedicated to the PUF is restricted to the kernel as well. Thus, a crucial security assumption is that firmware and operating system are trusted and an attacker never gains root privileges, similar to a controlled PUF setting [7].

III. EVALUATION OF DRAM PUF CHARACTERISTICS

We performed measurements using four different PandaBoards and five Intel Galileo devices. Furthermore, given the large amount of memory present, we measured two 32KB PUF regions on each device, resulting in eight different PUFs for the PandaBoard and ten PUFs for the Intel Galileo. Each logical PUF was measured at five different decay times t, with 50 measurements each. Based on these measurements we evaluated the robustness, uniqueness, and randomness, as well as the temperature dependency of DRAM PUFs. The results are shown in Table I.

The characteristics of DRAM PUFs are different compared to SRAM PUFs. Rather than being considered as an array of bits, a DRAM PUF response is a set of flipped bits within a memory region. Thus, classic metrics that are used to evaluate memory-based PUFs, which are usually based on fractional Hamming Distance (HD), do not properly reflect the properties of DRAM PUFs. This effect is particularly noticeable when evaluating the uniqueness of PUF instances. In case of DRAM PUFs, it is the locations, i.e., the indices, of the bit flips, that present the uniqueness of the PUF response. If one would apply the *fractional* inter-HD, the whole 32KB measurement would be considered, including those cells that did not flip within the observed time period, resulting in a very low value (in column *min. fractional inter-HD* in Table I), which does not capture uniqueness to the full extent.

Thus, we use Jaccard index to evaluate the robustness and uniqueness of DRAM PUF. The Jaccard index is a well known metric to quantify the similarity of two sets of different size: the index results in a value of zero if both sets share no common elements and a value of one if both sets are identical. In particular, based on two PUF responses m_1, m_2, we construct two corresponding sets s_1 and s_2 that store the indices of the flipped cells. Jaccard index $J(s_1, s_2)$ is calculated as:

$$J(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}. \qquad (1)$$

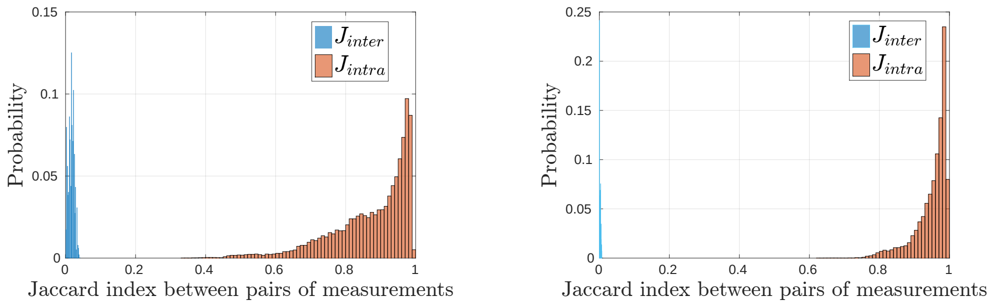

Fig. 2. Distribution of J_{intra} and J_{inter} values for (left) the PandaBoard and (right) the Intel Galileo.

Uniqueness and Robustness. In order to evaluate the uniqueness of the PUF, we consider the set of indices of DRAM cells that flipped due to decay among different PUFs. For an ideal PUF, the value of $J_{inter}(s_1, s_2)$ should be close to zero, indicating that two logical PUFs rarely share flipped bits. Indeed, as Table I shows, our DRAM PUFs depict an almost perfect behavior with the Intel Galileo having a maximum of $J_{inter} = 0.0072$ at $t = 360$s. The PandaBoard shows larger values with a maximum value of 0.0405 at $t = 300$s which, however, is still close to the optimal value of zero.

In order to quantify the inherent noise in the PUF measurements and consequently PUF robustness, we computed the Jaccard index $J_{intra}(s_1, s_2)$ between two sets containing the indices of flipped bits in two measurements of the *same* logical PUF at identical decay times. An ideal PUF should show values close to one, indicating that responses are stable.

Figure 2 displays the distributions of $J_{intra}(s_1, s_2)$ and $J_{inter}(s_1, s_2)$ of all measurements, corresponding to identical decay times, for both device types. A clear divide between the two distributions indicates that individual devices can be distinguished perfectly, while the PUF response is stable over subsequent measurements.

Entropy. In order to generate cryptographic keys from the PUF response, PUF measurements must exhibit sufficient entropy. We estimate the Shannon entropy of DRAM PUF responses as follows. We again consider the set s of indices of DRAM cells that flipped after time t. We denote with k the cardinality of s and with N the total number of DRAM cells. Assuming that the flipped bits are distributed uniformly, as confirmed by our experiments, the probability of observing one set s is: $P(v) = 1/\binom{N}{k}$. The Shannon entropy of DRAM PUF for a given decay time can thus be calculated using

$$H_t = log_2 \binom{N}{k}. \tag{2}$$

Note, that simply observing the number of bits decaying after time t has elapsed, is not sufficient for determining k, as the bit decay will be due to two effects: (i) short-term noise that must be corrected and (ii) stable long-term decay characteristics. In order to approximate k, indicating the stable PUF characteristics, multiple measurements for a single PUF can be averaged in order to eliminate the noise component. Table I lists the fractional entropy H_t/N computed this way. We observe that the entropy is significantly bigger on the PandaBoard, indicating more bit flips than on the Intel Galileo. This is most likely due to the different technologies used to implement DRAM cells.

Decay Dependency on Temperature. In Figure 3 (top), we show the dependency between the temperature and the decay rate of DRAM modules on the Intel Galileo and the PandaBoard. In order to control the temperature, we used a metal ceramic heater to heat the surface of DRAM modules to the desired temperature.

Although temperature affects the decay rate significantly, it does not change the decay characteristics much; instead, it affects decay time: We observed that by using a carefully chosen smaller decay time $t'_{T'} < t$ at a larger temperature $T' > T$, the same PUF response can be obtained as with decay time t at temperature T. In our experiments, we derive the following dependency for the Intel Galileo boards:

$$t'_{T'} = t * e^{-0.0662*(T'-T)}. \tag{3}$$

Hence, if the PUF is evaluated at a different temperature than during enrollment, this can be compensated through adapting the decay time according to Equation (3). In order to support this statement, we calculated the noise J_{intra} between an enrollment measurement at room temperature (40°C) and a measurement taken at a different temperature by adjusting the decay time. For this purpose, we created reference measurements at room temperature with decay times $t_x = \{$120s, 180s, 240s, 300s, 360s$\}$. In a next step, we used equivalent decay times $t'_{T'}$ that correspond to temperatures $T' = \{$40°C, 50°C, 60°C$\}$ and measured the PUF accordingly. As shown in Figure 3 (bottom), for all measurements, J_{intra} lies within the usual noise level. Thus, differences in temperature can be accommodated by adjusting decay time accordingly. An attacker may try to change the ambient temperature in order to influence the bit flip characteristics, but a legitimate user can compensate the temperature effect by adjusting the decay time. As shown in [8], such temperature dependency allows to use DRAM as a temperature sensor. DRAM regions other than the PUF region can be used as temperature sensors to decide the decay time for the PUF, such that no additional sensor is needed.

IV. Future Research Directions on DRAM PUFs

Our work demonstrates that the proposed run-time accessible DRAM PUFs can be deployed on existing commercial

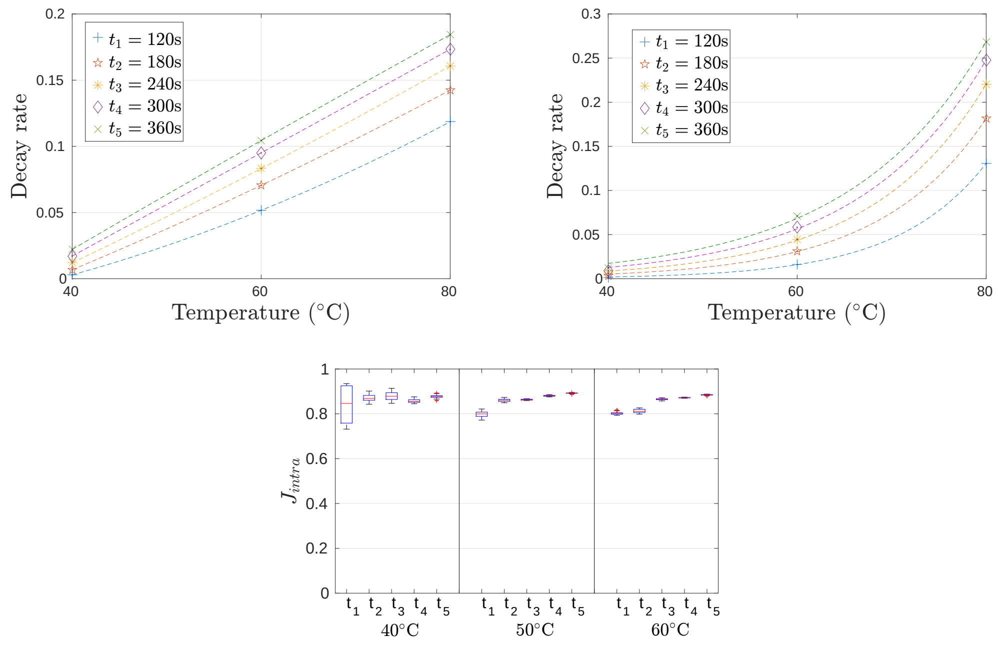

Fig. 3. (top) Relation between the temperature and the decay rate measured on (left) the PandaBoard and (right) the Intel Galileo. (bottom) J_{intra} values (i.e., similarity) of enrollment measurements taken at room temperature and measurements at higher temperatures $T' = \{40°\text{C}, 50°\text{C}, 60°\text{C}\}$, with adjusted decay times $t'_{T'}$.

systems as a hardware security primitive of minimal cost, especially in IoT devices and embedded systems that lack other security mechanisms. We, therefore, believe that our work will continue to serve as a catalyst for higher security, especially in low-end IoT devices currently utilized in health care, home automation, transportation, or energy grids. The results of this work have paved the way for future research on DRAM PUFs.

Characteristic of DRAM PUFs. The properties of decay-based DRAM PUFs have been further investigated. The temperature dependency of DRAM decay can be further evaluated under a wider temperature range, such as in [9], [10]. We also anticipate more research projects considering error correction models for DRAMs.

Potential Attacks and Countermeasures of decay-based DRAM PUFs. One potential attack is to access the PUF and obtain a digital copy of the PUF. In the current implementation, it requires kernel privilege to access the runtime DRAM PUF. Thus, a controlled PUF countermeasure can be built leveraging the privilege level to prevent the attacker from accessing PUF response. It is also possible to attack DRAM PUF on the physical level. Effects such as temperature, radiation, DRAM disturbance error could influence the robustness of the DRAM PUFs. Also, there may be correlation in the locations of bit flips in the DRAM PUF responses. These physical properties need to be further investigated.

Schemes for Key Storage and other Applications. An index-based helper data system for key storage and a mutual authentication scheme are proposed in [10]. There is only a small percentage of bit flips in the DRAM PUF responses, and thus, the proposed helper data system chooses a fraction of the bits in the response to extract bit strings with high-entropy. Further, different decay times can be used to generate more DRAM PUF challenge-response pairs for device authentication and software protection [9].

Novel PUFs in DRAM. Our PUFs leverage the DRAM decay to induce bit flips. Recent works have successfully discovered PUFs based on other operations that can cause bit flips in DRAM, such as the rowhammer effect [11] and changes in the latency of DRAM operations [12].

V. Conclusion

In this work, we presented intrinsic PUFs that can be extracted from DRAM in commodity devices. In contrast to existing DRAM and SRAM PUFs, we demonstrate a system model that is able to query the PUF instance directly during run-time using a Linux kernel module on unmodified, commodity devices, in particular the PandaBoard and Intel Gallileo. New metrics based on the Jaccard index have validated the robustness, uniqueness and randomness of DRAM PUFs. Consequently, our work presents a method for device authentication by leveraging DRAM in commodity devices. Finally, the impact of our research approach has been significant as other researchers have used our initial work as an inspiration for a more extensive investigation of the properties of DRAM, as a basis for the implementation of DRAM-based security schemes and cryptographic protocols, and as a motivation for the development of other types of DRAM PUFs.

References

[1] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the Design Automation Conference*, 2007, pp. 9–14.

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

[3] 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 *Proceedings of the Cryptographic Hardware and Embedded Systems*, 2012, pp. 283–301.

[4] A. Bacha and R. Teodorescu, "Authenticache: harnessing cache ECC for system authentication," in *Proceedings of International Symposium on Microarchitecture*, 2015, pp. 128–140.

[5] W. Xiong, A. Schaller, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Run-time accessible DRAM PUFs in commodity devices," in *Proceedings of the Cryptographic Hardware and Embedded Systems*, 2016, pp. 432–453.

[6] S. Rosenblatt, D. Fainstein, A. Cestero, J. Safran, N. Robson, T. Kirihata, and S. S. Iyer, "Field tolerant dynamic intrinsic chip ID using 32 nm high-k/metal gate SOI embedded DRAM," *IEEE Journal of Solid-State Circuits*, pp. 940–947, 2013.

[7] B. Gassend, M. V. Dijk, D. Clarke, E. Torlak, S. Devadas, and P. Tuyls, "Controlled physical random functions and applications," *ACM Transactions on Information and System Security*, vol. 10, no. 4, p. 3, 2008.

[8] W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser, and J. Szefer, "Spying on Temperature using DRAM," in *Proceedings of the Design, Automation, and Test in Europe*, ser. DATE, March 2019.

[9] S. Sutar, A. Raha, and V. Raghunathan, "D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems," in *Proceedings of the International Conference on Compliers, Architectures, and Synthesis of Embedded Systems*, 2016, pp. 1–10.

[10] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Škorić, S. Katzenbeisser, and J. Szefer, "Decay-Based DRAM PUFs in Commodity Devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 462–475, 2019.

[11] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, S. Katzenbeisser, and J. Szefer, "Intrinsic rowhammer PUFs: Leveraging the rowhammer effect for improved security," in *Proceedings of the International Symposium on Hardware Oriented Security and Trust*, 2017, pp. 1–7.

[12] J. S. Kim, M. Patel, H. Hassan, and O. Mutlu, "The DRAM latency PUF: Quickly evaluating physical unclonable functions by exploiting the latency-reliability tradeoff in modern commodity DRAM devices," in *Proceedings of the International Symposium on High Performance Computer Architecture*, 2018, pp. 194–207.

Wenjie Xiong received her Ph.D. degree from the department of Electrical Engineering at Yale University, advised by Prof. Jakub Szefer. Her research interests comprise Physically Unclonable Functions and cache side channel attacks and defenses.

André Schaller received his M.Sc. and Ph.D. from the Technical University of Darmstadt. He is currently working as a security engineering consultant in the aerospace sector. His main research interest comprised hardware-based cryptography, security of mobile and embedded systems and Physically Unclonable Functions in particular.

Nikolaos Athanasios Anagnostopoulos (S'18) is pursuing a Ph.D. degree at the Technical University of Darmstadt. He is currently working as a research assistant at the University of Passau. He is a student member of the IEEE, the IEEE SSC and CAS Societies, and the ACM. His research interests include hardware design, and security testing, with a focus on physical security mechanisms.

Muhammad Umair Saleem received his B.Sc. in Electronics Engineering from the Bahauddin Zakariya University Multan in 2012 and his M.Sc in Information and Communication Engineering from Technical University of Darmstadt in 2018. He is currently working as a Software Engineer in Germany, and his interests include Embedded Systems, Internet of Things and Software Development.

Sebastian Gabmeyer received the Ph.D. degree on model checking based verification techniques for graph transformations from the Vienna University of Technology, in 2015. He has been a post-doc at Security Engineering Group with Stefan Katzenbeisser, in Technical University Darmstadt, between January 2016 and September 2017. He is currently working in the automotive industry, where he develops security concepts for autonomous driver assistance systems. His research interests include hardware security and software verification.

Stefan Katzenbeisser (S'98–A'01–M'07–SM'12) received the Ph.D. degree from the Vienna University of Technology, Austria. After working as a Research Scientist with the Technical University of Munich, Germany, he joined Philips Research as a Senior Scientist in 2006. After holding a professorship for Security Engineering at the Technical University of Darmstadt, he joined University of Passau in 2019, heading the Chair of Computer Engineering. His current research interests include embedded security, data privacy and cryptographic protocol design.

Jakub Szefer (S'08–M'13–SM'19) received B.S. with highest honors in Electrical and Computer Engineering from University of Illinois at Urbana-Champaign, and M.A. and Ph.D. degrees in Electrical Engineering from Princeton University where he worked with Prof. Ruby B. Lee on secure hardware architectures. He is currently an Associate Professor of Electrical Engineering at Yale University where he leads the Computer Architecture and Security Laboratory (CASLAB). His research interests are at the intersection of computer architecture, hardware security, and FPGA security.