UnTrustZone: Systematic Accelerated Aging to Expose On-chip Secrets

Jubayer Mahmod
Virginia Tech
jubayer@vt.edu

Matthew Hicks
Virginia Tech
mdhicks2@vt.edu

***Abstract*—As technology scaling brings society closer to the vision of smart dust, system designers must address the threat of physical attacks. To address the threat of physical access to computing devices, defenders move secrets on the chip, keeping them out of reach of non-nation-state-level attackers. Modern systems allow hardware-backed security enclaves called Trusted Execution Environments (TEEs); TEEs add hardware-level protections on top of keeping secrets on chips that extend protection against privileged software and flaws within the untrusted parts of the software. While the best TEEs protect against concurrent and temporally recent attacks (e.g., the cold boot attack), we uncover a new threat to all forms of on-chip crypto: long-term data remanence.**

We show that the most ubiquitous form of on-chip memory, Static Random-Access Memory (SRAM), changes at the analog-domain-level in a data-dependent way as software uses it. Under normal conditions, these changes occur gradually over a device's lifetime, but we show how an attacker can systematically accelerate this data imprinting on SRAM's analog domain to effectively burn-in on-chip secrets. We then reveal the imprinted secrets through measurements of SRAM's power-on state. We use this capability to demonstrate three attacks: one that reveals an AES key protected by TrustZone, proprietary firmware protected by TrustZone, and secrets stored in cache memory. Overall, we show that it is possible to imprint and exfiltrate secrets from a range of SRAM-based memories across 13 devices, from 8 manufacturers, produced across three decades—with up to 98% accuracy. To address this threat, we provide guidance to chip vendors and programmers on the defensive trade space.

1. Introduction

In 1988 Mark Weiser coined the term ubiquitous computing to describe his vision of a future society where computers are deeply-ingrained, ever-present, yet invisible [91]. We are rapidly fulfilling this vision as transistor scaling brings us into the Internet of Things (IoT) era of computing and brings smart dust [57] within reach. As a consequence, we are already seeing critical infrastructure [51], medical devices [78], and defense systems [14] become more dependent on IoT devices. This dependence will only increase in the future as we fulfill Mark Weiser's vision.

Given society's increasing dependence on IoT-scale systems, their security is paramount. Security concerns for this class of devices extend beyond more traditional devices to emphasize the physical domain. This is due to their ingrained nature, where physical access is the common case. Thus, one of the grand challenges of security in the era of ubiquitous computing is protecting secrets when the attacker can hold a device in their hands.

An accepted solution to the threat of physical access is to shrink a system's security perimeter to the chip that performs the security-critical computation. This prevents an attacker from interposing and capturing secrets while they are in-flight between system components or at-rest in a less-secure component. For example, the cold boot attack captures secrets while they are at-rest in DRAM chips [31]. In the cold boot attack, the attacker leverages the data remanence effect of DRAM (enhanced by reducing the chip's temperature) to transfer the DRAM chip(s) from the victim's system to the attacker's system—without corrupting the data. Once the attacker installs the DRAM into their system, they employ forensics tools to hunt for secrets (e.g., cryptographic keys). To eliminate such attacks, researchers advocate fully on-chip secure computation, where plaintext secrets are never allowed off-chip [64], [23], [18].

Recently, researchers and processor vendors alike further increase system security by constricting the security perimeter to hardware-enforced partitions within the chip (called enclaves), creating a Trusted Execution Environment (TEE). TEEs increase security by protecting against bugs in software executing outside the TEE—even if it is privileged software. The most widely deployed TEE is ARM TrustZone [6]; most modern ARM processors offer the TrustZone extension, including both microcontroller- and application-class devices. TrustZone provides hardware support to software by switching a core between a *Secure World* and a *Normal World*, where the *Secure World* can access the entire system, but the *Normal World* cannot access *Secure World* resources. At run-time, TrustZone hardware checks the access permission of each transaction to ensure that it is within the appropriate security boundary; attempts to access *Secure World* resources from the *Normal World* trigger a *HardFault*. Fully on-chip crypto-operations combined with TEE support, as exemplified in as *Cache assisted Secure Execution (CaSE)* [97], create the most comprehensively secure software execution environment today.

Undeterred, researchers continue to identify weaknesses

that processor vendors must address in their defenses. Attacks on TrustZone focus on side-channels created by implicitly shared physical resources. The first attack type exploits shared cache and TrustZone's security-state-oblivious eviction policy [17], [49]. The second attack type exploits the energy management policy, which applies across both *Secure* and *Normal* worlds, to induce faults in security-critical software executing in the *Secure World* [86], [66]. The solution to these attacks is a further reduction in the security perimeter, but an open question remains, "*Is security perimeter reduction sufficient to eventually secure the system?*"

We show that the current focus on placing temporally-adjacent restrictions is insufficient by uncovering and exploiting the threat of long-term data remanence. Data remanence is when a memory device retains information past when it is assumed to no longer exist. Short-term data remanence (due to capacitance in the circuit) is well-studied, having led to cold-boot-style attacks on both DRAM [31], [88], [11] and SRAM [83], [84], [95], [89]. In a cold-boot-style attack, attackers leak secrets by resetting software or moving memory between machines, before the memory loses state. The attackers can then use their own software to copy the latent data in the memory device—without restrictions of the original software/system. More recently, short-term data remanence attacks have been used to clone SRAM-based Physical Unclonable Functions (PUFs) [95] using the same process. **Given the threat of data remanence, TrustZone eliminates such attacks by wiping all secure memory as part of all *Secure World* software updates.**

We show that wiping memory is insufficient as circuits "remember" data for much longer through subtle changes to their analog-domain properties—changes that depend on how the software uses the circuits. Previous research reveals that circuits undergo a process called aging [42], [2], [46], and this aging is data-dependent [30]. Researchers have leveraged data-dependent aging in memory circuits and processing cores as a denial-of-service attack on SRAM-based PUFs [74] and processors [39] and to reveal a small fraction of RSA key bits [4], [33], [58] in simple discrete SRAM chips. This paper builds on those papers, showing how device aging can be controlled and accelerated exponentially to reveal up to 98% of secret data stored in on-chip SRAM—even in the most recently produced ARM TrustZone devices. Our results motivate extending existing short-term data remanence defenses to include long-term data remanence.

There are two challenges that we address to create a practical attack out of long-term data remanence: ❶ measuring analog-domain changes in an efficient and non-destructive manner, i.e., a digital measurement, and ❷ increasing the rate of analog-domain change such that secrets measurably change the system at the digital level. We solve the first challenge by leveraging a unique property of the memory used to hold secrets on-chip SRAM: When SRAM cells power on at boot time, its construction leads to a hardware-level race condition to set the initial value of the cell (i.e., its power-on state). The relative analog-domain properties of the logic gates at the heart of the SRAM cell influence this hardware-level race. Thus, by capturing the power-on state of an SRAM cell, we get a digital window into its analog-domain properties. As we show, these analog-domain properties depend on the value held by the SRAM cell—which software dictates—in a process known as transistor aging; meaning that software gradually imprints itself and its data into SRAM's power-on state over a device's lifetime. Since it is not practical to wait for secrets to imprint themselves over several decades, we solve the second challenge by using non-invasive stressors (i.e., temperature and voltage) to accelerate the imprinting process over 100,000,000x. **The resulting foundational capability is the power to pause software, imprint software's secrets into SRAM for less than 24 hours, and reveal the imprinted secrets through power-on state samples with up to 98% accuracy**.

This paper makes the following technical contributions:

- We demonstrate that modern SRAMs are sensitive to accelerated stress, which allows an attacker to retrieve on-chip secrets without sophisticated tools (§5).
- We demonstrate how SRAM's aging creates a low-cost side-channel for on-chip cryptographic key exfiltration (with 97.2% accuracy) (§6).
- We show how an attacker clones proprietary firmware (with up to 95.82% accuracy) that is protected using TrustZone-enforced on-chip execution (§7).
- We extend the attack to general-purpose processors and show how cache memories are susceptible to long-term data remanence attacks (with 79%- 93% accuracy)(§8). We combine all three attack scenarios in a commodity Cortex-A72 processor to demonstrate the generality of the proposed attack (§9).
- We discuss the defensive landscape from both software and hardware levels (§10).

2. Background

TrustZone, a widely deployed security extension for ARM-based devices, facilitates TEE for secure software by applying hardware-enforced barriers among applications; our attack reveals on-chip secrets from such systems exploiting transistor aging and SRAM's power-on state. Aging-induced change in the power-on state is slow and takes decades to leave exploitable data traces under normal operation. From an attacker's perspective, artificial stress eliminates these downsides of the natural aging response of a device. This section provides background on TrustZone, SRAM's power-on state, and aging acceleration methods to imprint software secrets into an SRAM's analog domain.

2.1. ARM TrustZone

A TrustZone divides a system into the *Secure World* and *Normal World* where the same physical core executes secure and non-secure applications in a time-sliced manner by switching between the *worlds* [6]. The secure state has full system access, while the non-secure state (*i.e.,* Normal

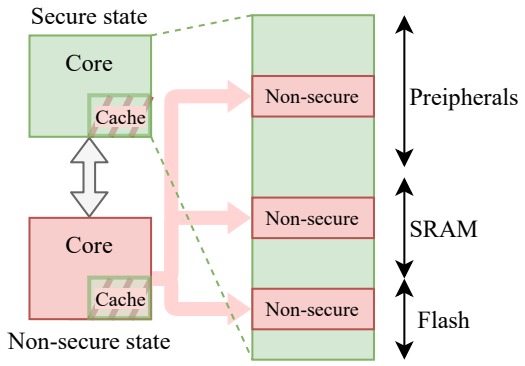


Figure 1: An abstract view of a TrustZone-capable processor and its different access permissions throughout the system. In an ARMv8-A architecture, caches are shared between *Secure World* and *Normal World*. ARMv8-M architecture allows configurable partitions in the processor's memory map.

World) only has access to non-secure memory regions. An `NS` flag identifies the security state of the processor, and the *AMBA* bus carries this flag to all the system components to maintain hardware-enforced isolation between secure and non-secure transactions. The tag of each cache line carries an `NS` security flag to indicate whether the line belongs to a *Secure World* or a *Normal World*. It is architecturally impossible to access a secure cache line with non-secure access [9]. While in a *Secure World*, a core can switch to *Normal World* and vice versa by executing specific instructions such as `SG` and `SMC` for Cortex-M devices and Cortex-A devices, respectively. Figure 1 illustrates an abstract view of a processor with TrustZone extension.

Cortex-M devices allow *Normal World* applications to request services from *Secure World* using Non-Secure Callable (NSC) functions. The NSC functions are 'black box' to a non-secure application, and any data inside these functions are not exposed to the *Normal World*. A call to these functions switches the core into *Secure World*, and then it executes the function.

A Cortex-M device can be configured into multiple levels of security settings in its lifetime. The hardware isolation paves the way to allow independent software development for *Secure World* and *Normal World*, providing security to the proprietary software and its data. For example, one user can design only secure software and program a device before handing it over to another user. The second user programs the non-secure part of the system without any access to the secure area other than through NSC functions. The only way to gain upper-level security access is by erasing the entire chip or having an authorization key from the secure user. *UntrustZone presents an innovative approach to accessing secrets across security boundaries by leveraging SRAM's analog behavior* (§6, §7, and §8).

2.2. SRAM Power-on State

Figure 2a illustrates a typical 6-transistor schematic of an SRAM cell. The cross-coupled CMOS inverters store a bit of data through positive feedback. Outputs of Inverter ① and Inverter ②, Q and ~Q, respectively, are connected to each other's inputs, forming a loop. When *Word Line* is asserted, transistors N3 and N4 provide access to the data stored in the loop. These two transistors remain off unless the address decoder performs a reading or writing operation.

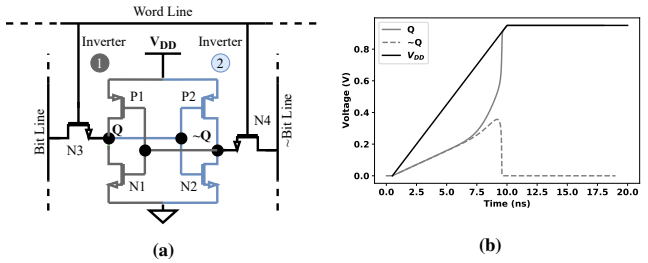


Figure 2: (a) Schematic of a conventional 6-transistor SRAM cell and (b) power-on behavior of an SRAM cell at the *22nm* technology node.

At power-on, nodes Q and ~Q are the ground state and gradually settle their respective steady-state voltages; hardware race condition between Inverter ① and Inverter ② determines the final values [22]. Since one inverter feeds the input of the other inverter and outputs are the same voltage at power-on, the hardware race condition puts these two nodes into opposite logic states. We simulate this behavior using *HSpice* (@22nm predictive technology model [10]) and plot the resulting waveform in Figure 2b. The output nodes, Q and ~Q, follow the supply voltage ramp and resolve into opposite logic states long before the V_{DD} ramp settles.

Inverter ① and Inverter ② are electrically identical as they are designed to be mirrored versions. A cell with such symmetric characteristics produces random power-on values depending on the operating condition of the cell. Most cells, however, power on into a defined logic state due to the asymmetry introduced by manufacturing-time process-related variabilities [3], [96]. As a result, some SRAM cells show random power-on values across trials (*i.e., weak cells*), while others show a strong bias towards one logic state over the other (*i.e., strong cells*). A cell's bias 0.5 indicates that 50% of the time, it powers on as logic 0, and in the other 50% of the trials, it becomes 1. We dump the power-on values of a SAML11 device [59] multiple times and construct a distribution of the bias of each cell. Approximately 10% cells show weak characteristics ($0 < bias < 1$), and 90% cells strongly prefer either logic 1 or 0. Figure 3 illustrates a 128×512-bit snippet of the SAM11's SRAM power-on state (majority-voted). In the next section (§2.3), *we will show how an attacker can change this hardware race condition (and power-on state) to reveal the value stored in an SRAM cell.*

2.3. Analog-domain Changes and Power-on State

As transistors operate, they go through electrical and thermal stress, which affects their performance. Physical phenomena such as Hot Carrier Injection (HCI) [12] and Bias Temperature Instability (BTI) [2], [46] are responsible for a transistor's performance degradation over time, which

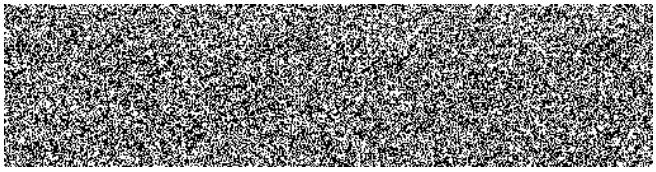


Figure 3: Heatmap of a SAML11 device's power-on SRAM state.

is known as *aging*. Aging results in a gradual increase of the threshold voltage ($|V_{th}|$) and reduced carrier mobility (μ) of transistors [80]. Hot carrier injection is responsible for aging when a transistor switches between logic states, and switching frequency determines the magnitude of HCI-induced aging [80]. Bias temperature instability ages a transistor as long as it remains ON (*i.e.,* gate to source voltages $|V_{gs}| > |V_{th}|$). In an SRAM cell, regardless of the stored logic state, two transistors experience BTI-induced aging: in Figure 2a, Q = 1 ages P1 *and* N2 while Q = 0 ages P2 *and* N1. A MOSFET turns ON in the *saturation region* where the drain current is $I_d \propto (|V_{gs}| - |V_{th}|)^2$ [77]. A transistor's drain current I_d reduces over time because the aging-induced degradation increases the threshold voltage ($|V_{th}|$), slowing it down. Note the impact of BTI in PMOS (NBTI) is much more dominant than BTI's effect on NMOS (PBTI) beyond 45nm Technology [40]. However, PBTI has a significant impact on aging a modern transistor with high-K dielectric and metal gate [43].

Aging affects the switching speed of a transistor, which in turn influences the outcome of the hardware race condition at an SRAM's power on. That is, as the power-on state depends on this race condition of an SRAM's startup, *aging* potentially *impacts* the *state* at which the cell eventually resolves its transient state. For example, let us assume the cell in Figure 2a powers on at logic 1 because Inverter ① turns on faster and pulls the node Q to V_{DD}. If we leave the cell operational in this state (with Q = 1, and ~Q = 0), P1 undergoes static stress because $|V_{gs}| > 0$ (we ignore the effect of NMOS aging for this example). According to the charge trapping/de-trapping theory of BTI [93], the threshold voltage of P1 ($|V_{th_{P1}}|$) logarithmically increases over time, making P1 slower. When this analog-domain change makes Inverter ② faster than Inverter ① (*i.e.,* $|V_{th_{P1}}| > |V_{th_{P2}}|$), the power-on state of this cell flips (*i.e.,* the cell's post-stress power-on state is 0). On the other hand, if the cell undergoes aging with logic 0, the initial speed difference between P1 and P2 increases in the same direction; transistor P2 becomes slower, reinforcing the power-on state (logic 1) of this cell. The complement of the power-on state indicates the logic state it holds during the aging process. *Thus, given sufficient operation (aka stress), the power-on state of an SRAM cell reveals the software value held by that SRAM cell during operation/stress.*

3. Attack Overview

In the overarching threat model of our attack UntrustZone, we assume *an attacker with physical access* to a device that holds secrets on-chip guarded by countermeasures (e.g., TrustZone) to conventional hardware-and software-level attacks. We create target-information- and SoC-specific threat models to represent distinct attack scenarios, each fulfilling the overarching threat model.

We observe that SoCs allow access to uncontaminated SRAM power-on state and core voltage pins, and these architecture/circuit-level features are exploitable to induce aging in SRAM (and imprint secrets). This observation leads us to develop UntrustZone attack. Its secret retrieval accuracy depends on aging acceleration (1) voltage, (2) temperature, and (3) stress time (§5.2).

The attack begins with identifying the suitable voltage pin(s) and corresponding off-chip power delivery components, followed by measuring the nominal voltage at the pin(s). Once these attack parameters are available from experiments with guidance from the publicly released device datasheet, we place the target device in the stress condition (*i.e.,* accelerated aging). The secrets imprint into the analog domain of SRAM during the 'burn-in' effect (*i.e.,* aging) of the victim device, and after a full chip erasure or privilege change (due to TrustZone enforcement) SRAM's power-on state reveals the secure-access-only information to non-secure state. To demonstrate the potential threat of long-term data remanence, we conduct the attack on three recent commodity devices and show that fully on-chip execution with support from hardware-enforced security isolation is insufficient against the proposed attack. The following is a summary of the attacks and their outcomes:

- **Exfiltrate cryptographic key from TrustZone:** In the first attack, we illustrate how UntrustZone exposes a cryptographic key from a system that runs fully on-chip computation, guarded by TrustZone. We use a popular microcontroller, SAML11, which is marketed on its security features [60]. The victim application uses on-chip crypto—in addition to hardware crypto accelerators—to protect the plaintext. Even with these protections in place, we capture the key with **97.2% accuracy** (§6).
- **Exfiltrate proprietary firmware from TrustZone:** In the second attack, we target proprietary firmware designed to be secure against off-chip cloning. Despite being fully on-chip while the CPU performs computations, UntrustZone is able to expose this firmware's instructions and data from the SRAM with up to **95.82% accuracy** (§7). For this demonstration, we use a dual-core LPC55S69 microcontroller [63].
- **Exfiltrate secrets from cache:** In the third case, we extend the attack to target more complex SoCs (Cortex-A53 and Cortex-A72 from Broadcom), which use private caches as nonvolatile secret storage (§8 and §9). We steal secrets from caches of a Cortex-A53 and Cortex-A72 processor with 79% and 93% accuracy, respectively.

When exploring software mitigation, we observe that TrustZone does prevent access to secrets by erasing memory when security access level changes, but long-term data

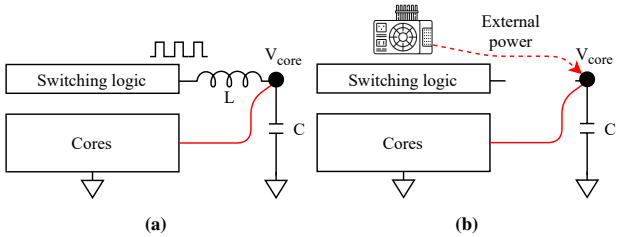


Figure 4: Illustrates the target system's (a) nominal power supply configuration where a switching regulator supplies the SoC's cores. (b) Our attack configuration, where we remove the inductor (L) and connect an external voltage probe to control the victim's voltage.

remanence still reveals secrets through the analog domain. Software-level defenses to UntrustZone include preventing access or erasing the power-on state during an SoC's boot phase and/or scrambling SRAM data at runtime to reduce an adversarial *accelerated burn-in* effect. These defenses have shortcomings, such as eliminating security applications of SRAM power-on state and boot speed reduction. On the hardware side, system and SoC designers have options to limit electric and thermal acceleration and control debug access (once TrustZone has been deployed). While effective, these mitigation techniques pose many challenges, such as strictly modeling runtime core voltage fluctuation and enforcing debug authentication. To facilitate future defenses, we provide a detailed qualitative analysis of the trade-space between software- and hardware-based defenses in Section 10.

4. Identifying Attack Enablers

An attacker needs access to an SRAM's uncontaminated power-on state and a way to accelerate the secret imprinting to launch a long-term-data remanence attack in a reasonable time frame because normal circuit aging takes decades to 'burn-in' data in SRAM's analog domain [54]. We apply high voltage and temperature to accelerate aging so that we can imprint data to and extract data from SRAM's analog domain within hours. While prior work shows that higher than nominal voltage accelerates circuit aging, they fail to recognize the circuit-and system-level complexity associated with the modern SoC, which prevents elevating SRAM voltage [4], [33], [74]. *This leaves numerous questions unanswered, for example, how to elevate a cache's voltage to attack cache-based on-chip computation while keeping the processor from resetting the data on the cache?* That is, attack execution techniques on a discrete SRAM chip and a complex SoC are significantly different; *UntrustZone bridges this gap* by exploiting the power delivery network of a modern SoC to provide access to the internal power rail and accelerating the secret imprinting process. The following discusses three hardware and software challenges that we address to execute UntrustZone on real-world devices.

C1: Overdrive SRAM's power bus.

Electric and thermal stresses during regular operation are responsible for a transistor's performance degradation (*i.e.,* aging), but from an attack perspective, natural aging is slow and impractical for exposing secrets [54]. Exposing a device to higher voltage and temperature induces stronger electric and thermal stress expediting aging.

Modern microprocessors are equipped with brown-out detectors and high-voltage protection circuitry to prevent under and over-voltage in their power buses. Applying an increased voltage directly at the V_{DD} pins does not increase the SRAMs voltage because of internal protection circuitry and voltage regulators. Such an attempt only overheats a device due to a significant voltage drop in the linear regulators (or voltage clipping circuitry). However, this does not impact the memory domain voltage in a way that would induce accelerated imprinting.

In SoCs, however, the internal power supply system divides logical blocks (and physical blocks) which are electrically reachable [53] from external pins bypassing protection and regulator circuitry. SOCs need dedicated external voltage pins (*i.e.,* V_{core} to supply power to the core components) to filter noise generated from fluctuating power demand or to supply power from efficient switching regulators instead of linear regulators.[1] The passive circuit components needed for these operations are large and must be placed on the Printed Circuit Board (PCB); we elaborate on this concept as follows:

Noise-free power delivery: The power consumption of a processor depends on micro-architectural events, and a power delivery system must handle the fluctuation generated by a variable workload. High current fluctuation generates voltage drop in the supply input (V_{DD}) due to parasitic inductance from the die, package, and PCB (*i.e.,* $V_{drop} = L\frac{dI}{dt}$) [44], [45].

Efficient power delivery: The most common reason for leaving internal voltages (*e.g.,* V_{core}) exposed comes from a system's need for energy efficiency at run-time. A device can run from both linear (*e.g.,* LDOs) and switching regulators (*e.g.,* buck converters). Switching regulators are inherently more efficient for meeting variable demands as they control voltage levels by changing the *duty cycle* of a pulse which is then filtered out to a *stable voltage* using passive electronic components such as diodes, inductors, and capacitors, before feeding it to the CPUs and memories.

As mentioned, the passive components are placed outside the SoC to save the die area, manage heat, avoid timing violations, and maintain a wholly-digital process. Figure 4a illustrates a power supply seen in typical SoCs. The duty cycle of the generated pulse from the switching circuit controls V_{core}. Caches (and most other SRAM) are tightly coupled with the CPU core and, by default, draw energy from the same internal power rail (i.e., V_{core}). Our observation is that removing the inductive component while keeping the supply voltage at a nominal level using an

1. Microcontrollers usually boot from internal Low Drop-Out (LDO) regulators, then switch to switching regulators (if available) to save energy due to their power efficiency compared to LDOs.

external power supply (Figure 4b) allows direct control to the SRAM power supply rail, bypassing complex power supply circuits that sit between V_{DD} and the internal power rail that feeds SRAM.

C2: Capture SRAM's power-on state from software.

Processors allow access to SRAM's power-on state like any other memory read operations if it is the main memory (*e.g.,* microcontrollers) or memory-mapped cache (*e.g.,* RISC-V architecture [81]). Some processors do not provide access to their caches directly using a memory-mapped interface, such as *Cortex-A profile* processors. However, we find that these processors provide coprocessor instructions to debug cache coherency issues and low-level memory translation errors [8]. ARM coprocessors are capable of performing maintenance operations without enabling the caches and regardless of the cache lines' validity. These operations allow direct access to the uninitialized states of cache-lines even if they are invalid at startup.

C3: Reduce contamination of SRAM power-on state.

The ability to read SRAM's state directly is not enough; an attacker must be able to capture SRAM's power-on state before it is overwritten—intentionally or otherwise. We observe that processors do not erase the power-on state as part of the hardware reset sequence, as it does not provide any clear benefit. In fact, there are a few downsides to executing power-on state erasure through hardware or software.

On-chip SRAMs can be large, ranging from a few kilobytes to a few megabytes, and distributed in many banks. Iterating over a large memory at a word or line granularity adds significant time to a processor's boot-up. Also, it is not clear (outside of preventing our attack) what value clearing all of a device's SRAM has.

Another deterrent for SRAM initialization is the fact that SRAM power-on states have numerous security applications. First, *weak cells* in an SRAM array capture randomness from the operating noise, which a system uses as a source for random number generation (TRNGs) [32], [67]. Second, *Strong cells* carry unique hardware characteristics that serve as the foundation for Physical Unclonable Functions (PUF) [50], [27], [90], fingerprint generations [32], [16]. Third, the behavior of these cells changes over time due to aging, which is leveraged in counterfeit IC detection [52], [92]. Access to the uncontaminated SRAM power-on state is essential for these applications. These reasons are why processors do not initialize the SRAM [5] as part of their boot-up sequence.

In summary, unrestricted electrical access to an SRAM's power bus and software-accessible, uncontaminated power-on state lead to our analog-domain attack that makes it possible to leak secrets from a secure memory area.

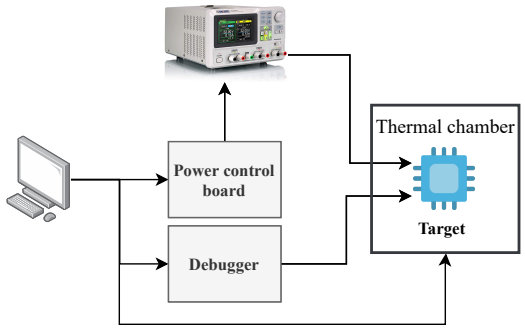


Figure 5: A diagram illustrating the high-level configuration of the experimental setup.

5. Attack Evaluation

This section discusses our baseline experimental setup, and the following sections (§6, §7, §8) show the execution of long-term data remanence attacks on three different scenarios.

5.1. Setup and Target Devices

Figure 5 illustrates a high-level block diagram of our experimental setup, which automates the following tasks:

- Controls experiment temperature and a target's core voltage at different stages of the attack execution.
- Loads software to a target when needed.
- Provides access to the target's memory through a debugger or in-system power-on state extraction software.

Figure 6 shows the essential parts of our experimental setup. The in-house power control board manages different voltage levels while the debugger provides access to the target's internal resources during power-on state extraction. To induce controlled thermal stress, we use a thermal chamber from *TestEquity* [87]. As mentioned in Section 2, some cells are susceptible to operational noise (*e.g.,* temperature, power supply noise, etc.) and behave unpredictably. We model their probabilistic behavior using multiple trials (*i.e.,* power cycles) and calculate a representative power-on state for each cell using majority voting. For every target device, we extract the power-on state 51 times, which, by Bernoulli Trials, filters out (> 99%) the impact of noise. The system remains at a power-off state for 5 seconds between power cycles to discharge all the capacitance in the system, and all of our experiments extract power-on states at 25°*C* and a target's nominal supply voltage.

The list of our target hardware includes a wide range of devices from different silicon vendors, including *NXP, Broadcom, Microchip, and Texas Instruments* (see Table 1). We test the targets for their power-on state accessibility, debugging techniques, and aging acceleration methods.

5.2. Tuning Stress Conditions to Enhance Accuracy

To find an acceleration condition that maximizes retrieved information accuracy, we study SRAM's response

System-on-Chip	Core	SRAM size	TrustZone	Access to uncontaminated power-on state	Aging acceleration	Manufacturer
ATSAML11E16A [59]	ARM Cortex-M23	16KB	✓	✓	✓	Microchip
LPC55S69JBD100 [62]	Dual-core ARM Cortex-M33	320KB	✓	✓	✓	NXP
M263KIAAE [21]	ARM Cortex-M23	96KB	✓	✓	✓	Nuvoton
M2351SFSIAAP [19]	ARM Cortex-M23	96KB	✓	✓	✓	Nuvoton
M252KG6AE [20]	ARM Cortex-M23	32KB	✓	✓	✓	Nuvoton
M251SD2AE [20]	ARM Cortex-M23	12KB	✓	✓	✓	Nuvoton
STM32L562 [85]	ARM Cortex-M33	40KB	✓	✓	✓	STMicroelectronics
BCM2837 (RPi3) [69]	Quad-core ARM Cortex-A53	L1:128KB, L2:512KB	✓	✓	✓	Broadcom
BCM2711 (RPi4) [70]	Quad-core ARM Cortex-A72	L1:320KB, L2:1MB	✓	✓	✓	Broadcom
R7FS1JA783A01CFM [25]	ARM Cortex-M23	32KB	✗	✓	✓	Renesas Electronics
MSP432P401 [35]	ARM Cortex-M4	64KB	✗	✓	✓	Texas Instruments
MSP430G2553 [36]	MSP430 single cycle	0.5KB	✗	✓	✓	Texas Instruments
EFM32WG990F256 [82]	ARM Cortex-M4	32KB	✗	✓	✓	Silicon Labs

Table 1: A list of devices that we verified are susceptible to our long-term data remanence attack.

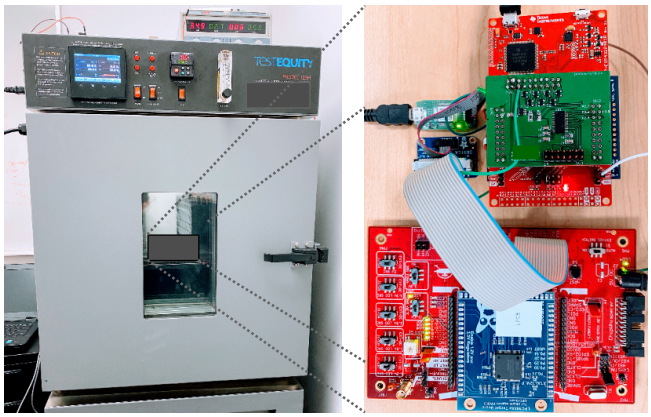


Figure 6: Pictures of our experimental setup. Detachable target hardware sits on a CW308 baseboard [34] (bottom right), which is connected to the power-control board and debug hardware (top right).

by filling three SAML11 [59] SoCs with *all 0s* (to allow static aging of cells) and exposing them to different aging conditions. Figure 7a shows an upward shift in the number of 1s because stress-induced degradation inverts the power-on state of a cell when we stress it with the power-on state stored (§2). Following a logarithmic trend, the $0 \rightarrow 1$ conversion rate (*i.e.,* aging-induced change) reduces over time. We observe an insignificant change in the number of 1s in the SRAM when it operates under nominal voltage and elevated temperature ($V_{core} = 1.2V$ and $T = 85°C$), *but* elevated voltage ($V_{core} = 4.8V$) combined with high temperature ($T = 85°C$) increases the acceleration significantly. Although SoCs allow operating temperatures higher than $85°C$, we keep it at the standard industrial limit to stay within a realistic attack scenario as other components in the target system (*e.g.,* soldering) become vulnerable to damages beyond this temperature.

To understand how *strong* and *weak* cells behave under stress, we conduct another experiment with three SAML11 devices. We write all 1s to one chip and all 0s to another chip and expose them to stress ($V_{core} = 4.8V$ and $T = 85°C$) for 16 hours, which allows us to observe how cells change when stressed with both logic states. The third chip contains its power-on state at nominal operating conditions (*i.e.,* no stress). Figure 7b plots the relation between pre and post-stress bias. The SRAM with a power-on state in it and nominal operating condition still contains *weak cells* (cells along the diagonal line). Chips that undergo accelerated aging have no weak cells present in both pre-stress and post-stress stages. When a weak cell undergoes stress, the data-directed aging aligns its power-on state with the complement of the data; in the following sections (§6, §7), we will show how this observation allows accuracy improvement in retrieved information.

We cannot complement the power-on state for all the cells: the relative power-on speed between the inverters (§2) can be so different that the aging-induced degradation fails to reverse the relation. This phenomenon introduces errors in information retrieval. Longer stress time improves the error performance of the attack, and to provide a better picture of how stress time affects the accuracy of retrieved data, Table 2 provides accuracy vs. stress time for LPC55S699 and SAML11 devices, respectively. *It is evident that longer stress time increases accuracy for both devices, although the rate of changes decreases over time.*

Stress time (hours)	8	16	24
Accuracy	77.34%	81.78%	87.99%

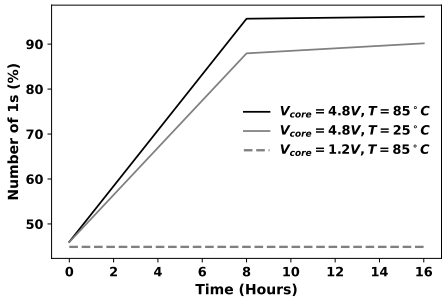
(a)

Stress time (hours)	2	4	8	16
Accuracy	88.23%	90.66%	95.67%	96.11%

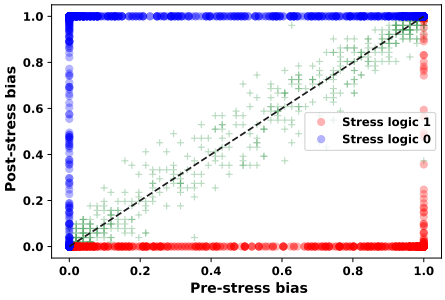
(b)

Table 2: Effect of stress time on the retrieved information accuracy in (a) an LPC55S69 [62] device. and (b) a SAML11 [59] device.

As mentioned before, the magnitude of aging-induced degradation varies across device families. For example, we observe an 18% difference in accuracy between an LPC55S69 and SAML11 after 8 hours of stress at their respective acceleration voltages (§6 and §7), showing significant accuracy variation across device classes. When exposed to the same aging condition, the variation is much less within the same device class (but among different chips). This variation purely comes from the fabricated transistors' process variation, not on-chip or off-chip circuit designs. For example, we observed a 2% variation in the retrieved data accuracy on LPC55S69 devices when they go through the



(a)



(b)

Figure 7: (a) Response of a SAML11 chip when exposed to stress at different stress conditions with extremely biased software (i.e., all 0's). (b) Stress response of cells with different logic states.

same stress condition (*i.e.,* voltage, temperature, and stress time).

The main takeaway is that, although the fundamental vulnerability of long-term data remanence remains unchanged, the accuracy of secret retrieval varies based on the manufacturing technology node and underlying circuit structure of the device. As a result, achieving maximum accuracy requires device-family-specific fine-tuning of stress variables, such as total stress time and acceleration voltage.

6. Attack #1: Exfiltrate an AES key from TrustZone

In this section, we examine a **threat model** where a non-secure user exploits SRAM's *long-term data remanence* to extract a secret key from the *Secure World*. For demonstration, we write a secure application that provides encryption/decryption services to the *Normal World* using NSC functions. The secure application receives encrypted texts from a non-secure application through NSC functions and decrypts them using an on-chip AES engine and a 128-bit secret key. This example emulates a communication scenario from outside of the chip where the *Normal World* is responsible for receiving information, performing error correction, and then forwarding the encrypted information to the *Secure World*. This scenario represents an extreme version of secure executions where TrustZone and on-chip cryptographic hardware guarantee the safety of all operations.

Memory type	Base address	Size (bytes)	Security attributes
Flash	0x00000000	0x7C00	Secure
Flash	0x00007C00	0x0400	NSC
Flash	0x00008000	0x8000	Non-secure
SRAM	0x20000000	0x2000	Secure
SRAM	0x20002000	0x2000	Non-secure

Table 3: The memory map of our system. A secure application (SW1) uses the memory area marked as secure and NSC, whereas the non-secure area belongs to the non-secure application (SW2).

To implement the above attack, we use a device with TrustZone and an on-chip cryptographic accelerator; a SAML11 microcontoller [59] satisfies these requirements. This SoC has a Cortex-M23 core with TrustZone extension, and it features an array of security hardware, including an on-chip crypto-engine (CRYA) designed for IoT applications [60]. SAML11 devices allow setting three Debug Access Levels (DAL), which controls the information accessibly by a debugger. A device with DAL2 setting allows access to any part of the system. A secure user with DAL2 access programs the device and sets the debug access level to DAL1, which prevents the debugger from accessing the secure part of the system. A non-secure user, however, can escalate the debug access level by erasing the entire chip (both volatile and non-volatile memory). This feature prevents access to secure software's code or data without compromising the debugging features available to a non-secure user. Table 3 lists the memory map of our demonstration system.

Figure 8 illustrates the flow of the attack. A secure user with DAL2 access programs the secure software (SW1) in the device (❶) and then restricts the access level to DAL1(❷). At this stage, the device's secure memory is inaccessible to a non-secure application. A non-secure user programs software (SW2) that requests services from SW1 using NSC functions (❸). The non-secure user exposes the device to stress for 16 hours at an elevated voltage (4x) and temperature (3.4x)(❹). Then, the non-secure user erases the full chip to escalate the debug access level from DAL1 to DAL2 (❺) and extract power-on states of the SRAM (❻). Note that TrustZone protects the memory by erasing it before increasing a debug level. The final stage of the attack involves complementing each retrieved bit from SRAM and running a scenario-dependent error correction scheme to construct the key from the memory dump (❼).

Figure 9 plots the heatmaps of the pre-stress memory dump and retrieved information.[2] We locate different sections (*e.g.,* global variables, stack, heap, etc.) of the secure software from the compiled binary and compare it with the retrieved data. **The accuracy of the retrieved data is 97.2%.**

Error source: The mean error in the retrieved words is 2.8%. We report that the words that compose the AES key contain no error; however, we consider the mean error in

2. We store two blocks of data, one with all 0s (black stripe) and the other with all 1s (white stripe), in the non-secure part of the memory to investigate whether there are any discrepancies between the imprinting process in the secure and non-secure area; our results indicate no difference.

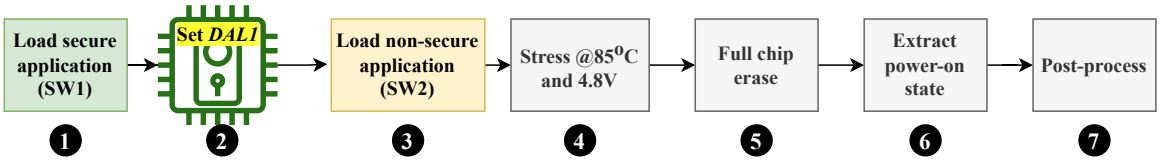


Figure 8: An attack model for *Secure World* AES key extraction.

Stored data	Power-on state		Interpreted data	Correctness	% of bits	Transition type
	Pre-stress	Post-stress				
0	0	0	1	✗	2.19%	Flipping failure
0	0	1	0	✓	30.31%	Flipping success
0	1	1	0	✓	23.11%	Reinforcing
1	0	0	1	✓	26.41%	Reinforcing
1	1	0	1	✓	17.38%	Flipping success
1	1	1	0	✗	0.61%	Flipping failure

Table 4: The cell power-on state changes observed in AES key retrieval attack.

all of our further calculations to keep it applicable to most cases. By analyzing different types of pre-stress to post-stress changes in the power-on state, we investigate the error source (see Table 4). Data in an SRAM cell either reinforces the power-on state or weakens it by affecting the relative current driving strength of inverters (§2.2). Manufacturing variation can make an SRAM cell so strongly biased towards a logic state that stress-induced degradation fails to flip the relative speed of the inverters. This phenomenon leads to errors in post-stress information retrieval. The red lines in Figure 10 illustrate such transitions.

Key extraction scenario #1: We assume the most common case is when an attacker cannot access the device before a *secure user* programs it. That is, the attacker has no access to the pre-stress power-on values for a target device. In this case, the retrieved key is a corrupted version of the original key with a 2.8% error. An exhaustive key search within $n = 4$ bit Hamming distance ($128 \times 0.028 \approx 4\ bit$) of the original $N = 128$ bit key produces an *expected* key search space (L) $\approx 2^{23}$ (see Equation 1), which a modern computer covers in seconds.

$$L = log_2[\sum_{1}^{n} \binom{N}{N-n}] \quad (1)$$

Key extraction scenario #2: In an attack model where a non-secure user has access to a target before shipping it to the secure user for proprietary software programming, an attacker (*i.e.,* the non-secure user) can further reduce the error using enrolled power-on states of a target device.

As discussed above, errors in the retrieved keys come from the cells that retain their pre-stress power-on state even after exposure to stress (see red lines in Figure 10). A probabilistic interpretation, instead of binary 1/0, of a cell's post-stress value improves the accuracy of the retrieved data. If a cell powers on at a defined logic state 100% of the time across trials at pre-stress and post-stress, it is impossible to correct the error generated from it. On the other hand, **pre-stress *strong* behavior and post-stress *weak* behavior reveal the cell's logic state during the stress.** These types of cells contribute to **45.25%** of the total erroneous interpretation: with enrolment power-on, we can correct this error. For example, let us assume a cell always powers on at 1 before stress (*i.e., strong 1 cell*). We expose it to stress, and the cell's power-on state becomes 1 and 0, 80% and 20% of time, respectively (*i.e., weak cell*). Such *weakening* of a cell's bias happens if it undergoes stress with its power-on state. Therefore, we interpret this cell contains 1 during stress as opposed to 0. We apply similar corrections to the *strong 0 cells* that become *weak cell* after stress. These considerations reduce the error from **2.8% to 1.27%**, and at this error at this level, the*expected* key search space comes down to $\approx 2^{13}$.

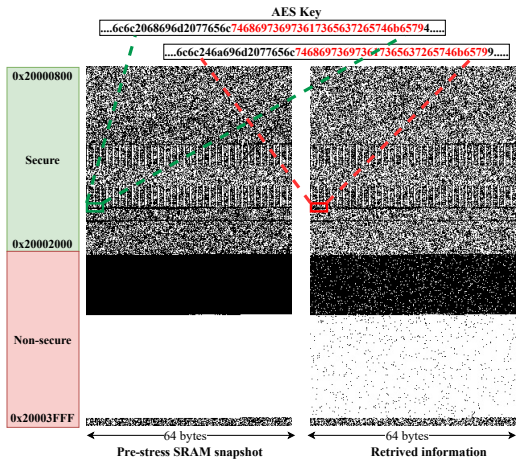


Figure 9: Heatmap of the pre-stress memory dump (left) and post-stress retrieved information (right). Note that the first 2KB of the SRAM is reserved for the on-chip ROM.

In addition to the SAML11 device, we launch this AES key exfiltration attack in a modern Cortex-A72 general-purpose processor in Section 9.

7. Attack #2: Exfiltrate Proprietary Firmware from TrustZone

SRAM provides lucrative benefits that motivate system designers to utilize them as a main memory for firmware

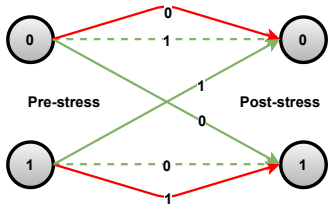


Figure 10: Data-directed aging response of an SRAM cell. When an SRAM cell stores the complement of its power-on state during stress, the post-stress power-on logic remains unchanged (dotted green lines). If it stores the same logic as its power-on state, the post-stress power-on logic flips (solid green lines). Contrary behavior (red lines) causes post-stress data interpretation errors.

execution. For example, cores execute instructions at a much faster speed when in on-chip SRAM compared to other common memories, such as Flash and DRAM. In addition, encrypting software while stored off-chip or Flash and decrypting it only in the on-chip memory prevents software piracy. Therefore, code executes faster and secures itself from low-effort physical attacks when in an on-chip SRAM.

Exposing proprietary firmware, our attack challenges the security of software that executes from an on-chip SRAM. For demonstration, we use an LPC55S69 device, a dual-core Cortex-M33 microcontroller with TrustZone security extension, from NXP semiconductor. TrustZone and other advanced hardware security features, such as crypto accelerators and PUFs are part of CPU0, while CPU1 serves as a secondary core in support of CPU0.

Memory segments	Base address	Size (bytes)
CPU0 RAM	0x20000000	0x11000
CPU1 RAM	0x20012800	0x31800
Shared RAM	0x20011000	0x01800
CPU0 Flash	0x00000000	0xa0000

Table 5: Memory partitions between *CPU0* and *CPU1*.

We design firmware for both cores, and they stay in a TrustZone-guarded secure Flash region. At startup, CPU0 initializes the system and loads the CPU1's firmware in the SRAM, which reduces the need for simultaneous Flash access by the cores, improving the system's security and performance. Once initialized, both cores start executing their respective firmware from two memories — Flash and SRAM. The entire memory region is marked as secure and CPU1 is the secure bus master; Table 5 lists the memory partitions for this experiment.

With a target to expose the TrustZone guarded firmware from the SRAM, we expose the device to stress by elevating voltage and temperature (V_{core} = 5.5V and T = 85°C) for 24 hours. Then, we erase the full chip to escalate debug level and extract the power-on state of the SRAM. We compare the retrieved binary of each 32-bit word (instruction length) with a pre-stress memory dump and plot the fractional Hamming distance in Figure 11. Note that the pre-stress memory dump is for analysis only, *i.e.,* it is unavailable to an attacker. As the firmware for CPU1 stays in the SRAM statically, the power-on states of the cells in that region align with the complement of the stored firmware

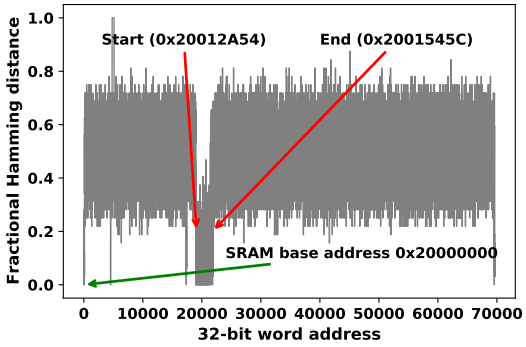


Figure 11: Post-stress fractional Hamming distance in each word (32-bit) of the SRAM (address base 0x20000000). *CPU1*'s firmware is within 0x20012800 to 0x2001545C, showing how firmware burns-in in the SRAM's power-on state.

bits, which leads to lower fractional Hamming distance (and error) in that area (0x20012A54 to 0x2001545C). The area where the fractional Hamming distance is around 0.5 contains a power-on state during stress. Table 6 shows the pre-stress and post-stress transitions among cells' power-on states.

Attack scenario #1: The mean error in our 3-device experiment is 12.4% (see Table 7). Unlike keys, proprietary firmware is the same across devices, which allows an adversary to attack several devices running the same firmware and use majority voting on extracted firmware bits to reduce errors. The error in the retrieved data is largely random, which allows us to estimate the error with Bernoulli Trials in a scenario where an attacker has access to multiple devices. Theoretically, the error becomes 4.2% when the attacker has access to just three devices, each with ≈ 12.4% random errors. Our 3-device experiment retrieves the CPU1 firmware with ≈ 4.2% error, which is the same as the expected theoretical result. We recreate ≈ 45% of the machine code without applying any device or architecture-dependent noise filtering.

Attack scenario #2: Similar to Section 6, we consider an attack scenario where an attacker has access to devices' pre-stress power-on values. Access to enrolment data reduces errors in the retrieved firmware from individual target devices (see Table 7), which leads to much higher accuracy when combined with multi-device error correction. **The firmware extraction accuracy reaches 98.29%**, and at such high accuracy, we recreate 70.90% instructions error-free instructions—without applying more advanced hardware- or instruction-set-aware error correction.

An SRAM's stress-induced degradation shows spatial variation in the LPC55S69 device. Figure 12 plots the percentage of 1s in each 0.5KB block in the SRAM after a device goes through stress with all 0s. We observe different levels of aging responses at 64KB intervals. An LPC device has 5 SRAM blocks; the first four blocks are 64KB each, and the last block is 16 KB [62]. Therefore, firmware retrieval accuracy varies based on which SRAM bank it maps to—

Firmware bit	Pre-stress	Post-stress	Interpreted firmware bit	Correctness	% of bits LPC1	LPC2	LPC3	Transition types
0	0	0	1	✗	7.80%	6.31%	7.08%	Flipping failure
0	0	1	0	✓	23.01%	22.23%	24.23%	Flipping success
0	1	1	0	✓	25.83%	28.04%	25.28%	Reinforcing
1	0	0	1	✓	22.23%	20.41%	23.51%	Reinforcing
1	1	0	1	✓	16.54%	15.93%	15.49%	Flipping success
1	1	1	0	✗	4.59%	6.97%	4.04%	Flipping failure

Table 6: Pre-stress and post-stress cell transitions of three LPC devices.

	LPC1	LPC2	LPC3	Combined
Scenario # 1 accuracy	87.70%	86.70%	88.50%	95.82%
Scenario # 2 accuracy	93.20%	91.76%	93.36%	98.29%

Table 7: Firmware extraction accuracy for three target devices.

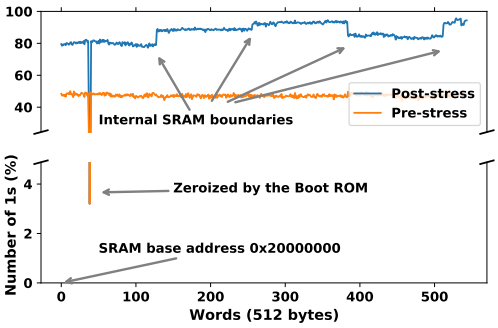


Figure 12: SRAM banks in the LPC device show different levels of stress-induced degradation based on their locations inside the die.

opening the door to the attacker forcing the placement of target code/data[3].

This section shows how UntrustZone extracts proprietary firmware from a Cortex-M33's SRAM. Combining with the previous attack (§6), we demonstrate the attack's capability using a general-purpose processor, Cortex-A72 core, in Section 9.

8. Attack #3: Exfiltrate Secrets from Cache

Many systems and architecture papers consider the *CPU cache* as the panacea against external memory disclosure attacks [23], [18], [97]. This notion stems from the fact that on-chip SRAM are safe from physical attacks, motivating encrypting external memory-bound transactions as a standard across trusted execution environment implementations. We show that long-term data remanence challenges this assumption by revealing secrets held even in the private cache of a CPU.

The ARM CPU's cache lines are shared among the *Secure World* and *Normal World*: a *Secure World* application protects its cache line from *Normal World*'s application using an `NS` security flag. When a cache line carries a secure tag, the line is not available in any *Normal World* software, including the operating systems. Our attack relies on the analog-domain changes of the constituent transistors of a cache, bypassing the security attribute of a cache line. First, we use a quad-core ARM Cortex-A53 processor (BCM2837) that comes with standard *Raspberry Pi 3 Model B(v1.2)* to demonstrate this attack. Then, we summarize all three attack scenarios with a Cortex-A72 processor.

8.1. Accessing Cache's Power-on State

Raspberry Pi (RPi) provides pre-compiled binary blobs that boot an embedded GPU core from external storage devices, *e.g.,* an SD card. It follows a boot chain that involves a combination of both on-chip and off-chip boot instruction sequences. First, the embedded GPU (*VideoCore IV*) boots from an on-chip ROM, which then loads a pre-compiled binary, `bootcode.bin`, in a shared (with ARM cluster) L2 cache. Second, it executes `start.elf` from an off-chip Non-Volatile Memory (NVM), which loads a kernel[4] [68]. We introduce another stage of firmware before the ARM cores start executing a kernel and place it along with the pre-compiled binaries in the external NVM. Our firmware ensures cores startup in `aarch64` mode at `EL3` exception level and utilizes `CP15` coprocessor to extract L1 caches at power-on regardless of the cache lines' validity. `CP15` instructions such as `msr s3_3_c15_c4_1, <Xd>` and `msr s3_3_c15_c4_0, <Xd>` allow reading i-cache and d-cache by iterating over ways, sets, and offsets [7]. Here, `Xd` is a 64-bit general CPU register.

A Cortex-A53 allows invalidating cache lines by executing `IC IVAU` and `DC ISW` instructions for i-cache and d-cache, respectively. Invalidation prevents a CPU from accessing a line but does not reset the cached data in data RAM. For example, `IC IVAU` instruction invalidates the entire L1 i-cache by resetting all i-cache tags to `0x7FFFFFFF`. However, the data in the cache lines remain unmodified, but the processor ignores them (*i.e.,* no cache hit).

We write a tool that programs and controls the core through *openocd* [71] and *JTAG interface*. Cache's power-on state extraction tool performs the following tasks:

1) Disables L1 memories (*e.g.,* caches, TLBs,) and makes the core busy-wait (*e.g.,* `spin: b spin`) to eliminate cache's power-on state contamination and to avoid hitting any undesired exceptions, respectively.
2) Establishes JTAG communication and halts all cores.

3. This scenario resembles *Flip feng shui* [72] attack where an attacker Virtual Machine(VM) allocates memory such that memory deduplication process merges an attacker page with victim VM's page, facilitating row hammering.

4. The source codes for `bootcode.bin` and `start.elf` are confidential, and we do not have access to them.

3) Loads a *cache extraction software*[5] in the main memory (*i.e., SDRAM*). This software reads the contents by iterating over the entire target cache and dumps it in the *SDRAM*.
4) Transfers the *SRDAM* contents to a debug host (*e.g.,* PC) through *JTAG interface*.
5) Repeats steps 3 and 4 for all the target cores and caches.

8.2. Attacking ARM Core's L1 Cache

To apply accelerated stress on a device, we need access to the V_{core} pin(s). The detailed schematic of the Raspberry Pi boards and pin description of BCM2837 SoC are not publicly available. However, a closer examination on public RPi3 reduced-schematic [26] reveals that an `RT8088A` [73] switching regulator supplies 1.2V to the ARM cores and internal SRAM; this a typical power management scheme in almost any modern system [53]. The output of the switching regulator passes through an inductor (`L3`) before reaching the core voltage pins (§4). We remove this inductor from the board (while maintaining the live system voltage externally), which disconnects the switching regulator from supplying power to the SoC. A probe connected at the test pad `PP58` of RPi3 from our power-control board manages the voltage level at the SoC's V_{core}, which allows us to elevate the voltage supplied to the SRAM.

For attack demonstration, we fill out the caches of all the cores with preset instructions fetched from the *SDRAM* and expose the system to stress. **At an acceleration voltage $V_{core} = 2.2V$ and $T = 85°C$, we extract contents of an i-cache with 79.2% accuracy after 120 hours of stress**. Since our *cache extraction software* reads out *invalid* cache lines at startup, it bypasses the TrustZone-enforced security boundary.

9. Unifying the Attack Scenarios

Consolidating the attack scenarios into one system, we summarize UntrustZone using a newer ARM SoC, Broadcom BCM2711, which has a quad-core Cortex-A72 CPU (aka Raspberry Pi 4) [70]. To show the generality of UntrustZone attack by combining all the attack scenarios described above with an overarching threat model that assumes secret data (attack #1) and proprietary software (attack #2) are in the on-chip cache (attack #3).

The first step in the UntrustZone deployment phase is to identify the acceleration voltage pin that is located near the power management integrated circuit ($MXL7704$); this chip drives the V_{core} pin (`TP15` pad in the PCB [65]). As described in section 5.1, we determine the acceleration voltage with guidance from the datasheet [65] and experimentation. While the software and toolchain needed for power-on extraction remain the same as BCM2837 (§8), the acceleration voltage is 1.62V (2.025× nominal voltage).

At attack deployment, the victim cores 0 and 1 run on-chip encryption/decryption using a 128-bit AES engine; this AES does not use memory access once loaded into the L1 cache due to its implementation using vector registers and on-demand calculation of the sub-keys and other intermediate states.[6] Encryption/decryption operations using this AES never save intermediate states in the off-chip memory, making it secure against external memory attacks, such as cold boot [31]. It is essentially the same as attack #1 (§6), where the cryptographic service is provided by the on-chip secure state of a processor. The other two cores execute typical software operations while keeping both software and data in the cores' L1 private cache, making it inaccessible from the RAM in plain text (*i.e.,* attack #2).

The AES key extraction accuracy reaches **93.2% after 120** hours of aging, and we observe similar results in the other two cores' instruction (93.3%) and data (93.4%) extraction. Similar accuracy across different cores and cache types is expected as they share the same power bus, hence experience the same accelerated stress. The mean accuracy of all four cores (both i-cache and d-cache) reaches ~ 95% after 168 hours of stress.

10. Countermeasures

Long-term data remanence is an analog-domain attack that exploits device-level changes to leak on-chip secrets. A viable defense needs careful consideration of both software and hardware avenues. We provide a qualitative analysis of the possible defense mechanisms and their expected implications on the system's design and performance.

10.1. Software Defenses

10.1.1. Initializing the SRAM at startup

SRAM's power-on state is the information carrier for this attack as it provides digital access to the data-directed changes in the analog domain. Therefore, initializing the entire SRAM as a part of the secure boot process prevents UntrustZone attack. This type of countermeasure come with two potential downsides:

- SRAM initialization as part of CPU power-up sequence stretches startup time by a few hundred cycles to thousands of cycles. For example, an LPC [62] device at its highest boot speed (96MHz) takes $640\mu s$ to initialize its 320KB SRAM (61440 cycles), which is a significant delay for embedded applications.
- Initializing an SRAM's power-on state prevents security algorithms (*e.g.,* TRNGs [32], PUFs [55], [75], [95], recycled IC detection [29]) from utilizing hardware uniqueness and randomness at run-time.

Instead of using CPU cores to reset, a more flexible and efficient option is to utilize self-test engines (*i.e.,* MBIST) to reset the SRAMs with an option to configure it according to a system designer's need.

5. The software is in assembly (`aarch64`) to tightly control the execution.

6. The implementation is *stackless* and in assembly, which allows doubt-free post-stress error calculation.

10.1.2. Scrambling the SRAM data at run-time

Balancing the stress between inverters in an SRAM cell is a potential software solution that prevents imprinting data in the aging-induced asymmetry. This scheme requires active run-time support where software periodically inverts or scrambles the stored bits [52]. Our attacks drive the core voltage well beyond the nominal, which freezes a processor's execution, making this software defense ineffective.

10.2. Hardware Defenses

10.2.1. Preventing aging acceleration

Aging acceleration is the key to launching UntrustZone attack, which comes from thermal and electrical stress. Averting thermal acceleration is relatively simpler and can be done with help from modern processors' temperature sensing and management hardware, such as Thermtrip [1] and Frequency throttling. Thermtrip in Intel CPU prevents damaging processors from operating under excessively high temperatures [1], and frequency throttling is common across all high-performance CPUs to keep the devices under their thermal limits. Orienting these mechanisms to detect adversarial aging and erasing sensitive information upon such detection can be a solution to eliminate the thermal component of the acceleration. Given that a device's thermal operating condition varies depending on its applications, a *combined* software and hardware solution is necessary to provide configurable solutions to thermal stress tolerance.

Electric stress is the driving factor in acceleration and is much harder to prevent (§2.3). A device needs to enforce strict operating voltage without any error margin, which is impractical because it requires modeling a power distribution network's behavior well before manufacturing. Process variations, the unpredictability of run-time noise, and the need for multiple operating modes complicate such a strict nominal voltage limit, making most devices respond to accelerated aging to some degree. We observe the following two types of behaviors in target devices that limit accelerated aging using voltage:

- **Bypassing the excess energy through circuits before reaching the core:** As a standard design practice, pins in a chip include measures for over-voltage (*e.g.,* ESD) and reverse-polarity protection. A chip's behavior in these circumstances depends on the design and implementation of the underlying circuitry. For instance, a BCM2837 draws > 2.5A current when core voltage V_{core} goes beyond 2V (1.67×). Such high current consumption indicates an over-voltage prevention mechanism bypasses excess power to the ground by clipping off the excess voltage.
- **Erasing the volatile memory contents:** A hardware measure that resets the CPU and memory beyond recommended voltage is also observable in devices. In experimentation with M2351SFSIAAP [19], we observe that the device resets the I/O registers and SRAMs if the voltage rises beyond 2.3V in its core voltage pin (LDO_CAP)[7].

While these protection mechanisms prevent over-voltage, they kick in only when to voltage crosses the design margin, which is higher than the operating voltage limit. It reduces the acceleration, but longer stress time compensates for the slower acceleration.

10.2.2. Restricting debug access

The proposed attack requires debug access or loading a modified kernel (after resetting TrustZone and deleting the contents of the memory in Cortex-M devices) to read out uninitialized SRAM values from secure memory area. Debug authentication features [61] or mandating a trusted boot process should thwart reading out imprinted SRAM data, ultimately preventing the attack. Disabling debug permanently complicates return material analysis and software update process in lower-end devices, which are essential methods to analyze a system failure and product improvement [79]. Note that there are attacks that bring the device to its factory reset condition or circumvent the disable debug settings [88], [41].

11. Discussion

Our experiments with a range of SoCs uncover *a general trend* among different devices that we expect to hold for relatively more complex processors employed by desktop- and server-class systems. First, an increasing number of devices— including desktop and server-class processors— allow access to their caches directly either through a memory-mapped interface (*e.g.,* RISC-V [81]) or co-processor interface (*e.g.,* Cortex-A53 [8], Cortex-A78 [48], Cortex-A72 [47]). Such architectural flexibility and feature offer several advantages, such as the ability to execute software quickly from on-chip memory, ensuring run-time security against off-chip memory attacks, and facilitating debugging features for low-level memory translation issues.

Second, the requirement for a robust and low-noise power supply from external sources becomes essential as hardware complexity increases. Current fluctuation in complex processors (*e.g.,* Cortex-A53) is significantly more compared to embedded devices. Therefore, we expect complex processors' power delivery systems to be similar to the device we assessed in our experiments.

Third, desktop- and server-class processors allow overclocking by elevating the supply voltage. This means the door is open to using this access for accelerated imprinting on other processors. For example, Intel Core and Intel Xeon E processors allow operating at a nominal 0.9V, but they allow voltage overdrive to at least 1.52V [37].

Fourth, voltage scaling lags behind geometric scaling, which increases the electric field density in a device, making smaller technology nodes more susceptible to aging-induced

7. The voltage at this pin is 1.2V when operating in Normal mode (48MHz) and 1.26V in Turbo mode (64MHz), which leaves a room for accelerating a device's aging.

degradation. This trend leads to increased information leakage from SRAM manufactured using newer technology nodes. For example, the accuracy of imprinted and retrieved information from a 40*nm* LPC device [24] is over 4x more than that of a 130*nm* [58] MSP430 [35], even when the LPC device spends less than half the time having software imprinted into SRAM's analog domain.

Considering the attack enablers outlined in Section 4 and the observed trends that extend to desktop- and server-class processors, we anticipate that our attack methodology will be applicable to those devices as well. Our analysis of Intel SGX and AMD SEV indicates that these more sophisticated TEEs need further experimental evaluation to determine the impact of UntrustZone attack.

12. Related Work

Given the popularity of ARM devices, there is a significant research drive to evaluate the security of ARM TrustZone.

Exploiting Voltage and Frequency Management Hardware: *CLKSCREW* [86] and *VoltJockey* [66] manipulate the energy management systems shared among multiple cores to push it beyond the vendor recommended limit. *CLKSCREW* manipulates the frequency, and *VoltJockey* exploits the voltage from a non-secure kernel driver to induce timing faults in the secure cores so that it produces incorrect computations. The authors show how an attacker bypasses RSA-based signature verification and loads an *untrusted* application into the TrustZone. These attacks apply only to multi-core CPUs with shared dynamic voltage and frequency management systems.

Exploiting Cache and DRAM: *Secure World* and *Normal World* share on-chip memory such as caches and Branch Target Buffer (BTB). Cache allocation and eviction are not restricted by the security attribute of a cache-line. That is, hardware does not restrict a *Normal world*'s application from evicting a cache-line that belongs to *Secure World*. While this design decision optimizes performance, it leads to numerous cache-contention-based side-channel attacks such as *ARMAgeddon* [49], *TruSpy* [98], *Cache storage channels* [28], and *Prime+Count* [17]. Since the Branch Target Buffer is a shared resource between the *Secure* and *Normal World*, cache-contention-based attacks also apply [76]. BTB-based exploits increase the spatial resolution of the probing mechanism because the BTB stores information at byte granularity rather than cache-line granularity [15].

The Rowhammer attack also impacts ARM TrustZone-enabled devices [13], where a malicious kernel from the *Normal World* generates high-rate memory read requests close to a secure memory boundary. These high-rate read requests corrupt an RSA private key in the secure memory (*i.e.,* DRAM), leading to faulty signature generation in the *Secure World*. The comparison of the message and signature in the *Normal World* leaks the RSA private key. Unlike shared-on-chip-buffer-based side-channel attacks, keeping secrets on-chip defeats Rowhammer attacks.

Exploiting Configurable Hardware: AMBA AXI interface of TrustZone-enabled devices contains a security flag (`NS` bit) for *read* and *write* channels in the main system bus. This bit carries the security state of the CPU. Reconfigurable hardware such as FPGA (*e.g.,* Zynq-7000 [94]) can be connected to the system's main bus to allow flexible and efficient hardware/software co-design. Benhani *et al.* show that such heterogeneous hardware architectures are security risks because malicious hardware in the FPGA fabric potentially breaks ARM TrustZone security [56]. Jacob *et al.* show that malicious hardware in the FPGA can interfere with the SOC's secure boot process, allowing a CPU to load authorized kernel [38].

13. Conclusion

In this paper, we show how long-term data remanence is a threat to hardware-backed secure computing environments. Our attack shows, contrary to the assumptions of previous defenses, keeping secrets on-chip—even when guarded by the hardware-level protections of a Trusted Execution Environment—is insufficient to keep them secure when an attacker has physical access to the chip. We implement and validate the long-term data remanence vulnerability on a range of devices spanning complexity, cost, performance, age, and manufacturer. We then use the capability to carry out three attacks. The attacks steal secret code and data from TrustZone-protected main memory and on-chip cache lines. Our results show how an attacker can efficiently imprint software secrets into SRAM's analog domain and reliably retrieve those imprinted secrets through measurements of SRAM's power-on state. Our analysis of the defensive tradespace suggests that a robust solution requires hardware and low-level software modification.

Beyond exposing a universal weakness in secure computing devices, this paper highlights a broader threat: transistor-aging-based side-channels. While the circuits community has long been aware of the effects of transistor aging, their focus has been on its performance and reliability impacts, not security. This paper shines a light on the ramifications of security of data-dependent transistor aging. We believe this paper will inspire future researchers to uncover other security issues related to transistor aging and new side-channels that leverage the latent information left behind by data-dependent transistor aging.

Responsible Disclosure

In accordance with the vulnerability disclosure guidelines, we reported our findings to ARM as well as the manufacturers of the devices against which we validated our attacks, including NXP Semiconductors and Microchip Technology. We made this paper public only after ARM released an architecture security advisory.

Acknowledgements

The project depicted is sponsored by the Defense Advanced Research Projects Agency. The content of the in-

formation does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. Approved for public release; distribution is unlimited.

References

[1] https://www.intel.la/content/dam/www/public/us/en/documents/datasheets/pentium-dual-core-desktop-e2000-datasheet.pdf, accessed: 2023-7-15.

[2] M. A. Alam and S. Mahapatra, "A Comprehensive Model of PMOS NBTI Degradation," *Microelectronics Reliability*, vol. 45, no. 1, pp. 71–81, 2005.

[3] E. Amat, E. Amatllé, S. Gómez, N. Aymerich, C. G. Almudéver, F. Moll, and A. Rubio, "Systematic and Random Variability Analysis of Two Different 6T-SRAM Layout Topologies," *Microelectronics Journal*, vol. 44, no. 9, pp. 787–793, 2013.

[4] R. Anderson and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," in *International Workshop on Security Protocols*. Springer, 1997, pp. 125–136.

[5] ARM Limited, "ARM Cortex-A Series: Programmer's Guide for ARMv8-A."

[6] Arm Limited, "Security in ARMv8-A systems," 2017, https://developer.arm.com/documentation/100935/0100/The-TrustZone-hardware-architecture-?lang=en.

[7] ARM Limited, "Armv8-M Architecture Reference Manual," 2020, https://developer.arm.com/documentation/ddi0553/bn/.

[8] ARM limited, "ARM Cortex-A53 MPCore Processor Technical Reference Manual," 2021, https://developer.arm.com/documentation/ddi0500/j/Introduction/Product-documentation-and-design-flow/Documentation.

[9] ARM Limited, "L220 Cache Controller Technical Reference Manual," 2021, https://developer.arm.com/documentation/ddi0329/l/functional-overview/functional-operation/trustzone-support-in-the-cache-controller.

[10] ASU, "Predictive Technology Model (PTM)," 2020, http://ptm.asu.edu/.

[11] J. Bauer, M. Gruhn, and F. C. Freiling, "Lest We Forget: Cold-boot Attacks on Scrambled DDR3 Memory," *Digital Investigation*, vol. 16, pp. S65–S74, 2016.

[12] A. Bravaix, C. Guérin, V. Huard, D. Roy, J.-M. Roux, and E. Vincent, "Hot-carrier Acceleration Factors for Low Power Management in DC-AC Stressed 40nm NMOS Node at High Temperature," in *International Reliability Physics Symposium*. IEEE, 2009, pp. 531–548.

[13] P. Carru, "Attack TrustZone with Rowhammer," 2017, https://grehack.fr/data/2017/slides/GreHack17_Attack_TrustZone_with_Rowhammer.pdf.

[14] A. Castiglione, K.-K. R. Choo, M. Nappi, and S. Ricciardi, "Context Aware Ubiquitous Biometrics in Edge of Military Things," *Cloud Computing*, vol. 4, no. 6, pp. 16–20, 2017.

[15] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems," in *Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1416–1432.

[16] S. Chellappa and L. T. Clark, "SRAM-based Unique Chip Identifier Techniques," *Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1213–1222, 2015.

[17] H. Cho, P. Zhang, D. Kim, J. Park, C.-H. Lee, Z. Zhao, A. Doupé, and G.-J. Ahn, "Prime+ Count: Novel Cross-world Covert Channels on ARM Trustzone," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 441–452.

[18] P. Colp, J. Zhang, J. Gleeson, S. Suneja, E. De Lara, H. Raj, S. Saroiu, and A. Wolman, "Protecting Data on Smartphones and Tablets from Memory Attacks," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2015, pp. 177–189.

[19] N. T. Corporation, "NuMicro®Family M2351 Series Datasheet," 2019, https://www.nuvoton.com/export/resource-files/DS_M2351_Series_EN_Rev1.01.pdf.

[20] ——, "NuMicro®Family M251/M252 Series Datasheet," 2020, https://www.nuvoton.com/export/resource-files/DS_M251_M252_Series_EN_Rev1.01.pdf.

[21] ——, "NuMicro®Family M261/M262/M263 Series Datasheet," 2020, https://www.nuvoton.com/export/resource-files/DS_M261_M262_M263_Series_EN_Rev1.02.pdf.

[22] M. Cortez, A. Dargar, S. Hamdioui, and G.-J. Schrijen, "Modeling SRAM Start-up Behavior for Physical Unclonable Functions," in *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2012, pp. 1–6.

[23] S. M. Datta, V. J. Zimmer, K. V. Vaid, W. A. Stevens, and A. L. Santoni, "Processor Cache Memory as RAM for Execution of Boot Code," Aug. 7 2007, uS Patent 7,254,676.

[24] EET-asia, "STMicroelectronics and NXP launch ARM Cortex-M33 Based Chips in Bid to Secure the IoT," 2021, https://www.nxp.com.cn/docs/en/application-note/AN13037.https://www.eetasia.com/18101805-mcus-answer-to-iot-security-worries/.

[25] R. Electronics, "Renesas Synergy™ Platform Synergy Microcontrollers S1 Series," 2019, https://www.renesas.com/us/en/document/man/s1ja-microcontroller-group-users-manual.

[26] R. P. Foundation, "RPi-3B-Reduced," 2015, https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3b_1p2_reduced.pdf.

[27] A. Garg and T. T. Kim, "Design of SRAM PUF with Improved Uniformity and Reliability Utilizing Device Aging Effect," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2014, pp. 1941–1944.

[28] R. Guanciale, H. Nemati, C. Baumann, and M. Dam, "Cache Storage Channels: Alias-driven Attacks and Verified Countermeasures," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 38–55.

[29] U. Guin, W. Wang, C. Harper, and A. D. Singh, "Detecting Recycled SoCs by Exploiting Aging Induced Biases in Memory Cells," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2019, pp. 72–80.

[30] P. Gutmann, "Data Remanence in Semiconductor Devices," in *USENIX Security Symposium*, 2001, pp. 39–54.

[31] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, "Lest We Remember: Cold-boot Attacks on Encryption Keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, 2009.

[32] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[33] M. Hutter and J.-M. Schmidt, "The Temperature Side Channel and Heating Fault Attacks," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2013, pp. 219–235.

[34] N. T. Inc., 2021, https://rtfm.newae.com/Targets/CW308%20UFO/.

[35] T. Instruments, "MSP430G2x53 Automotive Mixed-Signal Microcontrollers," 2014, https://www.ti.com/lit/ds/symlink/msp430g2553-q1.pdf?ts=1618423366624&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430G2553-Q1.

[36] ——, "SimpleLink ultra-low-power 32-bit Arm Cortex-M4F MCU With Precision ADC, 256KB Flash and 64KB RAM," 2019, https://www.ti.com/document-viewer/MSP432P401R/datasheet/device-overview-slas8261807#SLAS8261807.

[37] Intel, "8^{th} and 9^{th} Generation Intel Core Processor Families and Intel Xeon E Processor Families," 2021, https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/8th-gen-core-family-datasheet-vol-1.pdf#page=118&zoom=100,94,402.

[38] N. Jacob, J. Heyszl, A. Zankl, C. Rolfes, and G. Sigl, "How to Break Secure Boot on FPGA SoCs Through Malicious Hardware," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 425–442.

[39] N. Karimi, A. K. Kanuparthi, X. Wang, O. Sinanoglu, and R. Karri, "Magic: Malicious Aging in Circuits/Cores," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 12, no. 1, pp. 1–25, 2015.

[40] N. Karimi, T. Moos, and A. Moradi, "Exploring the Effect of Device Aging on Static Power Analysis Attacks," *UMBC Faculty Collection*, 2019.

[41] S. Keane, "Apple AirTags Apparently Hacked by Security Researcher," 2019, https://www.cnet.com/tech/mobile/apple-airtags-apparently-hacked-by-security-researcher/.

[42] S. Khan, N. Z. Haron, S. Hamdioui, and F. Catthoor, "NBTI Monitoring and Design for Reliability in Nanoscale Circuits," in *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*. IEEE, 2011, pp. 68–76.

[43] T. T. Kim and Z. H. Kong, "Impacts of NBTI/PBTI on SRAM V_{min} and Design Techniques for SRAM V_{min} Improvement," in *2011 International SoC Design Conference*. IEEE, 2011, pp. 163–166.

[44] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System Level Analysis of Fast, Per-core DVFS Using On-chip Switching Regulators," in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*. IEEE, 2008, pp. 123–134.

[45] W. Kim, M. S. Gupta, G.-Y. Wei, and D. M. Brooks, "Enabling On-chip Switching Regulators for Multi-core Processors Using Current Staggering," *Proceedings of the Work. on Architectural Support for Gigascale Integration*, 2007.

[46] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An Analytical Model for Negative Bias Temperature Instability," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, 2006, pp. 493–496.

[47] A. Limited, "ARM Cortex-A72 MPCore Processor Technical Reference Manual," 2016, https://developer.arm.com/documentation/100095/0003/.

[48] A. limited, "Arm Cortex-A78 Core Technical Reference Manual," 2021, https://developer.arm.com/documentation/102160/latest.

[49] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "Armageddon: Cache Attacks on Mobile Devices," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 549–564.

[50] T. Liu, C. C. Chen, J. Wu, and L. Milor, "SRAM Stability Analysis for Different Cache Configurations due to Bias Temperature Instability and Hot Carrier Injection," in *Intl. Conference on Computer Design*, ser. ICCD, Oct. 2016, pp. 225–232.

[51] X. Liu, C. Qian, W. G. Hatcher, H. Xu, W. Liao, and W. Yu, "Secure internet of things (iot)-based smart-world critical infrastructures: Survey, case study and research opportunities," *IEEE Access*, vol. 7, pp. 79 523–79 544, 2019.

[52] R. Maes and V. v. d. Leest, "Countering the effects of silicon aging on SRAM PUFs," in *Intl. Symposium on Hardware-Oriented Security and Trust*, ser. HOST, May 2014, pp. 148–153.

[53] J. Mahmod and M. Hicks, "Sram has no chill: exploiting power domain separation to steal on-chip secrets," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 1043–1055.

[54] A. Maiti, L. McDougall, and P. Schaumont, "The Impact of Aging on an FPGA-Based Physical Unclonable Function," in *Intl. Conference on Field Programmable Logic and Applications*, ser. FPL, Sep. 2011, pp. 151–156, star.

[55] A. Maiti, V. Gunreddy, and P. Schaumont, "A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions," in *Embedded systems design with FPGAs*. Springer, 2013, pp. 245–267.

[56] C. Marchand, A. Aubert, L. Bossuet *et al.*, "On the Security Evaluation of the ARM TrustZone Extension in a Heterogeneous SoC," in *2017 30th IEEE International System-on-Chip Conference (SOCC)*. IEEE, 2017, pp. 108–113.

[57] B. Marr, "Smart Dust is Coming. Are You Ready," *Accessed August 2023*, vol. 30, 2018, https://www.forbes.com/sites/bernardmarr/2018/09/16/smart-dust-is-coming-are-you-ready/?sh=42c415765e41.

[58] J. McMahan, W. Cui, L. Xia, J. Heckey, F. T. Chong, and T. Sherwood, "Challenging On-chip SRAM Security with Boot-state Statistics," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 101–105.

[59] Microchip Technology Inc, "SAM L10/L11 Family: Ultra Low-Power, 32-bit Cortex-M23 MCUs with TrustZone, Crypto, and Enhanced PTC," 2020.

[60] Microchip Technology Inc., "World-Class, Award-Winning SAM L10 and SAM L11 Microcontroller Family," 2021, https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/32-bit-mcus/sam-32-bit-mcus/sam-l/sam-l10-l11.

[61] NXP Semiconductor, https://www.nxp.com/docs/en/application-note/AN13037.pdf, accessed: 2023-7-15.

[62] NXP Semiconductors, "UM11126:LPC55S6x/LPC55S2x/LPC552x User manual," 2019.

[63] ——, "LPC55S6x: High Efficiency Arm® Cortex®-M33-Based Microcontroller Family," 2021, https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc5500-cortex-m33/high-efficiency-arm- cortex-m33-based-microcontroller-family:LPC55S6x.

[64] J. Pabel, "FrozenCache Mitigating Cold-boot Attacks for Full-Disk-Encryption Software," in *27th Chaos Communication Congress*, 2010.

[65] R. Pi, "Raspberry pi 4 model b specifications." [Online]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf

[66] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "VoltJockey: Breaching TrustZone by Software-Controlled Voltage Manipulation Over Multi-Core Frequencies," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 195–209.

[67] M. T. Rahman, D. Forte, X. Wang, and M. Tehranipoor, "Enhancing Noise Sensitivity of Embedded SRAMs for Robust True Random Number Generation in SoCs," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. IEEE, 2016, pp. 1–6.

[68] Raspberry Pi Foundation, "Boot Sequence," 2021, https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/bootflow.md.

[69] ——, "Raspberry Pi 3 Model B (v1.2)," 2021, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[70] Raspberry Pi Ltd, "Raspberry pi 4 model B specifications –," https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/, accessed: 2023-1-10.

[71] D. Rath, "Open On-Chip Debugger: Free and Open On-Chip Debugging, In-System Programming and Boundary-Scan Testing," 2021, http://openocd.org/about/.

[72] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos, "Flip Feng Shui: Hammering a Needle in the Software Stack," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 1–18.

[73] RICHTEK, "2.7MHz 3A Step-Down Converter with I^2C Interface," 2013, https://www.richtek.com/assets/product_file/RT8088A/DS8088A-00.pdf.

[74] A. Roelke and M. R. Stan, "Attacking an SRAM-Based PUF through Wearout," in *IEEE Computer Society Annual Symposium on VLSI*, ser. ISVLSI, Jul. 2016, pp. 206–211.

[75] ——, "Attacking an SRAM-based PUF Through Wearout," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 206–211.

[76] K. Ryan, "Hardware-backed heist: extracting ecdsa keys from qualcomm's trustzone," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 181–194.

[77] A. S. Sedra, D. E. A. S. Sedra, K. C. Smith, and K. C. Smith, *Microelectronic circuits*. New York: Oxford University Press, 1998.

[78] N. Sehatbakhsh, H. Hong, B. Lazar, B. Johnson-Smith, O. Yilmaz, M. Alam, A. Nazari, A. Zajic, and M. Prvulovic, "Syndrome: Spectral analysis for anomaly detection on medical iot and embedded devicesexperimental demonstration," in *Hardware Demo at IEEE International Symposium on Hardware Oriented Security and Trust (HOST*, 2017.

[79] N. Semiconductors, "LPC55Sxx debug authentication," 2021, https://www.nxp.com.cn/docs/en/application-note/AN13037.pdf.

[80] D. Sengupta and S. S. Sapatnekar, "Estimating circuit aging due to bti and hci using ring-oscillator-based sensors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1688–1701, 2017.

[81] SiFive, Inc., "SiFive U54-MC manual," https://simfive.cdn.prismic.io/sifive%2Fdc4980ff-17db-448b-b521-4c7ab26b7488_sifive+u54-mc+manual+v19.08.pdf.

[82] Silicon Labs, "EFM32 Wonder Gecko Family EFM32WG Data Sheet," 2012, https://www.silabs.com/documents/public/data-sheets/efm32wg-datasheet.pdf.

[83] S. Skorobogatov, "Low Temperature Data Remanence in Static RAM," University of Cambridge, Computer Laboratory, Tech. Rep., 2002.

[84] S. P. Skorobogatov, "Semi-invasive Attacks: a New Approach to Hardware Security Analysis," 2005.

[85] STMicroelectronics, "Ultra-low-power Arm® Cortex-M33 32-bit MCU+TrustZone®+FPU, 165DMIPS, up to 512KB Flash, 256KB SRAM, SMPS, AES+PKA," 2020, https://www.st.com/resource/en/datasheet/stm32l562ce.pdf.

[86] A. Tang, S. Sethumadhavan, and S. Stolfo, "{CLKSCREW}: Exposing the Perils of Security-oblivious Energy Management," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 1057–1074.

[87] TestEquity, "Model 123H Temperature/Humidity Chamber," 2020, https://www.testequity.com/category/Environmental-Chambers-Ovens/Temperature-Humidity-Chambers/TestEquity-123H-\Temperature-Humidity-Chamber-North-America-Version-17267-1.

[88] M. Tilo, S. Michael, and F. C. Freiling, "Frost: Forensic Recovery of Scrambled Telephones," in *International Conference on Applied Cryptography and Network Security*, 2014.

[89] T. Tuan, T. Strader, and S. Trimberger, "Analysis of Data Remanence in a 90nm FPGA," in *2007 IEEE Custom Integrated Circuits Conference*. IEEE, 2007, pp. 93–96.

[90] E. I. Vatajelu, G. Di Natale, and P. Prinetto, "Towards a Highly Reliable SRAM-based PUFs," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 273–276.

[91] M. Weiser, R. Gold, and J. S. Brown, "The Origins of Ubiquitous Computing Research at PARC in the Late 1980s," *IBM systems journal*, vol. 38, no. 4, pp. 693–696, 1999.

[92] H. Williams, A. Lind, K. Parikh, and M. Hicks, "Silicon Dating," *arXiv preprint arXiv:2009.04002*, 2020.

[93] G. I. Wirth, R. da Silva, and B. Kaczer, "Statistical Model for MOSFET Bias Temperature Instability Component Due to Charge Trapping," *IEEE Transactions on Electron Devices*, vol. 58, no. 8, pp. 2743–2751, 2011.

[94] I. Xilinx, "Zynq-7000 SoC and 7 Series Devices Memory Interface Solutions," 2018, https://www.xilinx.com/support/documentation/ip_documentation/mig_7series/v4_2/ds176_7Series_MIS.pdf.

[95] S. Zeitouni, Y. Oren, C. Wachsmann, P. Koeberl, and A.-R. Sadeghi, "Remanence Decay Side-channel: The PUF Case," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1106–1116, 2015.

[96] F. Zhang, S. Yang, J. Plusquellic, and S. Bhunia, "Current Based PUF Exploiting Random Variations in SRAM Cells," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 277–280.

[97] N. Zhang, K. Sun, W. Lou, and Y. T. Hou, "Case: Cache-assisted Secure Execution on ARM Processors," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 72–90.

[98] N. Zhang, K. Sun, D. Shands, W. Lou, and Y. T. Hou, "TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices," *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 980, 2016.

Appendix A.
Meta-Review

A.1. Summary

The paper shows how externally-induced accelerated aging of several SoCs allows secrets in TrustZone to be leaked when an attacker has physical access to the chip.

The authors observe that SRAM cells are subject to burn-in through the aging process, and that with artificially accelerated aging, they can effectively burn-in on-chip secrets into the SRAM cells on-demand, and subsequently exfiltrate them to steal the secrets.

The authors introduce three exfiltration attacks to demonstrate feasibility: extracting an AES key from TrustZone, extracting proprietary firmware from TrustZone, and extracting other secrets from processor caches.

A.2. Scientific Contributions

- Identifies an Impactful Vulnerability
- Provides a Valuable Step Forward in an Established Field
- Establishes a New Research Direction

A.3. Reasons for Acceptance

1) Identifies an impactful vulnerability: The authors show a new method for leaking secrets out of TrustZone enclaves.
2) The paper provides a valuable step forward in an established field: Malicious/directed SRAM aging was previously used only for attacks on availability and integrity; this work shows how these attacks can target confidentiality as well.
3) The paper establishes a new research direction, by creating a discussion on defenses against secret exfiltration by SRAM aging.