Abusing Commodity DRAMs in IoT Devices to Remotely Spy on Temperature

Florian Frank, *Graduate Student Member, IEEE*, Wenjie Xiong, *Member, IEEE*, Nikolaos Athanasios Anagnostopoulos, *Member, IEEE*, André Schaller, Tolga Arul, *Member, IEEE*, Farinaz Koushanfar, *Fellow, IEEE*, Stefan Katzenbeisser, *Senior Member, IEEE*, Ulrich Rührmair, and Jakub Szefer, *Senior Member, IEEE*

***Abstract*— The ubiquity and pervasiveness of modern Internet of Things (IoT) devices opens up vast possibilities for novel applications, but simultaneously also allows spying on, and collecting data from, unsuspecting users to a previously unseen extent. This paper details a new attack form in this vein, in which the decay properties of widespread, off-the-shelf DRAM modules are exploited to accurately spy on the temperature in the vicinity of the DRAM-carrying device. Among others, this enables adversaries to remotely and purely digitally spy on personal behavior in users' private homes, or to collect security-critical data in server farms, cloud storage centers, or commercial production lines. We demonstrate that our attack can be performed by merely compromising the software of an IoT device and does not require hardware modifications or physical access at attack time. It can achieve temperature resolutions of up to 0.5°C over a range of 0°C to 70°C in practice. The presented attack works in devices that do not have a dedicated temperature sensor on board; as the DRAM modules already present in the device are abused to spy on the temperature. To complete the work, the paper discusses practical attack scenarios as well as possible countermeasures against the new temperature-spying attacks.**

***Index Terms*— Security, privacy, IoT, DRAM, temperature.**

Manuscript received 25 July 2022; revised 26 March 2023; accepted 21 April 2023. Date of publication 27 April 2023; date of current version 15 May 2023. This work was supported in part by the U.S. National Science Foundation (NSF) under Grant 1651945; in part by the U.S. Air Force Office of Scientific Research (AFOSR) under Award FA9550-21-1-0039; in part by the German Research Foundation–Deutsche Forschungsgemeinschaft (DFG) under Project 236615297, Project 440182124, and Project 439892735; in part by the Commonwealth Cyber Initiative of the U.S. State of Virginia; and in part by the Air Force Office of Scientific Research under Award FA9550-22-1-0548. The work of Ulrich Rührmair's was supported by the AFOSR under Grant FA9550-21-1-0039. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Edgar Weippl. *(Florian Frank and Wenjie Xiong contributed equally to this work.) (Corresponding author: Florian Frank.)*

Florian Frank, Nikolaos Athanasios Anagnostopoulos, Tolga Arul, and Stefan Katzenbeisser are with the Faculty of Computer Science and Mathematics, University of Passau, Passau, 94032 Bavaria, Germany (e-mail: florian.frank@uni-passau.de; nikolaos.anagnostopoulos@uni-passau.de; tolga.arul@uni-passau.de; stefan.katzenbeisser@uni-passau.de).

Wenjie Xiong is with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: wenjiex@vt.edu).

André Schaller is with the European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT), Darmstadt, 64295 Hessen, Germany (e-mail: andre@andreschaller.de).

Farinaz Koushanfar is with the Electrical and Computer Engineering Department, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: farinaz@ucsd.edu).

Ulrich Rührmair is with the Institut für Informatik, LMU München, München, 80539 Bavaria, Germany, and also with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269 USA (e-mail: ruehrmair@ilo.de).

Jakub Szefer is with the Department of Electrical Engineering, Yale University, New Haven, CT 06511 USA (e-mail: jakub.szefer@yale.edu).

Digital Object Identifier 10.1109/TIFS.2023.3271252

I. Introduction

INTERNET of Things (IoT) devices have become more pervasive and ubiquitous than ever before in history, and are still enjoying an unbroken and continuous growth: As estimated by Taylor et al., the number of connected devices will exceed 100 billion by 2025 [1]. Their versatility allows applications in a large number of settings, including private and commercial uses in homes, companies, or factories. Unfortunately, this situation also induces pressing privacy and security problems: Once an IoT device has been compromised, attackers can gather sensitive information remotely, as, by definition, it will be connected to the World Wide Web.

To defend against such threats, intense efforts have been made to ensure that the software of IoT devices will handle any information generated by the devices' multiple sensors in a secure manner [2], [3], [4]. For example, in order to protect acoustic signals in the device environment, dedicated software may safeguard any information collected from the device's microphones [5]. However, even if all information collected from traditional sensors is properly protected, critical data may also be gathered from *other*, unprotected device components [6], [7], leading to covert and unforeseen espionage channels [8], [9], [10]. For example, while gyroscope measurements originally were considered suitable only for motion detection, researchers found ways to misuse them for measuring acoustic signals [6]. This allows gathering sound recordings and recognizing speech [5]. As another example, the power usage of a mobile phone can be used to track the current position of users [11], threatening their location privacy. As long as the ability of each device component to collect critical information is not comprehensively understood, attacks of this type are hard to prevent. They put the users' privacy and security at risk, even in the presence of standard sensor-protecting measures. Consequentially, protecting single selected sensors is insufficient to guarantee a holistic protection of users.

This article adds to this line of research. It discusses novel methods by which standard, widespread DRAM modules, which are part of every smartphone, laptop, or embedded

device, can be abused to spy on users, and to remotely monitor the ambient temperature around the host device. It is long known that such ambient temperature contains a significant amount of security-critical information:

- If measured at the victim's home, it can reveal the victim's daily routines, including holidays, or routinely repeating periods of the day during which no one is present to guard the home;
- If measured at a production line, it can reveal the temperature of the manufacturing process of a product;
- If measured at a data center, it can reveal the activity of the tenants [12].

On a technical level, our attack works by exploiting the temperature-dependent *"retention times"* of DRAM cells. The retention time of a single DRAM cell is defined as the maximal time period this cell can hold its stored value without being refreshed (i.e., how long it holds its value when the DRAM refresh operation has been disabled). It is long known that this retention time depends heavily on the ambient temperature and that it even decreases exponentially with increasing temperature [13], [14], [15]. Following this observation, we turn the individual retention times of DRAM cells into a highly sensitive thermometer. In greater detail, we observe the number of flipped cells (i.e., the number of cells that change their original content) in a given DRAM memory array with disabled refresh after a certain time period has elapsed. From this number, we can then indirectly conclude the ambient temperature. As we show in this paper, this approach allows temperature measurements over a large temperature range with a resolution of up to 0.5 °C.

Our attack can be applied *without* measuring or registering the DRAM module under attack at multiple known temperatures in advance, as long as the general temperature-dependent characteristics of the used class of DRAM modules are known to the attacker. Furthermore, no physical modifications such as hardware Trojans or the like are required, and neither is physical access to the device at attack time. Finally, and perhaps most interestingly, our technique can even be used to remotely spy on the temperature in devices that do not contain temperature sensors *at all*. It only requires compromising the software of an IoT device: Kernel access (in order to disable the DRAM refresh operation) is both necessary and sufficient for our method to work. This prerequisite can be met, for example, in an attack scenario where a malicious device driver is provided by an attacker that encapsulates the code of our attack. Installing the kernel driver necessitates kernel access rights which permits the installation of our attack code without the user's knowledge. Once an attacker has gained access to the kernel, he or she can effortlessly control the memory controller and disable its refresh operation.

A. Our Contributions

This article is an extended version of our earlier conference paper [16]. In the original work, it was demonstrated that DRAM decay can be used to measure the ambient temperature only utilizing modified software on IoT devices. An attacker can practically conduct the DRAM decay enrollments at a constant ambient temperature to later map the DRAM decay measurement results to different temperatures and guess the user's behavior or environmental changes. The temperature resolution was shown to be as good as 0.5 °C in commodity, off-the-shelf IoT devices, enabling attackers to measure fine-grained temperature changes around the IoT devices.

To the best of our knowledge, this is the first paper evaluating temperature spying attacks based on DRAM decay with such a high temperature precision. Furthermore, this extended version now contains the following additional contributions:

- Additional measurements were carried out with higher precision and using at a greater temperature range compared with the original paper. Now, each measurement is taken within a stable temperature environment (a temperature chamber) with a deviation of at most 0.1 °C, and over a temperature range from 0 °C to 70 °C in 2.5 °C increments. In the original paper, only measurements from 20 °C to 45 °C were taken.
- A novel temperature estimation function has been developed and subsequently demonstrated in two attacks. This function facilitates the ability to enroll the measurements on a similar device like the one under attack, thereby increasing the practicability of our attack. The absolute ambient temperature in the vicinity of a device can be measured by capturing only a single measurement at a predetermined temperature on the device under attack instead of merely measuring relative temperature changes as in the initial paper.
- Finally, we first present methods to detect and prevent such kinds of attacks in practice. For example, we experimentally prove that putting the device inside a closed box will consequently distort the temperature measurements, disabling the attacks.

B. Related Work

Various related works exist that describe the dependence of the DRAM retention time on ambient temperature. We additionally analyze other temperature-dependent methods that can be utilized on DRAM modules to determine the device's ambient temperature.

A paper discussing sensors based on intrinsic Physical Unclonable Functions (PUFs) was published by Chen et al. [17]. The paper demonstrated the feasibility of measuring the temperature through the DRAM retention effect on Raspberry Pis when disabling the DRAM refresh operation. The authors evaluated the temperature estimation in the range of 15 °C to 40 °C using the bit-flips occurring at a decay time of 60 s. Compared to our work, an accuracy of only ±4 °C could be achieved. Moreover, the authors presented a test setup that modifies device firmware to execute the measurements in a bare-metal program, instead of booting an entire operating system kernel.

Tian et al. [18] published a paper which presents techniques to generate fingerprints on FPGAs to identify them in cloud infrastructures. One contribution of this paper is to monitor the temperature in the vicinity of FPGAs using bit-flips obtained from DRAM decay measurements. This work demonstrates temperature-spying attacks which were conducted on Xilinx Virtex Ultrascale+ Boards equipped with DDR4 memory modules.

Liang et al. [19] further analyzes temperature side-channel attacks executed on FPGAs. The authors exploit the high sensitivity of the signal propagation of Time-Digital Converters (TDCs) to estimate the ambient temperature of FPGAs in cloud systems. These measurements serve as the basis for analyzing the correlation between the temperature in the proximity of the FPGA and the temperature outside the data center, therefore providing insights if cool air from outside is used to cool the data center. Moreover, this new approach is complemented with DRAM retention-based temperature side channels on DDR4 memory yielding a more precise temperature estimation.

Giechaskiel et al. [20] published a paper about information leaks in cloud infrastructures. Here, bit-flips in DRAM memory modules of FPGAs were used to monitor the temperature in the vicinity of FPGAs within data centers. A second paper published by Giechaskiel et al. [21] describes the same attack as in the previous paper in more detail and additionally demonstrates an attack using 24 FPGAs to monitor the temperature over 24 hours.

Another work describing the dependence of the DRAM retention time on the temperature was published by Wang et al. [22]. This paper focuses on computing at cryogenic temperatures, for example, required to implement quantum computers. For this reason, the retention of DRAM memory modules was examined on temperatures from 358 K, 77 K and 263 K.

Orosa et al. [23] utilize row-hammering to measure the ambient temperature in the vicinity of FPGAs equipped with DDR4 memory. Overall, absolute and relative temperature changes could be measured with a precision of less than 2.5 °C, but not better than ±1 °C.

In comparison to the above-mentioned papers, our paper provides a more detailed evaluation of various temperatures, resulting in a precision of up to ±0.5 °C; two different attacks are demonstrated and countermeasures are implemented and proven. Additionally, some of the papers exploit temperature-dependent methods aside from the DRAM retention effect to spy on the device's ambient temperature. Such methods include, calculating the temperature from the amount of bit-flips caused by row-hammering or by exceeding the memory module's write and read latency. However, these methods have various drawbacks that will be analyzed in the following. There are several reasons why we choose DRAM retention among these alternatives. One reason is that our attack is more generic since only the DRAM refresh needs to be disabled. In contrast, DRAM-latency and row-hammering methods depend on the physical structure of the memory. For instance, in the case of row-hammering, the physical structure of the memory has to be reverse-engineered to hammer on neighboring cells. Similarly, DRAM latency measurements differ, depending on which region the requested cells are situated in, requiring further knowledge about the physical structure of the memory. Furthermore, we assume that a higher precision could be obtained using DRAM retention due to the ability to adjust the decay time in a very fine-grained manner.

The DRAM retention effect was also utilized to generate cryptographic keys or to provide authentication in the context of PUFs. Our research leverages the same effect but without relying on PUF properties like uniqueness, uniformity, or randomness. In support of our approach, various PUF papers have been studied, containing valuable information regarding the DRAM retention effect in general.

Anagnostopoulos et al. [24] and Schaller et al. [25] investigated the DRAM retention effect under different temperatures in a PUF context. In these publications, row-hammering PUFs and DRAM-retention PUFs were examined on PandaBoards and Intel Galileo boards. They describe the dependence of the two PUF constructions on the supply voltage and ambient temperature, tested from temperatures ranging from 25 °C to 40 °C [24] and from 40 °C to 80 °C [25].

Müelich et al. [26] present a theoretical model which captures the instabilities of PUF responses. DRAM retention PUFs are evaluated on temperatures from 25 °C to 90 °C.

There are several additional papers related to DRAM retention PUFs, e.g. [14], [15], [24], [25], [27], [28], [29], [30], [31], [32]. For instance, PUFatt [33] demonstrates an ALU PUF-based secure remote attestation scheme for embedded systems. SHAIP [34] is a PUF-based mutual authentication framework with an unlimited number of authentications and a privacy-preserving property. BIST-PUF [35] enables the real-time assessment of PUF's unpredictability and stability in hardware. The emerging trends and challenges of PUFs and robust protocols are discussed in [36] and [37]. Similar to other PUF solutions [38], [39], [40], [41], [42], [43], these PUFs may potentially provide improved resilience against invasive [44] and side-channel attacks [45]. The said DRAM decay, among other things, depends highly on the temperature [13], [14], [15], [24], [32], with an increased temperature accelerating the charge leakage and the decay process.

C. Organization of This Paper

The rest of this article is organized as follows: Section II presents background information on DRAMs and their decay behavior when the refresh operation is disabled. Section III discusses how DRAM retention was implemented. In particular, we describes how the enrollment is done along with an explanation of the temperature estimation process necessary for two presented attacks: The first one spies upon relative temperature changes, and the second one is monitoring absolute temperature and can be utilized on a device not measured beforehand. The subsequent Section IV evaluates the methods introduced in the previous section; it includes an analysis of the measurement results from the enrollment and the attack execution. Attack detection and prevention methods are suggested, implemented, and successfully demonstrated in Section V. Finally, Section VI concludes this work.

II. Background: DRAM Cells and Their Decay Characteristics

DRAM is one of the most widely used memory types in computer devices. In DRAM, each bit of data is stored in a DRAM cell that consists of an access transistor and a capacitor, as shown in Figure 1 (a).

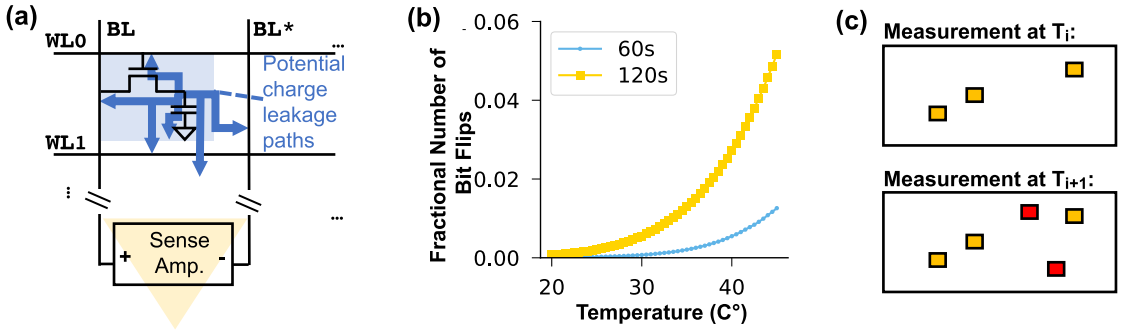


Fig. 1. (a) DRAM cell schematic. (b) Temperature dependency of the fractional bit-flips for DRAM modules from a tested Intel Galileo board. (c) Illustration of a sample DRAM array decay measurement at temperatures T_i and T_{i+1}, both with the same decay time t; the highlighted cells are the cells where a bit-flip occurs – for the same decay time, more cells flip at the higher temperature (illustrated as red cells).

Each capacitor of the DRAM cell has two states, charged and discharged, which are used to store one bit of data. Word Lines (WL) are used to enable the access transistors, and Bit Lines (BL) to read out data. Data from two complementary bit lines (BL and BL*) are amplified through a sense amplifier. The amplifier is used to convert analog voltage levels into either full V_{DD} or 0 V depending on the present voltage on the capacitor.

DRAM is a volatile memory, and the stored data will be lost if refresh or power is turned off. This is because capacitors lose charge over time.Figure 1 (a) shows possible paths through which the charge on each capacitor can leak. To prevent data loss, each DRAM cell requires a periodic refresh. Most memory modules have a refresh period of 64 ms.

When DRAM refresh is disabled, charged cells will steadily lose charge. If a charged cell loses enough charge, it becomes discharged, and the stored bit of data flips. The loss of charge over time is referred to as *DRAM decay*. The time, a cell can keep a bit value without refresh is called the *retention time*. Different DRAM cells have different retention times. Thus, if the refresh operation is disabled for a longer time, more cells' retention times are exceeded, and more bit-flips appear. Meanwhile, if a cell is initialized to a discharged state, its value will never flip as there is no charge to leak.

A DRAM decay measurement is a measurement showing which DRAM cells have flipped in a DRAM region after a given decay time t has elapsed. To perform a DRAM decay measurement, first, a DRAM region is selected and the cells in this region are initialized to a known value. For example, this work uses logical `0` as the initial value of all the cells.[1] The DRAM region is then allowed to decay, i.e., the refresh operation is disabled, for time t. After the elapsed time t, the DRAM region is read, to observe which cells have flipped their initialized logical value of `0` to logical `1`. These cells are the ones that have decayed during time t.

The fractional number of bit-flips (i.e., the number of bit-flips in the DRAM region divided by the size of this region) for an Intel Galileo board [46] is shown in Figure 1 (b). The number of bit-flips increases as the temperature increases. Also, using a decay time of $t = 120$ s will result in more bit-flips than using a shorter decay time of $t = 60$ s, for example. Figure 1 (c) illustrates example DRAM decay results for the same decay time t, but at temperatures T_i and T_{i+1} ($T_{i+1} > T_i$). More bit-flips appear at the higher temperature T_{i+1}, compared to the lower temperature T_i. Also, bit-flips that occur at a lower temperature are a subset of the bit-flips that occur at a higher temperature.

[1]Note that some DRAM cells map logical `0` to the charged state, whilst others map logical `1` to the charged state. The exact mapping is not published by DRAM manufacturers, but we have empirically derived that about half the cells in the tested DRAM modules map logical `0` to the charged state. The cells that map logical `0` to the discharged state simply do not contribute any value to the measurement, but also do not interfere with it.

III. Implementation

This section describes our implementation of the kernel module, which disables the DRAM retention, and keeps the operating system's kernel functioning. In addition, we provide an overview of the enrollment process that facilitates the development of precise estimation functions.

A. Compromising the Kernel and Disabling the DRAM Refresh

Many IoT devices do not have a dedicated temperature sensor – this work shows that even in the absence of a temperature sensor, attackers can still leverage DRAM cells in IoT devices to obtain the ambient temperature. The attacker first needs to compromise the device under attack remotely to be able to control the DRAM refresh. Usually, he or she needs to compromise the kernel (for some devices also the firmware needs to be modified) to measure the DRAM decay. Many IoT devices are vulnerable to exploits that can give kernel privileges.

Our presented attacks are utilized to spy on temperature changes in the vicinity of servers and IoT devices over time. Thus the DRAM decay measurements need to be carried out during system runtime. However, executing DRAM decay measurements while maintaining the system's reliability without additional hardware poses a significant challenge. This is due to the fact that the DRAM refresh operation can only be disabled for the whole DRAM module and not for arbitrary memory regions.

Once the whole DRAM module's refresh is disabled, all memory content will eventually decay, and errors in the memory contents will cause the system to crash. As a solution,

similar to the approach of [14], this work uses a kernel module to disable the refresh of the whole DRAM module while issuing extra memory accesses to the memory regions holding the critical system data. Each DRAM access also acts as a refresh, so the system data that are explicitly accessed will not decay. At the same time, the other cells in the DRAM, which are not accessed, will decay. To obtain a readout at a precise point of time, we modify the firmware of the compromised board in such a way that, after the decay time has elapsed, the reserved memory region is immediately copied to a different region in the RAM and the refresh is reactivated. This avoids additional bit-flips during reading and, thus, avoids distortion of the measurement results.

B. Spying on Relative Temperature Changes

In this section, we describe how we can estimate relative temperature changes in the vicinity of a device by capturing "indicator cells" in certain memory regions. With this method we can spy on temperature changes relative to a reference temperature, which must be known on the attacked device. This method requires an enrollment on the attacked device.

1) Simulating Enrollments Based on the Bit-Flips to Temperature Approximation: Based on measurements of bit-flips at certain constant temperatures, we show how a measurement with a DRAM decay t_{real} and temperature T_{real} can be utilized to simulate the decay t_{sim} at a temperature T_{sim}. This simulation is accomplished through a mathematical model, describing the relation between bit-flips and the temperature, which is further explained in this section.

We denote $\Delta T_{rs} = T_{real} - T_{sim}$, which is the temperature difference between the real temperature (T_{real}), and the temperature that the attacker can measure during enrollment to simulate the real temperature (T_{sim}). As indicated in the works of Xiong et al. [14] and Schaller et al. [25], the DRAM decay time t_{sim} and t_{real}, and temperature ΔT_{rs} have the following relationship:

$$t_{sim} = t_{real} * e^{k\Delta T_{rs}}. \tag{1}$$

Furthermore, we test that identical models of DRAM chips have the same temperature index k, so the attacker can compute k using his or her own device (where he or she can control the temperature), to then use that k for the attack on a remote device. The index k is estimated based on enrollment measurements discussed in Section IV-B.1.

2) Indicator Cells: The execution of temperature-spying attacks requires capturing the bit-flips in a certain memory region, which potentially consumes a high bandwidth by accessing the memory and transmitting the data to an attacker. One method to reduce the required bandwidth, making the attack also less detectable, is the usage of "indicator cells". An ideal indicator cell for a decay time T_i should never flip when being measured with the same decay time T_i and always flip when evaluated with a longer decay time T_{i+1}. The difference of two decay times T_i and T_{i+1}, depends on the desired temperature resolution. When the attacker attempts to derive the temperature from the DRAM decay measurements, he or she will use l indicator cells and perform majority votes. The minimum value of l is 3. With $l = 3$, an error rate of up to 33% can be corrected by the majority vote. With $l = 5$, the majority vote can correct an error rate of 40%, and so forth. In practice, the attacker can choose l based on the noise and the number of available candidate indicator cells in the DRAM region.

The Bit Error Rate (BER) for each temperature T_i is calculated as follows:

- in the enrollment measurement, the number of candidate indicator cells are counted;
- in the spy measurement, if an indicator cell flips at T_i or an indicator cell does not flip at T_{i+1}, this cell is seen as an error;
- the number of errors is counted and divided by the result from step (i) to compute the BER.

The evaluation of the temperature estimation based on indicator cells is evaluated in Section IV-B.1.

3) Enrollment: Deriving the mathematical model describing the relation between the number of bit-flips captured at different ambient temperatures and the corresponding temperature values, requires multiple measurements at different temperatures captured during an enrollment phase. For an attacker, it is typically not feasible to control the ambient temperature of a device and to capture all these enrollment measurements. To mitigate this problem, different temperatures can be simulated by measurements with multiple decay times at a constant temperature. For example, the decay result of decay time $2t$ at enrollment temperature T can be approximated by using the decay result of decay time t at temperature $T+10\,^{\circ}\mathrm{C}$. In this way, the attacker takes measurements at a constant temperature T_0 for decay times $\{t_0, t_1, t_2, \ldots, t_m\}$ to simulate decay results at $\{T_0, T_1, T_2, \ldots, T_m\}$ for decay time t_0 at each of these temperatures. In Section IV-B, we experimentally validate this approach.

4) Attack Phase: During the attack phase, first, the board under attack is compromised by the kernel module described in Section III-A. Afterwards, the enrollment phase is executed, during which indicator cells are detected and the approximation using the mathematical model shown in Equation (1) is derived, which requires to estimate k. The attack requires a dedicated memory region to be reserved to execute DRAM retention measurements. Afterwards, only the indicator cells are measured, and the results are transmitted to the attacker. Depending on the frequency of the temperature measurements and the adjusted decay time, the control over this region can be returned to the operating system kernel between attack executions. In Section IV-B.3 two practical attacks scenarios, one spying on the temperature in the vicinity of a server in a data center, and the second spying on the temperature in a user's home using an IoT device, are demonstrated.

C. Spying on Absolute Temperature

This section presents a second method on how to estimate the correlation between bit-flips and a device's ambient temperature. In contrast to the previously introduced method, this method returns absolute temperature values and enables an

enrollment on a different device than the one used during the attack.

1) Approximating the Temperature-Estimation Function: For the attacks spying on absolute temperatures, we propose the following function to approximate the dependency between the absolute temperature in the vicinity of a device and the number of bit-flips, both for the initial enrollment and during the subsequent attack.

$$T_{apx} = c_1 \cdot e^{c_2 \cdot (\mathsf{bf}^T \cdot p)}. \tag{2}$$

Here, a temperature T_{apx} is approximated, given bf^T, the average number of bit-flips of multiple similar boards with a decay time of t depending on the temperature T. The parameter p is calculated as follows:

$$p = \frac{\mathsf{bf}_{enr}^{T_k}}{\mathsf{bf}_{obs}^{T_k}}, \tag{3}$$

and relates the number of bit flips observed in a particular board under a known temperature T_k, denoted by $\mathsf{bf}_{obs}^{T_k}$, to the $\mathsf{bf}_{enr}^{T_k}$ that has been observed during the enrollment (in the board used for the enrollment) for temperature T, denoted by bf_{enr}^{T}, if $T_K = T$, or the number of bit-flips that is calculated for T_K based on the enrollment measurement for T and Equation (2). Thus, T_k can be any temperature as long as $\mathsf{bf}_{obs}^{T_k}$ and $\mathsf{bf}_{enr}^{T_k}$ correspond to devices of the same model, which have the same constants c_1 and c_2. Obviously, for the measurements of the same board, p is 1. The constants c_1 and c_2 are optimized during the enrollment process that utilises the whole temperature range, such that T_{apx} is close to T, given bf_{enr}^{T}.

To spy on the temperature of a previously unseen device (of a known model for which other devices have been fully enrolled), the number of bit-flips bf_{spy}^{T} captured at an unknown temperature T is used as bf^T in Equation (2). Additionally, the parameter p needs to be calculated using Equation (3), and a single enrollment measurement $\mathsf{bf}_{spy}^{T_k}$ of the attacked device under a known temperature T_k, which acts as the $\mathsf{bf}_{obs}^{T_k}$. This measurement is necessary to capture absolute temperatures in the vicinity of a device.

In this way, for example, enrollment measurements of multiple boards can be used to approximate the temperature based on the number of bit-flips, independent of the Galileo board that is used to spy on the temperature. Better results can be achieved at higher temperatures due to a higher number and stability of bit-flips.

The need for only a single enrollment measurement allows us to execute the attack much more efficiently because a smaller amount of data needs to be transmitted compared to an enrollment that utilises the whole temperature range. Additionally, the extrapolation using p allows a precise temperature approximation on devices with a high deviation in the absolute number of bit-flips, because the dependency between the bit-flips and the temperature will follow the same function in similar devices, notwithstanding the absolute number of bit-flips. Moreover, the extrapolation using p, and the fact that devices of the same type demonstrate the same dependency between the number of bit-flips and the temperature, allow for the use of much smaller memory regions, compared to the enrollment measurement that uses the whole temperature range, to spy on the ambient temperature.

2) Enrollment: To execute the attack, m enrollment measurements at a fixed decay time t and m different temperatures $\{T_0, T_1, T_2, \ldots, T_m\}$, covering the temperature range of interest, must be taken. It is not necessary to capture these measurements on the target device, which would require controlling its ambient temperature. The measurements can rather be obtained from an additional device that the attacker possesses. Afterwards, the relation between the bit-flips and the ambient temperature can be derived, by estimating the constants c_0 and c_1 of the mathematical model, presented in Equation (2). If it is necessary to obtain absolute temperatures, a single measurement at a known temperature is required on the target device.

To achieve precise evaluation results and thus to be able to create an accurate mathematical model to estimate the ambient temperature based on the captured number of bit-flips, we propose to perform the enrollment in a climate chamber that enables precise temperature control. A measurement server starts the tests on the boards and simultaneously monitors and controls the temperature of the climate chamber, which requires some time until the temperature stabilizes. That is the reason why after reaching the target temperature, the test program on the measurement server checks for 90 s if the temperature is within the acceptance range, which is in our case $\pm 0.1\,^{\circ}\mathrm{C}$. Only if this requirement is met, the experiments are started.

3) Attack Phase: Similar to the attack phase described in Section III-B.4, we require again a compromised kernel with our kernel module being installed. Compared to the previously explained attack scenario, the number of decayed cells in a reserved memory region is counted on the compromised device, and only the number of bit-flips is transmitted to the attacker. The attacker knows the mathematical model based on Equation (2). The constants c_0 and c_1 are derived from the enrollment measurements on a similar device. The factor p is retrieved from Equation (3), using the number of bit-flips $\mathsf{bf}_{enr}^{T_k}$ measured at a known temperature T_k from the attacked device and $\mathsf{bf}_{obs}^{T_k}$, the amount of bit-flips captured on the device used during the enrollment at the same temperature T_k. This allows the attacker to estimate the ambient temperature based on the transmitted number of bit-flips. In Section IV-C.3, two different attack scenarios are demonstrated in which bit-flips are transmitted by an Ethernet connection to the remote attacker.

IV. Evaluation

In our evaluation, first, we describe the setup of our experiments and how the tests are executed. Afterwards both attacks, one based on indicator cells and the second one based on simply counting the number of bit-flips are evaluated separately. In the end, the overall attack complexity of these two attacks is evaluated.

During the evaluation of our attacks, we adhere to the following assumptions: Overall, our attack targets devices

that are highly integrated, such as System-on-Chips (SoCs) utilized in the IoT domain or servers employed in data centers. Consequently, components like the memory controller are assumed to be highly integrated and thus not accessible to physical manipulation. Accordingly, we assume that, for example, the supply voltage of the memory controller is stable and there are no significant variations during the execution of our attack. Furthermore, we evaluate multiple devices in multiple temperatures without considering potential aging effects.

A. Hardware Setup

The execution of the attacks based on the DRAM retention effect are performed on Intel Galileo Gen 2 [46] IoT development boards, which are equipped with an Intel Quark X1000 SoC and with two 128MiB DDR3-SDRAM modules from Micron. On these boards, the DRAM decay measurements can be executed by loading a kernel module to measure the DRAM decay in the chosen DRAM region during operation, as described in Section III-A. In total, four Intel Galileo boards are measured. To allow for an accurate evaluation of the temperature-dependent characteristics of DRAM modules, a TestEquity 1007C [47] and a Weisstechnik LabEvent thermal chamber [48] are used to control the ambient temperature.

B. Evaluating Attacks to Spy on Relative Temperature Changes

In this section, the relation of bit-flips to temperature, and the enrollment measurements are evaluated. Furthermore, we demonstrate the feasibility of two practical attack scenarios spying on relative temperature changes.

1) Evaluating the Temperature-Simulation Function: To approximate the missing constants of Equation (1), the simulation measurements are taken at $T_{sim} = 25\,°\text{C}$ and $T_{sim} = 30\,°\text{C}$ to simulate the DRAM decay at the temperature ranges T_{real} =20 °C to 40 °C and T_{real} =25 °C to 45 °C, respectively. The measurements are designed to simulate the decay time of both 60 s and 120 s. To simulate decay time $t_{real} = 60\,\text{s}$ (120 s), ten different decay times in the range of $t_{sim} = 45\,\text{s}$ to 160 s (90 s to 320 s) are measured. To estimate the temperature index k, the simulation measurements and the real measurements are compared. For each simulation measurement, we find the real measurement that has the most similar number of bit-flips, and record the pair t_{sim} and $\Delta T'_{rs}$. With the pairs of t_{sim} and $\Delta T'_{rs}$ across $t_{real} = 60\,\text{s}$ or 120 s, T_{sim} =25 °C or 30 °C, the relevant T_{real} ranges, and four Intel Galileo boards, the best-fit temperature index of $k = 0.07$ can be computed.

To show that the simulation measurements are similar to the real measurements, using k, we compute the ΔT_{rs} for each t_{sim}. We then compare each pair of real measurements (t_{real}, T_{real}) and simulation measurements (t_{sim}, T_{sim}). To compare the two measurements, we use the *Jaccard Index*, which is a well-established metric to compare the similarity of two different data sets and thus very suitable for this use case [49]. Let $\mathcal{R}$ and $\mathcal{S}$ denote the set of bit-flips in the real measurement and the simulation measurement, respectively. The Jaccard Index is calculated by $J = \frac{|\mathcal{S} \cap \mathcal{R}|}{|\mathcal{S} \cup \mathcal{R}|}$. If the resulting J is close to 1, it indicates high similarity between the two

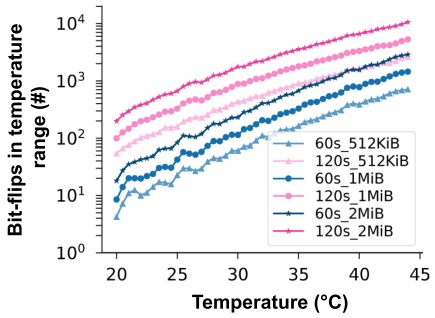


Fig. 2. Number of bit-flips in the temperature range $[T, T+1°\text{C}]$ versus temperature T for different DRAM region sizes.

measurements. Figure 8 shows the distribution of the Jaccard Index for $t_{real} = 60\,\text{s}$ or 120 s for all four Intel Galileo boards. Each box contains ten different simulation decay times and the corresponding real temperatures. The stars indicate the average of and the orange bars indicate the median of the Jaccard Index. The Jaccard Index is higher than 0.85 in almost every case, indicating that the simulation measurements and the real measurements are very similar. Thus, the enrollments can be taken at a fixed temperature to cover a range of different temperatures.

2) Enrollment: First, DRAM decay measurements are performed at $\mathcal{T} = \{20, 21, \ldots, 45\}°\text{C}$, where $dT = T_{i+1} - T_i$ denotes the step between temperature points in $\mathcal{T}$, thus here $dT = 1°\text{C}$. Figure 2 shows the number of candidate indicator cells, i.e., bit-flips at temperature T_{i+1} but not T_i. The results are the average of DRAM regions on four Galileo boards. Two decay times of $t = 60\,\text{s}$ and $t = 120\,\text{s}$ are tested, with DRAM region sizes of 512KiB, 1MiB, and 2MiB.

To evaluate the reliability, at each temperature five measurements are taken for each of the four Galileo boards. The first measurement is used as enrollment, and the other four measurements are used for testing the reliability. Figure 3 and 4 show the average and maximum BER of the spy measurements.

The number of available candidate indicator cells depends on the DRAM region size and the decay time t. In a 512KiB DRAM region, the smallest number of candidate indicator cells occurs when the attacker measures at 20 °C – one of the tested boards gives only 2 indicator cells. Figure Figure 3 and 4 show the average and maximum BER across the four boards and four spy measurements. As shown in Figure 3 and 4, a longer decay time, a larger DRAM region size, or a higher temperature will result in a smaller BER. This is because a longer decay time, a larger DRAM region, and a higher temperature, each yields a higher number of and more reliable indicator cells. The average BER is much smaller than the maximum case, meaning that there are only a few cases where the BER is high. As shown in Figure 4, to obtain reliable results for $l = 3$, at least, 1MiB DRAM region size is needed to achieve a BER of less than 33%. For $l = 5$ or more, a memory region of 1 MB is sufficient to achieve a BER of less than to 40%.

We further explore different temperature resolutions, where $dT = 0.5°\text{C}, 1°\text{C}, 2°\text{C}$ separately. The results in Figures 5 to 7

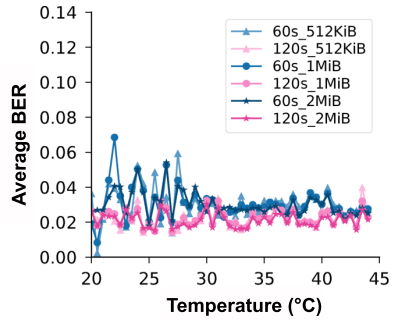


Fig. 3. Average BER in the temperature range $[T, T+1°C]$ versus temperature T for different DRAM region sizes.

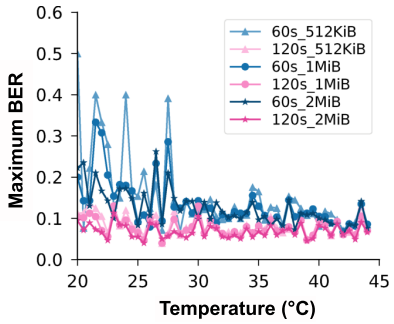


Fig. 4. Maximum BER in the temperature range $[T, T+1°C]$ versus temperature T for different DRAM region sizes.

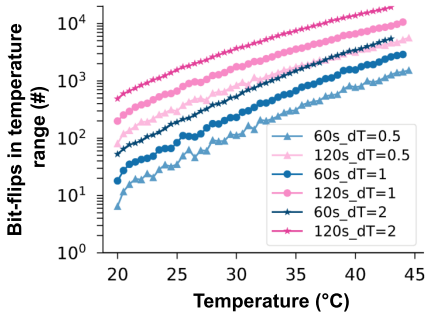


Fig. 5. Number of bit-flips in the temperature range $[T, T+dT]$ versus temperature T for different dT values in a 2MiB DRAM region.

are retrieved from all four Galileo boards, with a DRAM region of 2MiB and a decay time of either 60 s or 120 s. The data are processed in the same way as in Figures 2 to 4. As shown in Figure 5, at least 5 indicator cells can be found in 2MiB regions in the temperature range $[T_i, T_i + dT]$ for all T_i and dT considered.

The average BER, shown in Figure 6, is again much smaller than the maximum case, however Figure 7 shows that when $t = 120$ s and $dT = 0.5\,°C$, the maximum BER can still be corrected by the majority vote of $l = 3$ cells. Nevertheless, with a decay time $t = 60$ s and $dT = 0.5\,°C$, the maximum BER is too large to be corrected by the majority vote of $l = 3$ cells and, thus, a larger DRAM region or longer decay time should be used.

3) Attack Evaluation: This section presents the evaluation of the attacks already described in Section III-B.4. Specifically, two attacks are presented. The first attack monitors the temperature in a room, based on evaluating indicator cells using Intel Galileo boards. A second attack involves monitoring

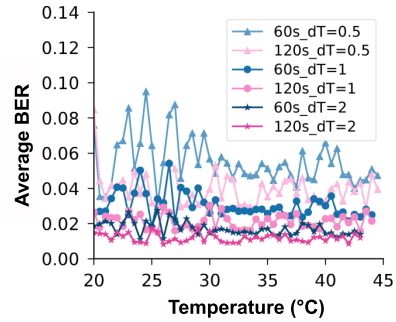


Fig. 6. Average BER in the temperature range $[T, T+dT]$ versus temperature T for different dT values in a 2MiB DRAM region.

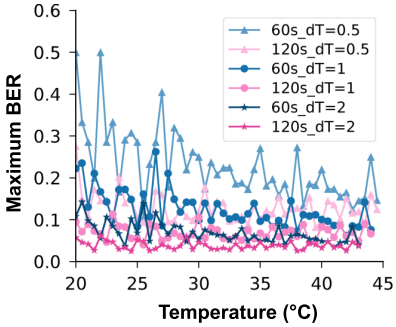


Fig. 7. Maximum BER in the temperature range $[T, T+dT]$ versus temperature T for different dT values in a 2MiB DRAM region.

the workload in a data center by spying the temperatures in a server rack. To show the practicability of the attacks, we deployed several Intel Galileo boards openly on a table in two rooms and in a server rack. A bare Yocto Linux kernel was used during the test. We measured the DRAM decay of 60 s with 2MiB DRAM region every 5 min to infer the ambient temperature. For each of the boards, enrollments with decay times ranging from 50 s to 75 s are taken. According to Equation (1) with $k = 0.07$, the enrollments can simulate the ambient temperature change of $[-3\,°C, 3\,°C]$. Note that, in total, 26 measurements are taken for the temperature range, so the actual temperature resolution is higher than 0.5 °C. A thermocouple is used to get the actual temperature readings during the reference measurements. Indicator cells are generated based on the enrollment measurements and later used to map the decay results to temperatures. More than twenty candidate indicator cells are found in 2MiB for each temperature, so $l = 21$ is used.

In Figures 9 (a) and (b), the temperature in two different rooms measured by the DRAM and a thermocouple for 24 hours is shown. The temperatures measured by the DRAM match the results of the thermocouple. As shown in the figure, during the night, the temperature is stable; while, during the day, due to human activities, the temperature fluctuates in both rooms. Thus, if the attacker can monitor the temperature, this puts the victim's privacy at risk.

In the second experiment, we deployed two Galileo boards in a server rack. Figure 9 (c) shows the temperature measured by two DRAM modules, DRAM1, and DRAM2. DRAM1 is located closer to the fans, with the thermocouple being placed near DRAM1. The arrows indicate when the server starts to

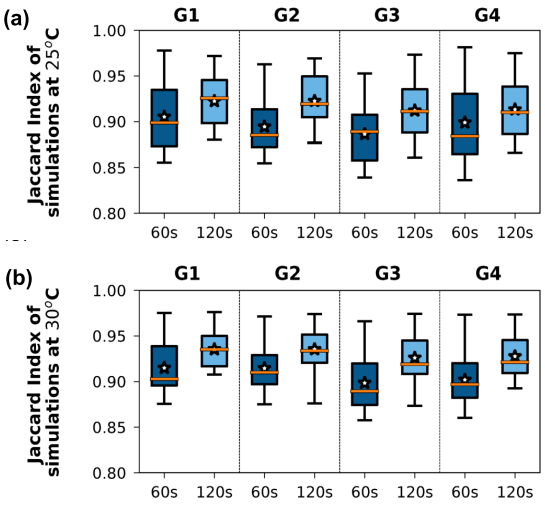


Fig. 8. Jaccard Index between simulation measurements at (a) 25 °C or (b) 30 °C and real measurements with a decay time of 60 s or 120 s.

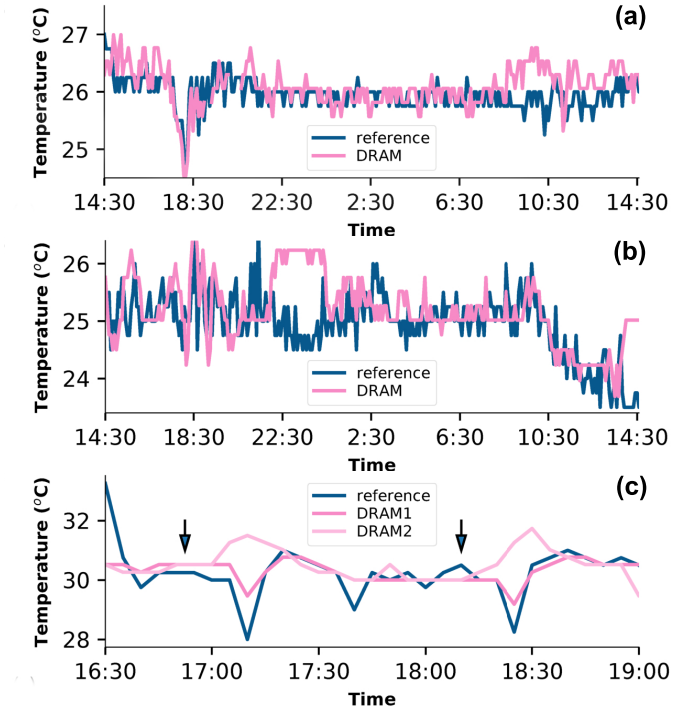


Fig. 9. (a), (b) Results of measuring the temperature every 5 min with DRAM module in two different rooms for 24 hours. (c) Results of measuring the temperature every 5 minutes with DRAM modules in a server rack. The arrows show when the server starts to run a job.

run a job for 25 min. When the server runs, it will gradually heat up DRAM2. Subsequently, the fan starts working, and ambient temperature drops (especially for DRAM1 and the thermocouple). Consequently, using only the IoT device's DRAM, the attacker can monitor the temperature change of the server, which could create a side-channel to reveal the activity of the tenants [12].

C. Evaluating Attacks to Spying on Absolute Temperatures

This section evaluates the attacks spying on absolute temperature values solely using the number of bit-flips. Additionally, the corresponding temperature approximation function is evaluated.

1) Evaluating the Temperature to Bit-Flips Approximation Function: Multiple measurements captured covering the temperature range of interest are used to estimate the constants c_1 and c_2 of our mathematical model, described by Equation (2). For this purpose, we use the enrollment measurements described in Section IV-C.2. To get a more precise approximation, the constants c_1 and c_2 are calculated separately for the temperature regions from 0 °C to 25 °C, from 25 °C to 45 °C, as well as from 45 °C to 70 °C. Additionally, we use measurements with t =240 s for the approximation, as this decay period yields a substantial number of bit-flips on all temperature regions. In Figure 10 (e), the real temperature is shown compared to the approximation function. There, the temperature region is subdivided into the three aforementioned intervals, and the approximation is calculated for each interval. For higher temperature values, the function T_{apx} approximates the temperature very precisely. The approximation of values close to zero is more imprecise due to the lower number of bit-flips.

2) Enrollment: To evaluate attacks based on the absolute amount of bit-flips, additional measurements at $\mathcal{T}_{new} = \{0, 2.5, \ldots, 70\}$°C are taken, to examine especially lower and higher temperatures. Here, we notice that a larger step size of $dT_{new} = 2.5$ °C is sufficient to approximate the temperatures between the 2.5 °C steps. For this attack, ten measurements are captured per temperature and decay time, and the average number of bit-flips of these measurements is used to approximate the temperature. This simple method can spy on absolute temperatures and can additionally detect small temperature changes with comparable precision to the method described in Section IV-B.1. We evaluated further decay times of $t = 60$ s, $t = 120$ s, $t = 180$ s and $t = 240$ s, and a memory region size of 1 MB. There, the total amount of bit-flips is measured per temperature and decay time, resulting in multiple measurements:

$$Exp^{240s} := \mathrm{bf}_{enr}^{0\,^\circ\mathrm{C}}, \mathrm{bf}_{enr}^{2.5\,^\circ\mathrm{C}}, \ldots, \mathrm{bf}_{enr}^{70\,^\circ\mathrm{C}}\}.$$

A visualization of these measurements is shown in Figure 10 (a) to (d).

3) Attack Evaluation: As for the attacks based on relative temperature changes, we also consider two different attack scenarios using the approximation function described in Section III-C.1. The first scenario concerns spying on the temperature next to a server to gather information about its workload. The second scenario spies the temperature within a room, e.g., in the context of a smart home, to find out if somebody is present in the room.

These attacks are implemented using two Galileo boards. The first device does not need to be physically in the area being spied on. Enrollment measurements on this device are taken over the whole temperature range. These measurements are then used to approximate the dependency of the number of bit-flips on the temperature using Equation (2), as already explained in Section IV-C.2. To implement the attacks, the approximation calculated by the measurements of the first device is used, as well as a single enrollment measurement of

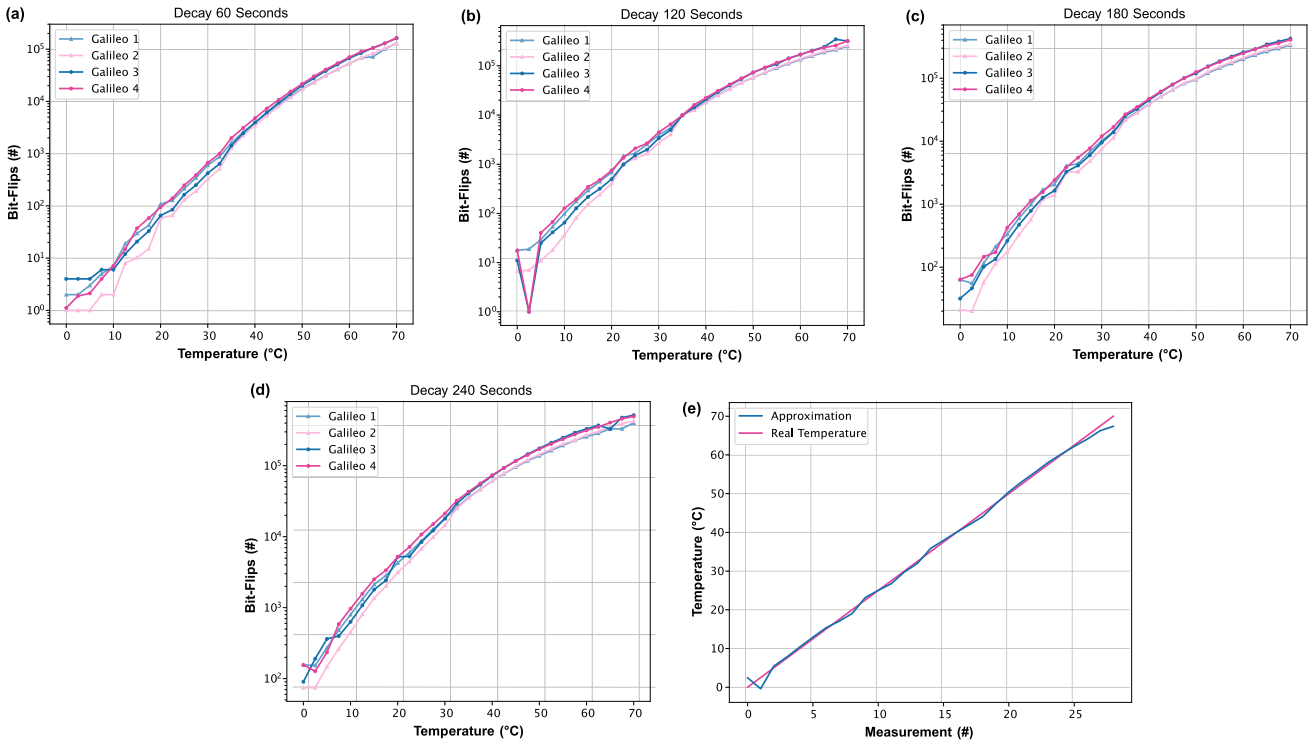


Fig. 10. (a) to (d) show the dependency between the temperature and the number of bit-flips of the four decay times 60 s, 120 s, 180 s and 240 s. Each curve shows the average amount of bit-flips of a specific device over ten measurements on the same device. (e) shows the approximation of the temperature given the average bit-flips per temperature of four Galileo boards, based on a decay time of 240 s.

the second Galileo board. This allows an attacker to execute temperature spying attacks on a board without capturing enrollment measurements over the whole temperature range using the spying board itself.

In the proof-of-concept implementations of both attack scenarios, the spying Intel Galileo board is connected to a local area network via an Ethernet cable. A second malicious device, in our case a Raspberry Pi, is located in a different room, but is connected to the same network. On the Galileo board, malicious software is installed, which continuously executes a DRAM decay measurement during runtime on a 256KiB (= 32KiB) memory region. Before the execution of the measurement, a 256KiB memory area of the Galileo board is filled with ones. Afterwards, the memory refresh is disabled, and for each 120 s, the number of bit-flips is collected and sent to the Raspberry Pi. After each measurement, a new 120 s DRAM decay measurement is executed. In the beginning, only one enrollment measurement is done at 40 °C. Subsequently, the factor p is calculated according to Equation (3). p is stored on the Raspberry Pi and used for each temperature approximation using Equation (2). This allows the Raspberry Pi to calculate the temperature in the vicinity of the Galileo board every 128 s when also considering the time to initialize and to read from the memory.

For both scenarios, the ambient temperature was regulated by a Weisstechnik LabEvent climate chamber, which was also controlled by a Raspberry Pi. To test the robustness of the temperature prediction, faster and slower temperature changes were tried. It was also tested how the temperature prediction works on minor temperature variations. The temperature curve that was obtained, can be seen in Figure 11 (a). Here, the approximation using the Galileo board almost follows the real temperature curve. We can see that this approach has a small delay, probably caused by the time needed for the DRAM modules to be affected by the ambient temperature changes. In Figure 11 (b), the deviation from the real temperature can be seen. Here, the deviation is below 1 °C most of the time. Only rapid temperature changes cause greater deviations.

Afterwards, the same attacks are evaluated on the device previously used for enrollment over the whole temperature range. As expected, a better approximation of the temperature based on the current DRAM bit-flips of this board can be achieved, in comparison to the execution on the other device, for which an enrollment over the whole temperature range has not been performed. As shown in Figure 12, the deviation of the approximated and the real temperature is almost always below 0.5 °C.

D. Attack Complexity

The attacker's efforts consist of making enrollment measurements and conducting each temperature readout during the actual attack. For both, the attacker needs to be able to run malicious kernel code on the victim platform to control the DRAM refresh.

The enrollment time consists of the measurement time, the data transfer time, and the time to identify indicator cells. The measurement time is the decay time plus the time to initialize (write) and read from the DRAM region. For a 2MiB memory region, on an Intel Galileo, it takes about half a minute to

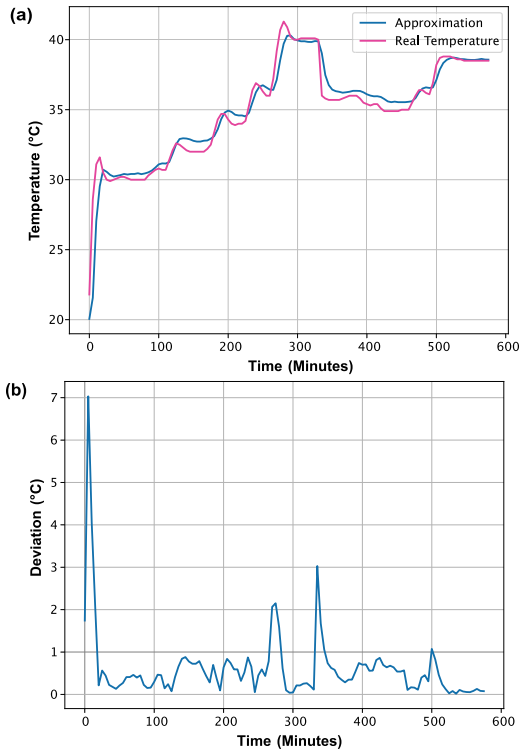


Fig. 11. This figure shows a spying attack on the temperature nearby a server system, simulated by a climate chamber. (a) shows a comparison of the real and the approximated temperature on a Galileo board for which only one enrollment measurement has been taken, and Equation (2) is used. (b) shows the deviation from the real temperature, which is below 1°C most of the time.

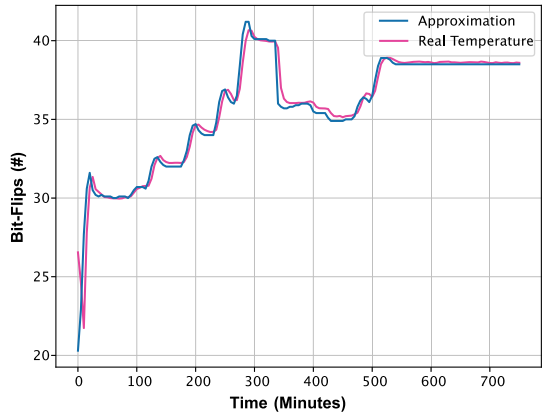


Fig. 12. Simulation of a temperature spying attack in the vicinity of a server. In this case, the approximation is based on the device for which an enrollment over the whole temperature range has been performed. As expected, an approximation of the temperature with higher precision is achieved.

read or write the region. Thus, one enrollment takes about two minutes, considering a decay time of $t = 60$ s: half a minute to initialize, one minute to allow decay to happen, and half a minute to read the DRAM region to locate the decayed bits is required. By using smaller memory regions of 256KiB for the attacks described in Section IV-C.3, the additional delay caused by the initialization and the read operations is only 8 s.

The total number of measurements depends on the temperature range and the required temperature resolution. To acquire ten enrollment measurements, assuming an average enrollment decay time of $t = 60$ s, it takes less than half an hour. The data transfer time depends on the size of the data to transfer, and the network speed achieved each time. The time to compare the enrollment measurement and identify the indicator cells is negligible.

The temperature readout time consists of a single measurement. Furthermore, because only the indicator cells need to be measured, the time to initialize and read the result is negligible compared to the decay time.

It is important to note that the continuous refresh operation of the kernel memory causes a significant CPU load. This effect has been previously analyzed in a work by Xiong et al. [14]. The authors observed that the refresh operation of a 128 MB memory segment, holding the kernel's memory of the Galileo Boards, results in 33% CPU load, when executing a refresh operation every 64 ms. However, it should be noted that this refresh rate is highly conservative and longer refresh intervals can be chosen without risking bit-flips in the kernel's memory. Consequently, a refresh rate of 200 ms produces only 10% additional CPU load. Moreover, the control over the memory region can be returned to the kernel in the absence of an attack. Furthermore, the previously disabled DRAM refresh operation of the memory controller can be reactivated when no attack is executed. By increasing the interval of the refresh operation and by activating the memory controller's DRAM refresh when no attack is executed, the likelihood of our attack being detected can be reduced. This method reaches its limits with larger memory regions of multiple gigabytes, resulting in significant CPU loads.

V. Countermeasures

This section outlines different countermeasures, mitigating the impact of temperature-spying attacks. Specifically, we differentiate between passive attack detection and active attack prevention.

A. Attack Detection

There are various methods to detect DRAM-spying attacks.

1) Monitoring the CPU Time: One approach relies on the fact that the DRAM refresh can only be disabled for an entire DRAM module. In order to maintain the data integrity of memory regions like those used by the operating system's kernel, a periodic refresh operation is mandatory, typically every 64 ms. In case the refresh is not performed by the memory controller, the CPU needs to initiate read access to memory regions not reserved for our temperature-spying attack. This causes an increased, and therefore detectable CPU load.

2) Detection Based on Memory Accesses: Another detection method involves monitoring the repeated pattern that is written regularly to certain memory regions. This could be detected, for example, by a process monitoring different memory regions.

3) Monitoring the Network Traffic: In a third detection method, the network traffic can be observed to detect suspicious data frames containing the number of bit-flips, sent to an attacker regularly.

All these detection measures enable active countermeasures, such as restricting the network traffic, disallowing the access to specific memory regions, detection and disabling of processes with high CPU load, to prevent or mitigate the impact of such attacks.

B. Attack Prevention

In contrast to the attack-detection, attack-prevention measures try to proactively prevent the system from such attacks beforehand.

1) Protection of the Kernel and the Firmware Code: One way to mitigate the said temperature spying is to prevent disabling the DRAM refresh, as the attacker needs to disable the DRAM refresh to measure the DRAM decay. Since disabling the DRAM refresh can only be achieved in the kernel of the operating system and/or the relevant firmware, on almost all platforms, the attacker has to inject untrusted code into the kernel or firmware. Thus, one simple countermeasure is to protect the kernel and firmware code.

However, forcing the DRAM refresh to be always on is not desired from an energy-saving perspective. To this end, a deep sleep mode usually exists, where the DRAM refresh is off. Therefore, an attacker can write initial values into the DRAM region and force the DRAM into that sleep mode, such that the memory decays. To prevent this attack, the system needs to always zero out the whole memory immediately when the memory returns from the sleep mode.

In general, although the implementation of this countermeasure would seem possible, it would also cause a certain level of inconvenience and also potentially require the addition of extra resources to the system.

2) Impact of Using a Cover: We also examined the effect of placing the Galileo boards/DRAMs inside a closed box made of PLA (Polylactide). As illustrated in Figure 13 (a), more bit-flips occur when the Galileo board is inside the box in comparison to conducting the same experiment without a cover. The observed difference in the number of bit-flips with and without the cover even increases with rising temperature. For comparison, also the behavior when executing the experiment with constant temperature was examined. In Figure 13 (b), the number of bit-flips with and without the cover is given in 25 measurements at a constant temperature of 40 °C. Again, the box causes a much higher amount of bit-flips compared to the execution without the cover. Because the cover does not only add a constant offset to the number of bit-flips but also distorts the slope of the function, this mechanism can be used to mitigate temperature spying attacks. We expect that the higher amount of bit-flips may be caused by the heat produced by the Galileo boards and the missing air circulation. The missing air circulation can cause the risk of overheating, which is especially critical for actively cooled devices requiring fresh air floating to a fan. For passively cooled devices like our Galileo boards, experimenting with different materials, like aluminum could probably increase the cooling capabilities of the device. Different materials, or varying production parameters, like the thickness of the material, could aggravate the attack even more. Overall, we note that a simple box is sufficient to effectively distort our attacks.

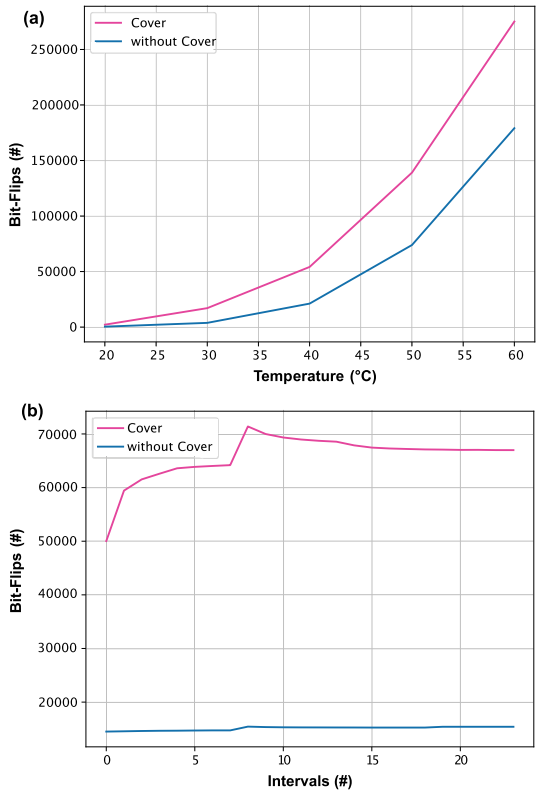


Fig. 13. (a) comparison of the number of bit-flips when measuring the boards within a cover box and without it at temperatures from 20 °C to 60 °C. Here, the cover causes a higher amount of bit-flips in comparison to the measurement without the cover. (b) Execution of 25 measurements with and without the cover at a constant temperature of 40 °C. The cover causes a much higher amount of bit-flips, compared to the bit-flips caused without it.

VI. Conclusion and Further Work

This work demonstrated that commercial, off-the-shelf DRAM modules can be abused to act as remote temperature-spying sensors in ordinary IoT devices. We showed that attackers only need to modify the software of a device and take enrollment measurements at a constant temperature. Subsequently, they can monitor the ambient temperature over a large temperature interval by measuring the DRAM decay while the DRAM refresh operation is disabled. We proved in experiments that this approach can achieve a very high temperature resolution of 0.5 °C in practice.

Moreover, we analyzed various papers exploiting the DRAM retention effect to measure temperatures across various device types, such as small boards like Raspberry Pis, or FPGAs using DDR4 memory. This indicates that temperature-spying attacks based on DRAM retention, as presented in this paper, can be generalized to different types of devices, given that the decay behavior of these memory modules is understood.

In addition, this work for the first time suggested and tested different detection and prevention methods. The most obvious of these consists of enforcing the DRAM refresh to be turned on permanently. However, this is not desirable from an energy consumption perspective. Another measure is shielding the device against the environment by using a

box or a similar encapsulation mechanism. While this can be laborious in practice, it does work effectively, as demonstrated by experiments in this paper. Future research will have to investigate in detail whether there could be other, perhaps yet more effective, countermeasures at the circuit or architectural level. The employed attack and analysis code is available under the GPLv3 license at http://caslab.csl.yale.edu/code/tempspy/.

Although the DRAM-spying attack has been intensively evaluated, two different attacks have been demonstrated, and several countermeasures were presented, there are still areas requiring further investigation. Despite the ambient temperature, there are further environmental conditions that potentially influence our attack and are not considered in this paper. Especially the effect of aging of DRAM memory modules could potentially influence our attack implementation. Although the observation by Bepary et al. [50] observe that these aging effects do not significantly influence the DRAM retention effect, it may be necessary to perform further enrollments after a certain period of time to ensure the same temperature precision as previously achieved. Another possibility requiring further investigation is to consider the aging effects in our mathematical approximation functions, which we leave for future work.

Our attack once more reminds us that the espionage potential and indirect sensor capacities of electronic IoT devices are currently not well-understood. Seemingly simple components with limited functionalities often can be abused in unforeseen manners; this applies to the recent gyroscope attacks [5] as well as to our novel espionage usage of DRAM cells. In addition, the interplay of several ostensibly trivial system components can often create unforeseen emergent behavior exploitable by attackers. This calls for new, perhaps yet more fundamental research to better protect our security and privacy in relevant environments like the IoT. The first steps towards applying highly developed cryptographic tools like the universal composition framework to complex hardware settings have been made only recently [51].

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force. The authors want to thank all reviewers for their constructive feedback.

REFERENCES

[1] R. Taylor, D. Baron, and D. Schmidt, "The world in 2025—Predictions for the next ten years," in *Proc. 10th Int. Microsyst., Packag., Assem. Circuits Technol. Conf. (IMPACT)*, Oct. 2015, pp. 192–195, doi: 10.1109/IMPACT.2015.7365193.

[2] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices," in *Proc. 9th Eur. Conf. Comput. Syst.*, Apr. 2014, pp. 1–14, doi: 10.1145/2592798.2592824.

[3] F. Kohnhäuser, A. Schaller, and S. Katzenbeisser, "PUF-based software protection for low-end embedded devices," in *Trust and Trustworthy Computing*, M. Conti, M. Schunter, and I. Askoxylakis, Eds. Cham, Switzerland: Springer, 2015, pp. 3–21, doi: 10.1007/978-3-319-22846-4_1.

[4] M. Potkonjak, S. Meguerdichian, and J. L. Wong, "Trusted sensors and remote sensing," in *Proc. SENSORS*, Nov. 2010, pp. 1104–1107, doi: 10.1109/ICSENS.2010.5690721.

[5] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing speech from gyroscope signals," in *Proc. 23rd USENIX Secur. Symp. (USENIX Secur.)* Berkeley, CA, USA: USENIX Association, 2014, pp. 1053–1067. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/michalevsky

[6] N. Matyunin, J. Szefer, and S. Katzenbeisser, "Zero-permission acoustic cross-device tracking," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Apr. 2018, pp. 25–32, doi: 10.1109/HST.2018.8383887.

[7] N. Matyunin, Y. Wang, and S. Katzenbeisser, "Vibrational covert channels using low-frequency acoustic signals," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, Jul. 2019, pp. 31–36, doi: 10.1145/3335203.3335712.

[8] N. Matyunin, Y. Wang, T. Arul, K. Kullmann, J. Szefer, and S. Katzenbeisser, "MagneticSpy: Exploiting magnetometer in mobile devices for website and application fingerprinting," in *Proc. 18th ACM Workshop Privacy Electron. Soc.*, Nov. 2019, pp. 135–149, doi: 10.1145/3338498.3358650.

[9] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser, "Covert channels using mobile device's magnetic field sensors," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 525–532, doi: 10.1109/ASPDAC.2016.7428065.

[10] N. Matyunin et al., "Tracking private browsing sessions using CPU-based covert channels," in *Proc. 11th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, Jun. 2018, pp. 63–74, doi: 10.1145/3212480.3212489.

[11] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "PowerSpy: Location tracking using mobile device power analysis," in *Proc. 24th USENIX Secur. Symp. (USENIX Secur.)* Berkeley, CA, USA: USENIX Association, 2015, pp. 785–800. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/michalevsky

[12] M. A. Islam, S. Ren, and A. Wierman, "Exploiting a thermal side channel for power attacks in multi-tenant data centers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1079–1094, doi: 10.1145/3133956.3133994.

[13] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, Jun. 2013, pp. 60–71, doi: 10.1145/2485922.2485928.

[14] W. Xiong et al., "Run-time accessible DRAM PUFs in commodity devices," in *Cryptographic Hardware and Embedded Systems—CHES*, B. Gierlichs and A. Y. Poschmann, Eds. Cham, Switzerland: Springer, 2016, pp. 432–453, doi: 10.1007/978-3-662-53140-2_21.

[15] A. Schaller et al., "Intrinsic Rowhammer PUFs: Leveraging the Rowhammer effect for improved security," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2017, pp. 1–7, doi: 10.1109/HST.2017.7951729.

[16] W. Xiong, N. A. Anagnostopoulos, A. Schaller, S. Katzenbeisser, and J. Szefer, "Spying on temperature using DRAM," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 13–18, doi: 10.23919/DATE.2019.8714882.

[17] S. Chen, B. Li, and Y. Cao, "Intrinsic physical unclonable function (PUF) sensors in commodity devices," *Sensors*, vol. 19, no. 11, p. 2428, May 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/11/2428

[18] S. Tian, W. Xiong, I. Giechaskiel, K. Rasmussen, and J. Szefer, "Fingerprinting cloud FPGA infrastructures," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2020, pp. 58–64, doi: 10.1145/3373087.3375322.

[19] Y. Liang, X. Gao, K. Sun, W. Xiong, and H. Wang, "An investigation on data center cooling systems using FPGA-based temperature side channels," in *Proc. 41st Int. Symp. Reliable Distrib. Syst. (SRDS)*, Sep. 2022, pp. 46–57.

[20] I. Giechaskiel, S. Tian, and J. Szefer, "Cross-VM information leaks in FPGA-accelerated cloud environments," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Dec. 2021, pp. 91–101, doi: 10.1109/HOST49136.2021.9702277.

[21] I. Giechaskiel, S. Tian, and J. Szefer, "Cross-VM covert- and side-channel attacks in cloud FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 16, no. 1, pp. 1–29, Mar. 2023, doi: 10.1145/3534972.

[22] F. Wang, T. Vogelsang, B. Haukness, and S. C. Magee, "DRAM retention at cryogenic temperatures," in *Proc. IEEE Int. Memory Workshop (IMW)*, May 2018, pp. 1–4, doi: 10.1109/IMW.2018.8388826.

[23] L. Orosa et al., "SpyHammer: Using RowHammer to remotely spy on temperature," 2022, *arXiv:2210.04084*.

[24] N. A. Anagnostopoulos et al., "Securing IoT devices using robust DRAM PUFs," in *Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS)*, Oct. 2018, pp. 1–5, doi: 10.1109/GIIS.2018.8635789.

[25] A. Schaller et al., "Decay-based DRAM PUFs in commodity devices," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 3, pp. 462–475, May/Jun. 2018, doi: 10.1109/TDSC.2018.2822298.

[26] S. Muelich et al., "Channel models for physical unclonable functions based on DRAM retention measurements," in *Proc. 16th Int. Symp. Problems Redundancy Inf. Control Syst. (REDUNDANCY)*, Oct. 2019, pp. 149–154, doi: 10.1109/REDUNDANCY48165.2019.9003355.

[27] S. Rosenblatt et al., "Field tolerant dynamic intrinsic chip ID using 32 nm high-K/metal gate SOI embedded DRAM," *IEEE J. Solid-State Circuits*, vol. 48, no. 4, pp. 940–947, Apr. 2013, doi: 10.1109/JSSC.2013.2239134.

[28] S. Rosenblatt, S. Chellappa, A. Cestero, N. Robson, T. Kirihata, and S. S. Iyer, "A self-authenticating chip architecture using an intrinsic fingerprint of embedded DRAM," *IEEE J. Solid-State Circuits*, vol. 48, no. 11, pp. 2934–2943, Nov. 2013, doi: 10.1109/JSSC.2013.2282114.

[29] A. Rahmati, M. Hicks, D. E. Holcomb, and K. Fu, "Probable cause: The deanonymizing effects of approximate DRAM," in *Proc. 42nd Annu. Int. Symp. Comput. Archit.*, Jun. 2015, pp. 604–615, doi: 10.1145/2749469.2750419.

[30] S. Sutar, A. Raha, and V. Raghunathan, "D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication in embedded systems," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst.*, Oct. 2016, pp. 1–10, doi: 10.1145/2968455.2968519.

[31] N. Mexis, N. A. Anagnostopoulos, S. Chen, J. Bambach, T. Arul, and S. Katzenbeisser, "A lightweight architecture for hardware-based security in the emerging era of systems of systems," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 3, pp. 1–25, Jun. 2021, doi: 10.1145/3458824.

[32] N. A. Anagnostopoulos, Y. Fan, T. Arul, R. Sarangdhar, and S. Katzenbeisser, "Lightweight security solutions for IoT implementations in space," in *Proc. IEEE Topical Workshop Internet Space (TWIOS)*, Jan. 2019, pp. 1–4, doi: 10.1109/TWIOS.2019.8771257.

[33] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann, "PUFatt: Embedded platform attestation based on novel processor-based PUFs," in *Proc. 51st Annu. Design Autom. Conf.*, Jun. 2014, pp. 1–6, doi: 10.1145/2593069.2593192.

[34] S. U. Hussain, M. S. Riazi, and F. Koushanfar, "SHAIP: Secure Hamming distance for authentication of intrinsic PUFs," *ACM Trans. Design Autom. Electron. Syst.*, vol. 23, no. 6, pp. 1–20, Nov. 2018, doi: 10.1145/3274669.

[35] S. U. Hussain, S. Yellapantula, M. Majzoobi, and F. Koushanfar, "BIST-PUF: Online, hardware-based evaluation of physically unclonable circuit identifiers," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 162–169, doi: 10.1109/ICCAD.2014.7001347.

[36] M. Rostami, J. B. Wendt, M. Potkonjak, and F. Koushanfar, "Quo vadis, PUF?: Trends and challenges of emerging physical-disorder based security," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2014, pp. 1–6, doi: 10.7873/DATE.2014.365.

[37] M. Rostami, M. Majzoobi, F. Koushanfar, D. S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 1, pp. 37–49, Mar. 2014, doi: 10.1109/TETC.2014.2300635.

[38] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, Sep. 2002, doi: 10.1126/science.1074376.

[39] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conf. Comput. Commun. Secur.*, Nov. 2002, pp. 148–160, doi: 10.1145/586110.586132.

[40] P. Lugli, A. Mahmoud, G. Csaba, M. Algasinger, M. Stutzmann, and U. Rührmair, "Physical unclonable functions based on crossbar arrays for cryptographic applications," *Int. J. Circuit Theory Appl.*, vol. 41, no. 6, pp. 619–633, Jun. 2013, doi: 10.1002/cta.1825.

[41] G. Csaba et al., "Application of mismatched cellular nonlinear networks for physical cryptography," in *Proc. 12th Int. Workshop Cellular Nanosc. Netw. Appl. (CNNA)*, Feb. 2010, pp. 1–6, doi: 10.1109/CNNA.2010.5430303.

[42] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. van Dijk, "The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2018, no. 4, pp. 243–290, Aug. 2019.

[43] U. Rührmair and D. E. Holcomb, "PUFs at a glance," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2014, pp. 1–6, doi: 10.7873/DATE.2014.360.

[44] D. Nedospasov, J.-P. Seifert, C. Helfmeier, and C. Boit, "Invasive PUF analysis," in *Proc. Workshop Fault Diagnosis Tolerance Cryptography*, Aug. 2013, pp. 30–38, doi: 10.1109/FDTC.2013.19.

[45] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, F. Koushanfar, and W. Burleson, "Power and timing side channels for PUFs and their efficient exploitation," *Cryptol. ePrint Arch.*, 2013. [Online]. Available: https://eprint.iacr.org/2013/851

[46] *Intel Galileo Gen 2*. Accessed: Jun. 2022. [Online]. Available: https://ark.intel.com/products/83137/Intel-Galileo-Gen-2-Board

[47] *Testequity 1007c Temperature Chamber*. Accessed: Jun. 2022. [Online]. Available: https://www.testequity.com/products/598/

[48] *Weisstechnik LabEvent*. Accessed: Jun. 2022. [Online]. Available: https://www.weiss-technik.com/en/products/produkte-detail/laboratory-test-chambers-type-labevent

[49] P. Jaccard, "Distribution de la flore Alpine dans le Bassin des Dranses et dans quelques régions voisines," in *Bulletin de la Societe Vaudoise des Sciences Naturelles*, vol. 37, no. 140, F. Roux, Ed. Chesapeake, VA, USA: Corbaz & Comp., 1901, pp. 241–272, doi: 10.5169/seals-266440.

[50] M. K. Bepary, B. M. S. B. Talukder, and M. T. Rahman, "DRAM retention behavior with accelerated aging in commercial chips," *Appl. Sci.*, vol. 12, no. 9, p. 4332, Apr. 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/9/4332

[51] R. Canetti, M. van Dijk, H. Maleki, U. Rührmair, and P. Schaumont, "Using universal composition to design and analyze secure complex hardware systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 520–525, doi: 10.23919/DATE48585.2020.9116295.

Florian Frank (Graduate Student Member, IEEE) received the B.Sc. degree in computer science from the Munich University of Applied Sciences, Germany, in 2018, and the M.Sc. degree from the University of Passau, Germany, in 2020, where he is currently pursuing the Ph.D. degree with the Computer Science Department. He is also a Research Assistant with the Chair of Computer Engineering, University of Passau. His research interests include hardware security, especially physical unclonable functions and new memory technologies, such as MRAM, FRAM, ReRAM, and FPGAs.

Wenjie Xiong (Member, IEEE) received the B.S. degree in microelectronics and psychology from Peking University, China, in 2014, and the Ph.D. degree from the Department of Electrical Engineering, Yale University, New Haven, CT, USA, in 2020. She is currently an Assistant Professor with the Bradley Department of Electrical and Computer Engineering, Virginia Tech. Her research interests include physically unclonable functions and side-channel attacks and defenses.

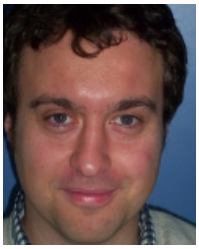
Nikolaos Athanasios Anagnostopoulos (Member, IEEE) received the B.Sc. degree in computer science from the Aristotle University of Thessaloniki, Greece, in 2012, the M.Sc. degree in computer science from the University of Twente, The Netherlands, in 2014, the M.Sc. degree in innovation in information and communication technology from the Technical University of Berlin, Germany, in 2014, and the Dr.-Ing. degree in computer science from the Technical University of Darmstadt, Germany, in 2022. He is currently a Research Assistant with the Chair of Computer Engineering, University of Passau. His research interests include hardware-based security, with a focus on embedded devices, chaos-based cryptography, true random number generators (TRNGs), physical unclonable functions (PUFs), and the Internet of Things (IoT).

André Schaller received the M.Sc. degree in computer science and the Ph.D. degree in computer science from the Technical University of Darmstadt, Germany, in 2012 and 2017, respectively. He is currently an Information Security Engineer with the European Organization for the Exploitation of Meteorological Satellites (EUMETSAT) and a Freelance Security Engineer. His main research interests include hardware-based cryptography, security of embedded systems, and physically unclonable functions in particular.

Tolga Arul (Member, IEEE) received the M.Sc. degree in computer science and the Ph.D. degree in computer science from the Technical University of Darmstadt, Germany, in 2010 and 2016, respectively. He has been a Research Associate with the Cyber-Physical Systems Security Laboratory, Center for Advanced Security Research, Darmstadt, Germany, since 2009. In 2012, he joined the National Research Center for Applied Cybersecurity, Darmstadt. He is currently a Post-Doctoral Researcher with the Chair of Computer Engineering, University of Passau, Germany. His current research interests include trusted computing, embedded security applied to the IoT, transportation, and broadcasting environments. He is a member of the Association for Computing Machinery (ACM).

Farinaz Koushanfar (Fellow, IEEE) received the M.A. degree in statistics and the Ph.D. degree in electrical engineering and computer science from UC Berkeley. She is a Professor and the Henry Booker Faculty Scholar with the Electrical and Computer Engineering (ECE) Department, University of California at San Diego (UCSD), where she is the Founding Co-Director of the UCSD Center for Machine Intelligence, Computing and Security (MICS). Her research interests include efficient computing and embedded systems, with a focus on system and device security, safe AI, privacy preserving computing, real-time/energy-efficient AI under resource constraints, design automation, and reconfigurable computing. She is a fellow of the Kavli Foundation Frontiers of the National Academy of Sciences. She has received several awards and honors for her research, mentorship, teaching, and outreach activities, including the Presidential Early Career Award for Scientists and Engineers (PECASE) from President Obama, the ACM SIGDA Outstanding New Faculty Award, the Cisco IoT Security Grand Challenge Award, the Qualcomm Innovation Award(s), the MIT Technology Review TR-35, the Young Faculty/CAREER Awards from NSF, DARPA, ONR and ARO, and a number of best paper awards.

Stefan Katzenbeisser (Senior Member, IEEE) 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 with the Technical University of Darmstadt, he joined the University of Passau in 2019, where he is heading the Chair of Computer Engineering. His current research interests include embedded security, data privacy, and cryptographic protocol design.



Ulrich Rührmair received the M.Sc. degree in mathematics from Oxford University, the first Ph.D. degree in computer science from TU Berlin, and the second Ph.D. degree in electrical engineering from TU Munich. He is a Research Professor with the University of Connecticut and also a Guest Professor with LMU München, where he has been a Co-Speaker of the research focus on physics and security with the Center for Advanced Studies, since 2022. His research interests include applied cryptography and computer security, in particular PUFs and related physical security primitives. He is the Founder and the Current Steering Committee Chair of the ASHES Workshop, an annual workshop at ACM CCS, since 2017. Furthermore, he is an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *Journal of Cryptographic Engineering*, *Journal on Hardware and Systems Security*, and *EURASIP Journal on Information Security*.

Jakub Szefer (Senior Member, IEEE) received the B.S. degree (Hons.) in electrical and computer engineering from the University of Illinois at Urbana–Champaign and the M.A. and Ph.D. degrees in electrical engineering from Princeton University. He worked with Prof. Ruby B. Lee on secure hardware architectures with Princeton University. He is currently an Associate Professor with the Electrical Engineering Department, Yale University, where he leads the Computer Architecture and Security Laboratory (CASLAB). His research interests include the intersection of computer architecture, hardware security, and FPGA security.