SRAM Has No Chill: Exploiting Power Domain Separation to Steal On-Chip Secrets

Jubayer Mahmod
jubayer@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Matthew Hicks
mdhicks2@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

ABSTRACT

The abundance of embedded systems and smart devices increases the risk of physical memory disclosure attacks. One such classic non-invasive attack exploits dynamic RAM's temperature-dependent ability to retain information across power cycles—known as a cold boot attack. When exposed to low temperatures, DRAM cells preserve their state for a short time without power, mimicking non-volatile memories in that time frame. Attackers exploit this physical phenomenon to gain access to a system's secrets, leading to data theft from encrypted storage. To prevent cold boot attacks, programmers hide secrets on-chip in Static Random-Access Memory (SRAM); by construction, on-chip SRAM is isolated from external probing and has little intrinsic capacitance, making it robust against cold boot attacks.

While it is the case that SRAM protects against traditional cold boot attacks, we show that there is another way to retain information in on-chip SRAM across power cycles and software changes. This paper presents *Volt Boot*, an attack that demonstrates a vulnerability of on-chip volatile memories due to the physical separation common to modern system-on-chip power distribution networks. *Volt Boot* leverages asymmetrical power states (e.g., on vs. off) to force SRAM state retention across power cycles, eliminating the need for traditional cold boot attack enablers, such as low-temperature or intrinsic data retention time. Using several modern ARM Cortex-A devices, we demonstrate the effectiveness of the attack in caches, registers, and iRAMs. Unlike other forms of SRAM data retention attacks, *Volt Boot* retrieves data with **100%** accuracy—without any complex post-processing.

CCS CONCEPTS

• **Hardware → Static memory**; • **Security and privacy → Embedded systems security**; **Hardware attacks and countermeasures**.

KEYWORDS

SRAM attack, power domain separation, cold boot

ACM Reference Format:
Jubayer Mahmod and Matthew Hicks. 2022. 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 (ASPLOS '22), February 28 – March 4, 2022, Lausanne, Switzerland.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3503222.3507710

1 INTRODUCTION

An increasingly connected world makes us dependent on computing devices that handle a wide range of security- and privacy-critical operations. On the personal side, we use smartphones and watches to manage bank transactions and store identity information. On the industrial and government side, embedded devices monitor remote system operations and feed data critical to the industrial processes and national defense. Physical access to these devices leads to a wide range of security exploits, including impersonation, proprietary software cloning, and infiltration and exploitation of industrial and defense infrastructures.

The most common approach to prevent data loss from a system's non-volatile memory is to encrypt it using full disk encryption methods, such as Bit-locker [28] and VeraCrypt [22]. These encryption methods protect user data using a password or PIN so that even if a device is lost or stolen, the non-volatile memory remains inaccessible to an attacker. An encrypted disk demands user authentication whenever there is a reboot, providing adequate security even if an attacker removes it physically from a system and attempts to access its content from another machine.

Password-protected disk encryption methods force the attackers to exploit other types of memories, such as DRAMs. Halderman *et al.* show how an attacker gains access to a disk encryption key by *cold booting* a system and dumping its main memory [17]. In this attack, the authors use low temperature ($-50°C$) to 'freeze' the data in DRAM cells so that even if the memory is *out of power* for a short time, it retains its logic states. Once 'frozen', an attacker physically removes and inserts the victim DRAM in another machine to run forensics on the dumped memory image. From this point, it is trivial to post-process and extract security- and privacy-critical information from a system's main memory. While this attack is practical for larger devices where DRAM is removable (*e.g.,* laptops), it poses a few technical challenges for mobile and other embedded devices. In an embedded device, the memory chips and processors are soldered on the PCBs, making it difficult to remove them from a system. *FROST* [31] overcomes this challenge by allowing device reset to factory default—preserving DRAM's content while the device boots from another media.

To defend devices from cold boot attacks, researchers proposed numerous methods where sensitive code and data remain encrypted in on-chip memory; decryption occurs only in the on-chip memories such as caches [30, 44]. Since SoCs already contain large enough on-chip storage to hold keys and cryptographic states, executing software on these memory requires no additional hardware. For example, *TRESOR* uses x86 debug registers to store sensitive AES states without leaking security-critical information to off-chip memory [30]. Researchers extend this idea to ARM devices where the CPU fetches the encrypted software and data to the on-chip memories before decryption and execution [8, 9, 13, 14, 39, 44]. These works advocate fully on-chip execution for cryptographic operations because attacking a processor's internal memory is expensive and demands sophisticated attack methods, such as decapsulation. This paper evaluates the security of on-chip computation schemes, specifically under the cold boot attack model. We empirically show that these methods are secure from the traditional cold boot attack [17], but the power domain separation common to modern SoCs allows us to create a similar effect, which exposes on-chip secrets to physical attackers.

On-chip memories are mostly SRAMs and built into the processor die itself (*e.g.,* caches), which alone makes it much more secure than off-chip memories against physical attackers (*e.g.,* probing attacks [18, 44]). Previous work shows that SRAM partially retains its state for a few milliseconds in extremely low temperature ($< -110°C$) [2], exposing a system to such low-temperature risks damaging its other components, such as the battery—not to mention the challenge of creating such a low temperature. In addition, attacking embedded devices requires a physical power disconnect which is typically much longer than SRAM's data retention time.[1]

This paper presents *Volt Boot*, a method that executes a physical memory disclosure attack on SRAM-based on-chip memories by exploiting power supply domain separation in SoCs. Different physical blocks in an SoC require a specific voltage level to meet performance demands while reducing power requirements. These blocks are divided into different domains that need external active and passive circuit components, such as a Power Management Integrated Circuit (PMIC), capacitors, and inductors. The circuit design of these domains allows fine-grained control of the system's total energy budget and performance. From a power management perspective, these domains are independent and allow full power down at runtime when not needed by the system. We uncover an attack vector in such design choice that is exploitable for a cold-boot-style attack, eliminating the dependency on low temperature and an SRAM's intrinsic data retention time.

In summary, this paper makes the following technical contributions:

- We demonstrate that traditional cold-boot-style attacks, which leverage low-temperature-induced data retention, are ineffective on embedded SRAMs (§3).
- We uncover a new attack vector that exploits *power domain separation* of modern SoCs and show that such *architectural design choice can be weaponized* for breaking fully-on-chip cryptography (§5).
- Using *three* commercially available Cortex-A profile devices, we demonstrate the attack by retrieving data from caches (§7.1), CPU registers (§7.2), and iRAMs (§7.3). Provided that an application runs from internal memory (*e.g.,* cache) undisturbed by operating system's background process, we retrieve memory image with **100% accuracy**.[2]
- We survey potential countermeasures and their trade offs (§8).

2 BACKGROUND

Embedded Static Random-Access Memory (SRAM) serves a variety of roles in modern computing systems including, storing microarchitectural states, buffering address translations, a cache for instructions and data, and as the main memory for many lightweight devices. Their embedded nature naturally protects from physical-level attacks that rely on interposing the interface between memory and the computation core (e.g., bus probing and cold boot attacks). This section discusses the fundamentals of on-chip SRAM, on-chip power networks, and the rationale behind on-chip cryptography.

2.1 On-Chip SRAM

SRAM is the building block of volatile internal memories, such as caches, iRAM, registers, TLBs, and BTBs, making them one the most common memory in modern computing systems. Figure 1 illustrates a typical 6-transistor SRAM cell, which is composed of two inverters in a positive feedback configuration to hold a data bit. The cell's state is accessible through transistors N1 and N2, and these transistors provide access to the data bit (Q) and its complement (~Q), respectively. Unless a processor executes a read/write command, *Word Line* remains de-asserted with data stored in the cross-coupled structure formed by inverter ① and inverter ②.

SRAM is energy-efficient, fast, long-life, and self-refreshing; the only requirement for data retention is sufficient voltage from the power supply to maintain the positive feedback loop between the two inverters. The voltage required by an SRAM cell to retain the state is called its data retention voltage [20]. Data retention voltage is both process variation and data-dependent but is generally much lower than the threshold voltage of either inverter (i.e., the 'turn-on' voltage). Provided the voltage of a cell is more than or equal to its data retention voltage, the cell retains its state. Exploiting this property, modern processors dynamically scale down the voltage when the RAM is not actively accessed because it reduces the *energy leakage* through parasitic paths.

Unlike other types of memory, *direct access* (*i.e.,* direct software read/write) to many types of on-chip SRAM (e.g., instruction cache) is uncommon. However, most SoCs provide access to these internal memories through various methods to debug low-level memory errors; ARM allows RAMINDEX [23] and direct memory access [4] operations in ARMv8 architecture, while RISC V processors [38] allow memory-mapped access. For example, a Cortex-A72 processor provides access to 15 different internal RAMs, including caches, TLBs, and BTBs through its *cp15* co-processor interface.

[1]We define the intrinsic data retention time of an SRAM cell as the time it takes to discharge (and lose the data) when disconnected from power [35].

[2]The reported accuracy indicates data retention accuracy and not the information extraction accuracy. Exact information retrieval accuracy varies depending on the data present in an SRAM.

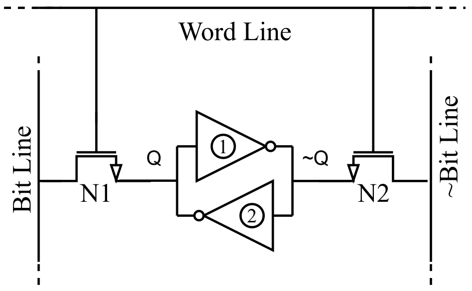


Figure 1: Typical 6-transistor SRAM cell.

2.2 Fully On-Chip Computation

Storing security-critical information in a DRAM (which is off-chip) is unsafe, motivating both industry and academic research to push for safer off-chip memory management schemes. Since Intel's 6th generation processor, all subsequent processors obfuscate data in the DDR3 and DDR4 DRAMs using session keys and pseudo-random numbers [29, 43]. On the academic side, researchers propose fully SoC-bound computations where sensitive information leaves the chip only when encrypted. Modern SoCs contain a reasonably large volatile memory. In most cases, it is inessential to encrypt and decrypt every transaction; software can store intermediate states on on-chip memories in plain text. This idea inspired numerous on-chip computation methods. *Sentry* [9] and *Copker* [15] uses iRAMs and caches as temporary memory to avoid exposing secrets in DRAM. *Cache-assisted Secure Execution (CaSe)* extends this idea by adding Trust-Zone support to a partially locked cache. The processor fetches encrypted software from main memory and stores it in a locked cache as plain text. From this point, the unencrypted software remains in the cache for the duration of execution. Similarly, *TRESOR* [30], *PRIME* [13], and *Security Through Amnesia* [39] use on-chip registers (*e.g.,* debug or multimedia) to implement cryptographic algorithms on-chip.

These methods essentially emulate a microcontroller's behavior in a large-scale application processor and reduce the attack surface to the borders of the chip itself. Given no unencrypted data is released off-chip, these methods provide strong security against the most sophisticated physical memory disclosure attacks, *e.g.,* cold boot. While these methods induce performance penalties on other applications by partially blocking hardware resources, our paper evaluates only their security aspects.

2.3 Power Domain Separation

SoCs contain a large number of circuit blocks with different analog characteristics. Because of performance and power efficiency constraints, these blocks operate under separate voltage domains. An SoC's Power Management Unit (PMU) manages voltage required for specific domains at runtime, depending on each domain's workload [1]. In a complex SoC, dozens of off-chip supply pins are connected to the power domains to regulate different analog circuit behavior, such as ground bounce, power-supply noise, and per-pin current supply. For this paper, we divide the SoCs power supply domains into three major areas (see Figure 2):

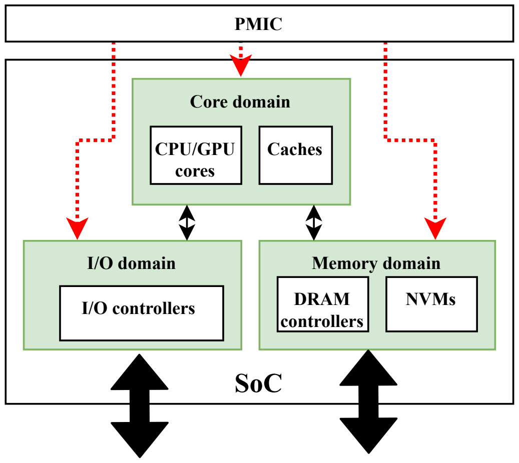


Figure 2: A simplified block diagram of a SoC's power domains. The PMIC is an external component that maintains a specific voltage level at each power domain supply pin.

- **Core power domain:** the processing elements are in this domain. For example, the ARM cluster in a multi-core SoC draws power from the core supply voltage domain. Apart from computing elements such as CPU extensions and GPUs, this domain supplies power to the L1 caches and their associated control circuitry.
- **Memory power domain:** this domain supplies power to the memories and their associated peripherals. Most SoCs manage main memory, non-volatile memory (*e.g.,* Flash), and L2/L3 caches with this power domain.
- **I/O power domain:** power for the I/O controllers and external peripherals is drawn from this domain.

SoC designers subdivide power domains into smaller logical blocks that allow for more fine-grain control of the different components within an SoC. For example, some processors allow powering down individual cache components in its L1 memory domain through software [23]. The domains are separated using power gating to balance energy consumption and performance of independent blocks at startup and runtime.

3 COLD BOOTING ON-CHIP SRAM IS INEFFECTIVE

DRAMs are typically separate IC packages externally connected through a memory bus, exposing these memory modules to different physical attacks, such as cold boot and bus probing. Given a system encrypts its code and data while 'at rest' in a disk and decrypts data while in volatile memories, capturing a live system and extracting the volatile memories leads to key disclosure, enabling secret exfiltration.

DRAM holds data as charge in a capacitor. Capacitors continually leak charge, which is why DRAM cells require periodic refresh operations to ensure data retention (typically every 64ms). In the absence of refresh, a DRAM cell's data retention time depends on the *time* it takes for capacitors to leak sufficient charge to change the digital value interpreted by sensing the voltage across the capacitor.

Table 1: Errors in d-cache data after a cold boot attack execution in a BCM2711 SoC. We compute a mean error for each core at different temperatures. The fractional Hamming distance between cache content after power cycle and cache's startup state is ~ 0.10, indicating no data retention.

Temperature	$0°C$ Recommended Min.	$-5°C$	$-40°C$ SoC's hard limit
Error	50.14%	50.06%	50.39%

Even without power, a just-refreshed DRAM cell will retain data for at least 64ms.

Temperature affects DRAM cell discharge rate; lower temperatures increase data retention time. Reducing the temperature of a memory device keeps the data in the memory for a short period even if the power is turned off, which is the fundamental idea behind the cold boot attack [17]. Both DRAM and SRAM partially retain their data across power cycles for a short time when the temperature is reduced below a certain level [2, 3, 17] due to their intrinsic capacitance (although DRAM retains for orders of magnitude more time due to its increased capacitance). Information retrieval accuracy of a *cold booted* SRAM cell depends on temperature, manufacturing technology, and power-off time. We discuss the technical hurdles that prevent a cold boot attack on SRAM as follows:

- **SRAM's placement:** As mentioned before, on-chip SRAM is tightly coupled with processing cores and are built into the processor die itself. Such placement of SRAM makes it inaccessible given the traditional physical attacker threat model. The *embedded nature* of SRAMs complicates the cooling process as well because an attacker needs to freeze the entire device. *Cold booting* this memory requires extreme low temperature and risks bricking of the system.
- **Short retention time:** We cannot launch a cold boot attack on embedded memories by resetting a device through software because a system easily prevents such attempts by purging residual memories as part of a core's power down sequence [7]. Disconnecting the power from a device is the only reliable way to prevent the system from executing any residual memory purging routines.
  An abrupt power disconnect from a device while executing critical security operations ensures target information remains in SRAM. To power cycle an embedded device, we must manually disconnect its power supply (*e.g.,* the battery). As the literature suggests, SRAM retains information for only a few milliseconds—even under extremely cold conditions, which is insufficient to execute a reboot by manually disconnecting power from a device [31]. In addition, SRAM is tightly coupled with the compute cores. An abrupt power disconnect draws energy from all parts of the SoC to the power-hungry processing elements, which accelerates the charge drain from SRAM's intrinsic capacitors. The combined effect of these SRAM-specific variables shortens the overall data retention time enough to make cold-boot-style attacks on SRAM infeasible.

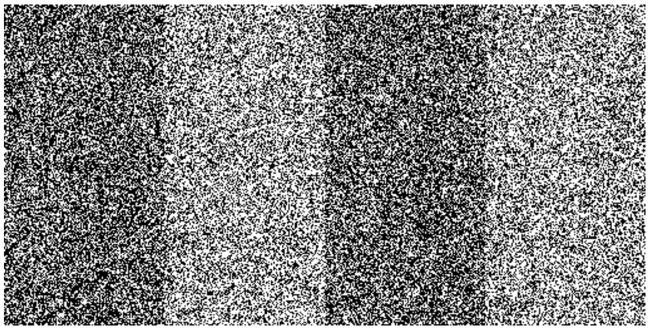


Figure 3: Data cache (L1) snippet ($WAY0 = 256 \times 512 = 16KB$) of a Cortex-A72 core when we disconnect the power for a few milliseconds at $-40°C$. An equal number of 1s and 0s in the cold-booted cache image indicates that the cache reset to its power-on state, *i.e.,* no data remained in the cache.

- **Effect of low-temperature:** Typically, systems turn off when the operating temperature crosses a certain threshold set by the manufacturers. Executing a cold boot attack on SRAM requires extremely low temperature ($< -110°$) [2], which is far beyond the operating limit of most devices. We reproduced a similar attack as *FROST* [31] in cache memory (SRAM, as opposed to the DRAM targeted in the original attack) of a Raspberry Pi 4 (a quad-core cortex-A72 device) [23] to study cold temperature data retention of its embedded SRAM. We load bare-metal software to populate both the d-cache and i-cache of each core and extract the cached data in a binary image. Then the device undergoes static cooling in a *TestEquity* thermal chamber [41] for an hour to stabilize the core temperature. We power cycle the device for a few milliseconds and extract the cache (§6) data to compare it to previously-stored binaries. Table 1 lists the *mean* mismatch between post-reboot retrieved cache and pre-stored binary for each core at different temperatures. The information retrieval errors indicate almost no data retention even at $-40°C$. Note SRAMs boot up into random states where approximately 50% of the bits are 1s. Figure 3 illustrates a snippet of d-cache to provide a clear picture of the post-power-cycle state of an SRAM.

4 ATTACK MODEL

This paper shows how power domain separation and exposed memory power supply pins induce artificial SRAM state retention, leading to a similar effect as a cold boot attack *without* requiring temperature control. Naturally, our attack model is based on the cold boot attack [17] where an attacker has physical access to a device. The most significant modifications to the original cold boot threat model are the location and type of the memory. Our target is SRAM embedded within the core of a device, which prevents direct access to the memory contents without damaging the chip. This threat model is consistent with the threat model presented in the *FROST* [31], *Sentry* [9], and *CaSE* [44] systems. These papers consider a threat model where an attacker captures a device (*i.e.,* lost or stolen) that is protected against memory disclosure attacks using a lock screen and non-volatile memory encryption. Then the attacker boots the victim device from a media, *e.g.,* boot ROM or USB, to dump the uninitialized volatile memory image.

Our threat model extends to headless embedded devices that collect, store, and transfer sensitive information in an unsupervised environment. A number of system protection methods exist where devices are permanently locked from programming or software updates. The behavior of such devices resembles an Application-Specific Integrated Circuit (ASIC), and we consider these systems out of the scope of our threat model.

5 *VOLT BOOT*

Volt Boot is an attack that exploits an SoC's power domain separation to induce cross-power-cycle data retention in embedded SRAM. In this section, we show circuit design choices made for power and performance reasons lead to a new cold-boot-style attack, but with increased precision and fewer requirements than traditional low-temperature-based data retention attacks.

5.1 Inducing Data Retention

SoCs need external pins to supply specific voltage to optimize performance and efficiently use energy under fluctuating loads. Usually, a PMIC supplies power to each domain in a particular order to bring up the board from reset. Figure 4 illustrates a typical power management IC used in modern SoCs. Generally, LDOs supply power to the domains where voltage fluctuation is limited, whereas domains with high load fluctuation (and dynamic voltage and frequency scaling) use switching regulators to save energy via heat loss (e.g., BUCK converters on the right side of the PMIC). When a power domain of an SoC draws a large current under the demand of software/hardware, the parasitic inductance of the board and the package drop the voltage at the supply lines; this is called droop. To counter droop, the supply voltage pins are extended out of the SoC to connect capacitors so that they 'absorb' the current surge, keeping supply voltage closer to nominal. Regardless of the type of regulator used in a power domain, the pins connected to the PMIC require passive components to filter out noise generated during load fluctuation.

Each power domain is power-gated to allow independent control at startup and run time. Our observation is that if we externally maintain a steady voltage to the pin that supplies power to a target memory domain, it retains data—even while the rest of the system undergoes a power cycle. We maintain the domain voltage level from an external voltage probe while disconnecting the main supply line to the PMIC. That is, *Volt Boot* artificially creates SRAM data retention across a power cycle using a voltage probe in a PCB's test pad or bare passive component's lead that is connected to a target memory domain.

As discussed in Section 3, cold booting is an impractical way of attacking on-chip SRAMs. Even if it is possible to retain information across power cycles, the extracted data is erroneous because SRAM's charge leakage follows a normal distribution, and so some cells will lose their data even for a short power disconnect [34]. Note that SRAM cells are bistable, which makes it harder to look for keys using the algorithm proposed in the original cold boot attack [17]. Since our attack involves no other physical variables (*e.g.,*

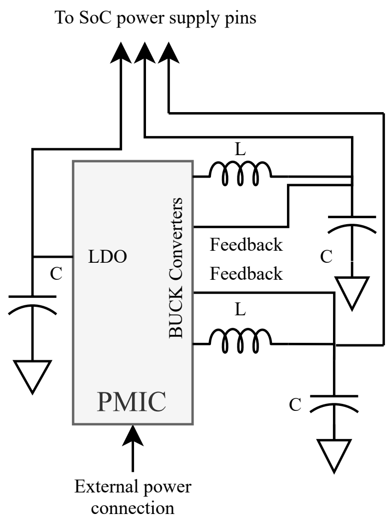


Figure 4: Typical power supply system. The LC combination supply lines are for switching regulators (right side), whereas the decoupling capacitor (left side) filters out noise during fluctuating load driven by LDOs.

temperature), memory discharge rate or manufacturing technology node is irrelevant and we achieve 100% data recovery.

5.2 Attack Enablers

While power domain separation serves as the basis for our attack, a combined effect of multiple aspects and attributes of a system enables *Volt Boot*. We identify each attack enablers as follows:

5.2.1 Ubiquity of SRAM. SRAM is available in every computing device ranging from resource-constrained microcontrollers to server-class processors. We are able to induce artificial SRAM state retention in any SoC that has separate SRAM and compute core power domains.

5.2.2 Internal RAMs Stores Data in Plain-text. As mentioned in Section 2, cryptography application developers consider on-chip memories safer compared to external memories. As a result, unlike external DRAMs, scrambling or encrypting SRAM's data is uncommon in commodity processors. Thus, access to data residing in on-chip SRAM guarantees a plain-text version of the information, even if external memories implement scrambling or encryption.

5.2.3 Domain Specific Exposed Power Supply Pins. As discussed above, performance, energy efficiency, and die area reduction are the primary reasons to expose domain-specific voltage pins out of an SoC; some of these pins are connected to the embedded SRAM. By construction, data remains error-free in SRAM as long as its supply voltage remains above its data retention voltage (§2). Note, this voltage is well below the nominal supply voltage of the power domain the cells are connoted to. The power domain separation through power gating methods and dedicated pins exposed from the SoCs let us shut down the entire system while the small portion that holds sensitive data 'alive'.

Table 2: Evaluated platforms and SoCs.

System on Chip	CPU core	Board	SRAM	Manufacturer
BCM2711	Quad-core Cortex-A72	Raspberry pi 4 Model B [12]	`L1I: 48KB, L1D: 32KB, L2: 1MB`	Broadcom
BCM2837	Quad-core Cortex-A53	Raspberry pi 3 Model B [11]	`L1I: 32KB, L1D: 32KB, L2: 512KB`	Broadcom
i.MX535 [32]	Cortex-A8	i.MX53 Evaluation board	`L1I:32KB, L1D: 32KB, L2: 256KB, iRAM: 128KB`	NXP Semiconductor

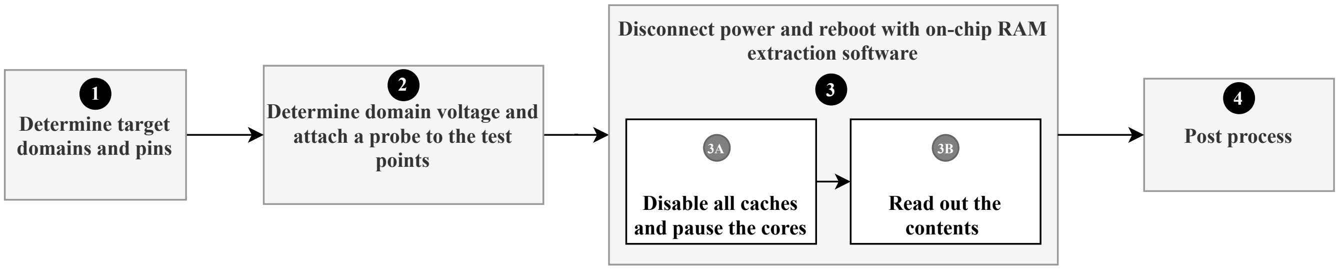

1
Determine target domains and pins
2
Determine domain voltage and attach a probe to the test points
Disconnect power and reboot with on-chip RAM extraction software
3
3A
Disable all caches and pause the cores
3B
Read out the contents
4
Post process

Figure 5: Attack flow.

5.2.4 No Default RAM Reset Hardware. SRAM stays at an uninitialized state after booting-up because of two main reasons: ① some SRAMs are large (>1MB) compared to the other on-chip resources, and resetting such large memory by iterating over the entire address (usually line by line for caches) space reduces boot speed significantly, and ② SRAM's startup state has numerous security applications, such as PUF [36] and TRNG [19]. Note that cleaning and invalidating a cache at the boot phase does not erase the contents; these operations set the invalid bits to prevent a cache hit, but the data remains unchanged. The co-processor interface still allows reading out the cache contents from a proper exception level (`EL3` for ARM devices). Therefore, initializing cache lines using a software interface is currently the only means to reset the power-on state of L1 caches, which needs the execution of `DC ZVA` instructions for every line [4, 23]. The purpose of this instruction is to allow initializing a large block of memory in the cache for a particular data structure without writing zeros in the external memories. Note cleaning/invalidation instructions apply to both instruction and data caches, but *resetting instructions* are exclusive to d-caches.

6 ATTACK EVALUATION

An SoC's domain-separated power management architecture allows us to supply voltage to a target memory through exposed pins and keep part of the chip active (i.e., retaining state) while the rest of the system resets. The actual method used to access embedded SRAM varies depending on the targeted SoC, but devices' power supply methods are very similar at a high level. We evaluate *Volt Boot* using ARM devices from different vendors, and to explore the generality of the attack, we choose devices that span a broad range of applications. For example, Raspberry Pis represents a wide array of systems, ranging from headless embedded devices common in IoT applications to systems capable of running a full-fledged operating system. To expand the targeted memory types, we include an SoC designed for multimedia applications, because it contains iRAM. Table 2 lists the specification of the evaluation platforms. The SoCs in these systems draw power from three different power management devices, and we observe similar circuit design choices for the off-chip passive components (see Figure 4).

We provide a stable voltage to each SoC's target power domain using a bench power supply with >3A current driving capability. While attaching a probe at the same voltage level draws only a few milliamps, an abrupt power disconnect from the compute core supply line of the PMIC spikes the current momentarily on the target power domain. Such a current surge drops the voltage below the data retention voltage of SRAM (§2), leading to errors in the extracted information. Therefore, a power supply capable of supplying sufficient current is essential when the target memory domain also supplies power to the CPU core(s).

6.1 Attack Execution Steps

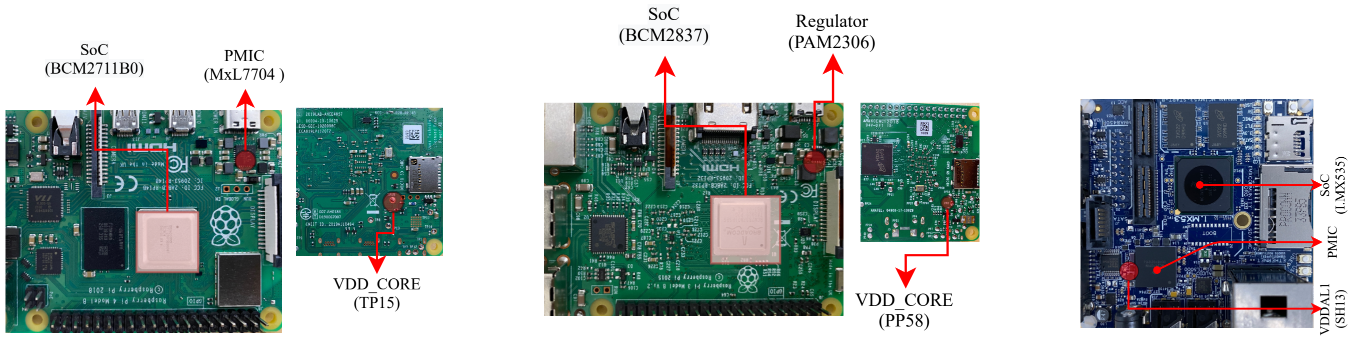
In this section, we discuss how to execute an attack on SoCs (see Figure 5 for summary).

(1) **Identifying target domains and their associated pins:** Once we identify a target device, the first step is to identify the pins that supply SRAM with power. In most cases, it is impossible to locate a specific pin on the circuit board, because SoC chips use advanced packaging, such as BGA. However, it is not essential to find exact pins in an SoC package as supply pins are connected to passive components (*e.g.,* decoupling capacitors) or circuit-board-level test pads, which tend to be located near the PMIC (§5). The layout of passive components follows a typical pattern illustrated in Figure 4. For our evaluation platforms, we list the test points and target domain's pin names in Table 3 and present them in a visual form in Figure 6.

(2) **Attaching a voltage probe:** We measure the nominal voltage at the pin(s) and attach an external power supply probe at the same voltage level. The power source needs to supply sufficient current so that the level stays the same when we turn off the device's main power; otherwise, we risk losing the data. As an example, a Raspberry Pi 4 draws current

Table 3: *Volt Boot* evaluation platforms, test pads, and their nominal voltages.

Boards	PCB test pads to probe	Nominal voltage	Target memories	Power domains
Raspberry Pi 3	PP58	1.2V	L1D, L1I, registers	Core (*VDD_CORE*)
Raspberry Pi 4	TP15	0.8V	L1D, L1I, registers	Core (*VDD_CORE*)
i.MX53	SH13	1.3V	iRAM	Memory (*VDDAL*1)



(a) **(b)** **(c)**

Figure 6: Pictures of our evaluation platforms (a) Raspberry pi 4, (b) Raspberry Pi 3, and (c) i.MX535, showing the test points we attach our voltage probe to.

through the test pad TP15 when we attach an 800mV probe. The current varies between 400mA to 600mA depending on software workload. When the SoC's main supply line (powered through a USB C) is disconnected abruptly, the cores draw power from the attached probe. The probe maintains the voltage level even if the cores demand a momentary current surge. The current consumption drops to 8mA after a few microseconds, and the memory domain stays in this retention state indefinitely.

(3) **Power cycling and booting the system:** Once the external probe is in place, we disconnect the device from the main power source while our voltage probe keeps the target SRAM active.

A system's boot-up method after power disconnect varies. Some systems allow booting up from alternative media only if the user data from the disk is erased, whereas some devices boot internally without needing any external boot media. We emulate this behavior by booting up the Raspberry Pis from another media through a USB mass storage device. We write a post-reboot data extraction program that performs the following tasks:

(A) Reduce contamination on the SRAM's retained data during boot-up by avoiding storing data to it (either explicitly or implicitly).

(B) Exfiltrate data from the SRAM to other memory (e.g., Flash, DRAM, or a debugger) for post-processing.

The cache extraction software executes `CP15` instructions and reads out the *data register interface* of the caches to general-purpose CPU registers. Cache access requires read-/write to system registers. For processors with out-of-order execution capability, we must use appropriate data and instruction synchronization barriers. For example, Cortex-A72 processors uses `SYS #0, c15, c4, #0, <xt>` instruction to execute RAMINDEX operation (cache access request to `CP15` co-processor). Data and instruction synchronization barrier instructions `DSB SY` and `ISB`, respectively, must follow this instruction before reading the cache data output register interface. A set of general load/store instructions moves the data from the general-purpose CPU registers to DRAM for further processing.

We directly dump the iRAM's through the debug interface, because i.MX535 requires no external firmware support for booting up. Thus, we connect a `JTAG` probe and directly read out the processor's (Cortex-A8) iRAM contents.

(4) **Analysing the memory contents:** Depending on the target SRAM and the objective, an attacker needs to adapt post-processing. Since *Volt Boot* reads out the memory without any error, the noise source in a successful attack is the *dynamic behavior* of software and its effects on the data stored in embedded SRAM. For example, error-free key extraction from a cache memory depends on the processing core's workload and other background processes.

6.2 How Much Memory is Accessible to an Attacker?

At startup, the CPU uses some types of embedded SRAM before even an attacker has access to those memories. What percentage of memory is available after SoC boot-up depends on the target memory type of an SoC. To find the accessible SRAM, we execute bare-metal software that populates a target memory with predefined patterns. The bare-metal setup allows us to calculate the effect of the CPU's boot phase on internal memories, avoiding dynamic behavior, such as cache eviction. Once the software loads the data/instruction in the target memory, we execute the steps discussed above (Section 6.1).

Our experiments on the L1 caches of the BCM2711 and BCM2837 indicate no clobbering from the initial boot phase. This result aligns with the fact that the L1 cache in these SoCs is software-enabled. Therefore, when an SoC releases the control to external software, an attacker simply never activates the cache. That is, an attacker has access to the full image of the L1 caches. Note that, these Broadcom devices contain a built-in video core that shares the L2 cache with ARM CPU cores. At startup, video-core initiates system initialization using pre-compiled binaries that clobber L2 cache contents, preventing any post-reboot data access.

The i.MX535 has similar behavior for caches, but boot ROM uses part of the iRAM as scratchpad memory before initializing the DRAM controllers. That is, the CPU resets part of iRAM before allowing any debug connection or software execution. Such a boot method is standard among Cortex-M devices; they usually clobber $2KB$ SRAM (main memory) at the boot phase [21, 27]. We experimentally show that approximately 95% of an i.MX535's iRAM is available to an attacker.[3]

7 ATTACK EXECUTION IN DIFFERENT MEMORIES

A successful cold boot attack depends on many variables, including temperature and an SRAM's intrinsic data retention time (mostly determined by the manufacturing technology). An attacker must turn off the device abruptly to prevent memory corruption or any defensive wiping. The next step is to reduce the temperature of the device; the lower the temperature, the more accurate the extracted information (given some fixed amount of time without power). Then, the victim device needs to boot. The most common way to execute an abrupt power-off in a commercial device is to remove the battery or power connection from the device. These operations require more than a few hundred milliseconds, far too long for SRAM data retention [2].

Volt Boot is a non-invasive memory disclosure attack that eschews temperature for voltage-induced cross-power-cycle data retention for SRAM. The ultimate result of such data retention resembles a cold-boot-style attack with higher accuracy—without exposing a device to unrealistic low temperatures. *Volt Boot* exploits power domain separation in modern SoCs, eliminating variables like data retention time and temperature. We execute three example attacks using the devices listed in Table 2. Our proof-of-concept attacks empirically demonstrate the vulnerability of computation methods that store secrets as plaintext in caches, registers, and iRAMs.

7.1 Attacking Caches

Large caches are common in SoCs, and numerous on-chip computation methods, for example, *CaSE* [44], exploit a portion of such

[3]A viable defense could hide secrets in the 5% of memory overwritten during the initial boot phase, although we did not verify if this entire portion of SRAM gets overwritten.

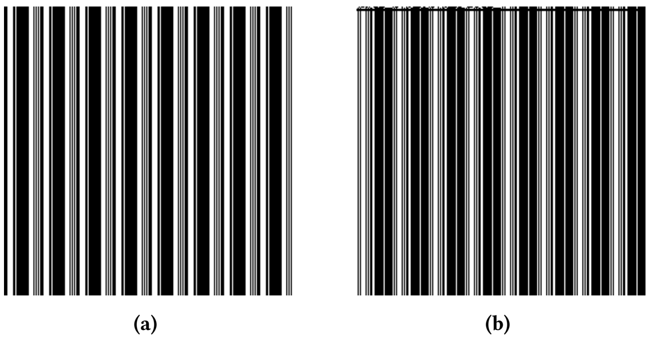


Figure 7: Snapshots of i-cache after attacking bare-metal software in (a) BCM2711 and (b) BCM2837 SoCs. Uninitialized cache cells power on into random sate (see Figure 3), but when we execute *Volt Boot* attack, instructions stay in the i-cache across power cycles.

caches to use it as the primary software execution memory (instead of external DRAM). We assess the security of such schemes under the threat model described in Section 4.

7.1.1 Attacking Caches with Bare-Metal Software. In this scenario, we study how *Volt Boot* attacks a device that runs bare-metal software, and emulate common embedded systems that are designed for specialized applications, such as monitoring and collecting data. We write a bare-metal program that enables the caches, then executes `NOP` instructions in all four cores. This allows us to quantify precisely how much of the software *Volt Boot* extracts. To tightly control the execution environment, we write the software in assembly (i.e., `aarch64`).

We execute the attack steps (Section 6.1) on Raspberry Pis and compare the cache content to the ground truth machine code.[4] Figure 7 provides a visual representation of the the cached content after we execute the attack (compare it to Figure 3). As expected, the accuracy of data retention in the caches is 100% in all four cores of both devices.

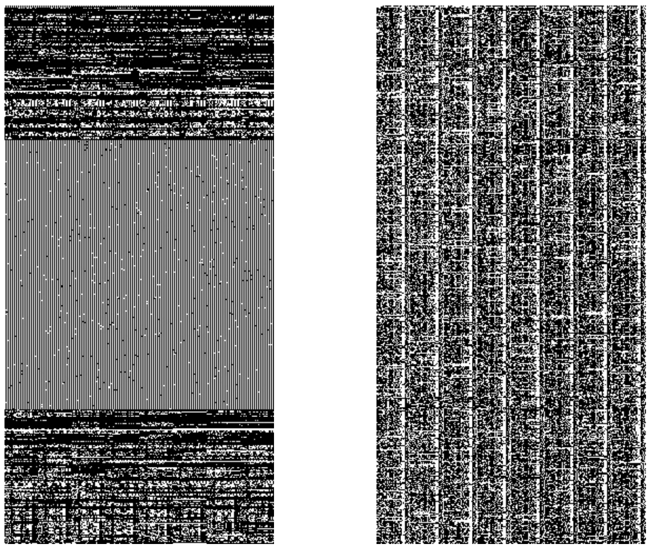
7.1.2 Attacking Caches with an OS. We show how *Volt Boot* leaks data from a user application running on a system for general-purpose operations—the Linux kernel. We write an application that stores a specific pattern (`0xAA`) in a large data structure and reads it back. While executing, we carry out the attack steps from Section 6.1 and plot the post-attack snapshots of the respective d-cache in Figure 8. The d-cache contains the expected pattern (*i.e.,* `0xAA`). To find out whether the instructions are also in the cache, we *grep* the i-cache contents and confirm that we find all the instructions for our software (compared with the ground truth machine code) within consecutive address spaces.

To quantify the effect of a cache's dynamic behavior on a Linux-based system, we write a microbenchmark with variable array size; the benchmark loads the array from the Flash to DRAM (and d-cache). We run the benchmark in a standard Raspberry Pi OS running on a Raspberry Pi 4 [12]. This SoC has a 32KB two-way

[4]BCM2837 (Cortex-A53) i-cache stores instructions and ECC in each cache line. The order of the bits is undocumented in the technical reference manual [4]. We extract the cached content before and after the attack, which provides an error-free comparison.

Table 4: Extracted data from d-cache of a BCM2711 SoC using *Volt Boot* attack. The size of the d-cache in this SoC is 32KB, which is divided into two ways, $W0$ and $W1$.

	4KB				**8KB**				**16KB**				**32KB**			
	Core 0	Core 1	Core 2	Core 3	Core 0	Core 1	Core 2	Core 3	Core 0	Core 1	Core 2	Core 3	Core 0	Core 1	Core 2	Core 3
$W0$	373.0	338.7	354.7	363.0	591.0	580.7	564.7	581.0	1177.3	1155.0	1179.7	1114.3	1956.7	1980.3	1984.0	1878.0
$W1$	309.0	341.0	340.7	318.0	633.0	659.7	656.7	656.7	1067.3	1097.3	1084.7	1139.0	1990.7	1970.7	1977.3	1815.3
$W0 \cup W1$	512.0	512.0	512.0	512.0	1024.0	1023.7	1024.0	1024.0	2048.0	2048.0	2045.0	2048.0	3747.3	3753.0	3759.3	3509.0
% data extracted	100.00%	100.00%	100.00%	100.00%	100.00%	99.97%	100.00%	100.00%	100.00%	100.00%	99.85%	100.00%	91.49%	91.63%	91.78%	85.67%



(a) d-cache **(b) i-cache**

Figure 8: Snapshots of the caches after executing *Volt Boot* on a system running a general application. We generate the cache images from one WAY of each type of cache.

set-associative data cache. We vary the number of 8-byte elements in the array by increasing the size of the array in each set of experiments. The size of the array varies from 12.5% (4KB) of the cache size to full-cache size (32KB); by extension, the number of elements in the array varies from 512 to 2048. We repeat *Volt Boot* attack on each array size three times and calculate an average number of elements retrieved for each array size.[5] We launch one benchmark process per core, which allows us to analyze how L1 cache's (per core) dynamic behavior affects *Volt Boot*'s data retrieval accuracy.[6] In each experiment, our post-processing script compares the array elements with the retrieved cache image of each core. Note that, other processes (and the kernel) evict cache lines, therefore, an element of the array can be in both ways of the cache in a modified state. We consider an element of the array present in the d-cache only when the entire 8-byte array element is present in the cache. Table 4 lists the results of the L1 data cache data extraction experiment. A 4KB array contains 512 array elements and *Volt Boot* retrieves all the elements. *Volt Boot* retrieves approximately 90% of the array elements when the array size is close to the cache size. That is, when the data size approaches the total cache size, information retrieval accuracy decreases. The kernel's background processes introduce errors in the data extraction by evicting cache lines when the size of a data structure is comparable to the cache size. Note that in the case of on-chip crypto, which uses cache locking (e.g., *CaSE* [44]), *Volt Boot* retrieves the entire binary of plain-text software since neither the kernel nor other processes can evict secret-holding cache lines.

[5]That is, a total of 12 experiments for 4 different array sizes.

[6]At the time of the attack, the victim system concurrently runs 4 processes in the 4 different cores of the Cortex-A72 CPU.

7.2 Attacking CPU Registers

Modern SOCs contain different types of CPU registers that are not part of a typical boot sequence, for example, ARM cores use vector registers `<v0...v31>` to process SIMD and floating-point instructions. These registers are relatively large (128-bit) and byte-addressable, making them suitable for storing security-sensitive states (*e.g.,* key schedules) of cryptographic algorithms, such as AES. Given our threat model, we investigate whether these registers are vulnerable to *Volt Boot*.

We write a bare-metal program in `aarch64` assembly that fills out the vector registers with distinguishable patterns, *e.g.,* 0xFF and 0xAA. Our post-attack analysis on BCM2711 and BCM2837 shows that these vector registers fully retain their states when we execute *Volt Boot* attack. Therefore, any on-chip cryptographic program that hides secrets in these registers is vulnerable to *Volt Boot*.

7.3 Attacking iRAMs

iRAMs (also known as OCRAM) are on-chip memories that an SoC uses as temporary storage for different applications, such as boot firmware and multimedia streaming. We study the vulnerability of these memories to *Volt Boot* attack using a multimedia SoC, the i.MX535 [32], which contains 128KB of iRAM. This memory is in the L1 memory power domain, and it draws power from the `VDDAL1` pin of the SoC. Unlike the BCM2711 and the BCM2837, the i.MX535's ARM core itself draws power through a different pin, `VCCGP`. The i.MX535 boots from internal ROM, and attacking this SoC does not require any external boot media (*e.g.,* Flash). That is, this device essentially behaves as a microcontroller at startup. We attach a JTAG reader to read/write to the iRAM directly and store four copies of a 512 × 512 (128KB) bitmap image to quantify the accuracy of data extraction through *Volt Boot* attack.

Figure 9 illustrates the images that we extract from an iRAM using *Volt Boot*. As mentioned before (Section 6.2), full iRAM is not retrievable because internal boot firmware partially clobbers it before releasing the core to external software. The overall error in the iRAM's extracted information is 2.7%. To locate the exact error source, we calculate the Hamming distance between the image binary and iRAM extracted binary at a 512-bit granularity (Figure 10). The location of the error is clustered around the beginning and end

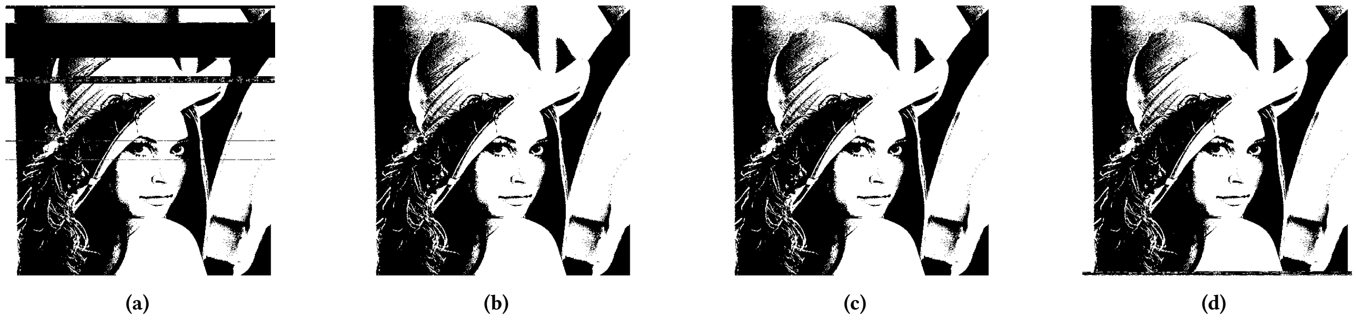


Figure 9: Visual representation of iRAM's data extraction for address (a) $0xF8000000$ to $0xF8007FFF$ (b) $0xF8008000$ to $0xF800FFFF$ (c) $0xF8010000$ to $0xF8017FFF$ (d) $0xF8018000$ to $0xF8020000$.

of the iRAM. The largest source of error address range is between 0x0xF800083C to 0xF80018CC. The device resets this part of the memory, and the error source is consistent with the other i.MX535 devices.

8 COUNTERMEASURES

To assess potential countermeasures, we break down the *Volt Boot* attack into two broad phases. To execute a successful *Volt Boot* attack on an SoC, an attacker must be able to:

(1) induce SRAM data retention across power cycles.
(2) access the unmodified SRAM contents after the system reboots.

To defend against the attack, one can eliminate any one of these essential steps. This section provides insight into some potential countermeasures to *Volt Boot* attack.

- **Eliminating power domain separation:** The decision to separate circuit blocks into power domains involves numerous levels of hardware design stack, ranging *from* device manufacturing *to* architecture. For example, voltage domain separation allows an SoC's power management unit to change a block's behavior when computational demand rises dynamically and to turn it off entirely when not needed. The exposed pins of these domains filter out the noise and maintain stable internal voltage. Eliminating power domain separation is not a practical countermeasure due to performance, efficiency, and implementation concerns.
- **Purging residual memory:** A straightforward way to avoid on-chip data retention in the caches and other on-chip RAMs is to erase the memory as part of the processor's power-down sequence. Such a software/hardware-driven approach is not a practical solution to defend against *Volt Boot* because an abrupt power disconnect from a live device stops all operations immediately.
- **Resetting SRAMs at startup:** Even if an attacker successfully retains the memory states after a power cycle, it becomes useless if there is no feasible method to extract the retained information after a reboot. Resetting the memory using hardware such as *MBIST* or other hardware-driven methods prevent this attack.
  Our experiments on numerous devices, ranging from microcontrollers to application processors, suggest that hardware memory reset at boot-phase is uncommon. Most devices boot into an undefined SRAM state and remain that way until software writes to memory. `armv8.A` suggests that the L2 cache can be reset by pulling `nL2RST` pin low for 16 cycles, but this is not applicable for L1 caches.
  When it comes to caches, the CPU does not benefit from resetting the data and instruction RAMs because the cache operation is dependent on the status of tag RAMs (*e.g.,* L1D-tag and L1I-tag), not the data RAMs (*e.g.,* L1D-data and L1I-Data). Therefore, there is no way to reset the data RAMs in the cache other than executing the zeroization operation provided by the *ISA*, which is a software method. Instead of attempting to reset the memory by writing zero to it, we can adopt a simpler hardware-based solution that internally toggles the power of SRAMs at reset. While effective, this solution requires hardware modification, and therefore, is not applicable to already manufactured SoCs.
- **Trust-Zone support:** ARM Trust-zones(TZ) is a hardware-backed memory isolation technique available in most Cortex-A profile processors.[7] Hardware checks every memory access to assess the validity of the transaction. Cortex-A profile processor stores a security bit (NS) for every line of its cache.

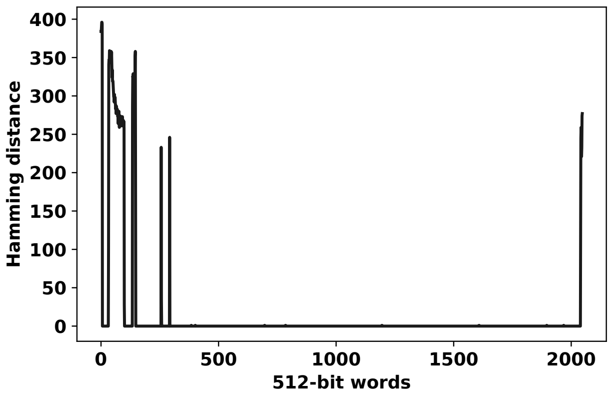


Figure 10: Hamming distance between image binary and post-attack binary.

[7] ARM introduced Trust-Zone in Cortex-M profile devices recently.

Enforcing TZ-support prevents unauthorized access (from non-secure state) to memory locations that are marked secure. An attempt to access a secure line from a non-secure state triggers a hardware exception. The secure data remains inaccessible to an attacker across power cycles when TZ is enforced because the only way to read secure memory content is to change the security attribute of the memory location; such reset erases the memory content.

- **Mandated authenticated boot:** *Volt Boot* needs to boot the system with an exploitable system image. Signing the system with OEM's signature and burning the hash of the image in the fuses prevents an attacker from booting a device from another media. Note that all devices do not have mandated authenticated boot functionality as it complicates post-deployment firmware updates. In addition, some processors boot from internal boot ROM using its internal RAM as a scratchpad and allow direct access to on-chip memory because, of the assumption that on-chip SRAM does not store any information across power cycles.

9 RELATED WORK

The trend towards ubiquitous computing results in an increased attack surface of devices where attackers have physical access. As society increases its dependence on these exposed devices, device security in the physical access threat model grows in importance and criticality. Realizing this, researchers have addressed begun to focus on physical memory disclosure attacks and defenses.

9.1 Cold Boot Attacks

Cold booting is a well-known attack that affects virtually any device with volatile memory with temperature-dependent data retention. *BootJacker* [7] proposes a reboot attack where an attacker gets physical access to a device and force reboots it to a malicious kernel. The authors construct a proof-of-concept attack where the system recovers its state exploiting the data retention property of DRAM cells. Since the reboot method is software controlled (*i.e.,* warm reboot), this attack is easy to mitigate using an OS-level residual memory purger [10]. Halderman *et al.* [17] proposes a more comprehensive attack on DRAM that exploits DRAM's ability to retain data when the temperature is reduced *well below freezing*. As DRAM's discharge rate is lower at cold temperature compared to the typical operating temperature, it allows an attacker to remove the DRAM from a victim machine and plug it in another machine. The attack reconstructs security-critical information such as disk encryption keys from a cold-boot-extracted memory image. Even though DRAM's discharge rate reduces significantly at low temperature, some bits do flip states during migration between the victim's and attacker's machine. These bit flips introduce errors in the extracted information. Consequently, the attacker needs to perform a time-consuming search to reconstruct the correct keys from the recovered content; the search space grows exponentially with the number of flipped bits. In follow-up work, researchers reproduce the cold boot attacks with more modern DRAM technology [5, 24].

DRAM in mobile and IoT devices is usually soldered on the circuit board, which impedes the conventional plug-and-play cold boot attack. FROST [31] shows how an attacker erases an android's non-volatile memory by triggering a factory reset (without erasing the DRAM). Once the user data is erased from a device, an attacker loads a lightweight kernel to dump and analyze the victim's cell phone contents.

Modern systems make it challenging for an attacker to extract sensitive information from post-reboot DRAM memory dump using a low-temperature effect. The defense mechanisms to DRAM cold boot attacks can be broadly classified into two sets. One set of methods scrambles or encrypts the outgoing data. For example, DDR4 uses session keys to scramble data in a DRAM to prevent cold boot attack [43]. Another set of methods advocates overwriting memory contents using built-in hardware every time a DRAM chip powers on [33, 37]. Researchers have evaded memory scrambling in more recent cold boot attacks [43] and self-resetting DRAM has not been implemented or deployed by commercial DRAM vendors.

9.2 SRAM Data Retention Attacks

SRAMs' ubiquity drives a large body of research that studies the implication of data retention properties across power cycles. One set of studies investigates how cooling affects the SRAM's data retention and shows how this is a potential exploit in cloning physical unclonable functions and leaking secrets [2, 3, 6, 16, 40]. These studies report that SRAM's data retention can be as high as ≈80% when removed from power for 20ms at $-110°C$, but post-reboot state retention becomes 0% when the temperature is lowered to $-40°C$ [2]. Another set of studies exploits SRAM's data imprinting effect for applications where a cell holds the same logic state for a long period of time [6, 25, 26, 42]. The fundamental idea behind these attacks is a natural phenomenon known as circuit aging. As software uses an SRAM cell, the circuit goes through analog-domain changes, revealing the logic state that was stored in the cell through its power-on state. These attacks involve a significant investment of time *and* other technical efforts to achieve reasonable data retrieval accuracy and require data to remain in the same SRAM cells with the same value for over a decade to have even modest data recovery. Note that, unlike DRAMs, SRAMs are bistable; therefore, an attacker needs to consider both the logic states for correcting any error in a piece of retrieved secret information.

10 CONCLUSION

The last two decades of hardware security research has seen a rampant increase in proof-of-concept *and* real-world attacks targeting *off-chip memories* designs. Recent efforts to mitigate these attack surfaces primarily turn to *on-chip* computation—i.e., *Cache as RAM*[44]—as their deeply embedded nature renders all previous classes of attacks (e.g., cold boot[17]) obsolete. However, these systems' reliance on *power domain separation* enables a *new* class of attacks: *Volt Boot*.

In this paper, we show that current on-chip SRAM is indeed resilient against conventional temperature "freezing"-based attacks (e.g., [2]). However, we show the effectiveness of a voltage-based attack that snapshots SRAM, *without* exposing an SoC to low temperature, effectively enabling the indefinite retention of SRAM data, while software changes. Compared to previous-generation cold boot attacks against standalone SRAM, *Volt Boot* achieves *error-free* data exfiltration on devices spanning *three* distinct microarchitectures—defeating the paradigm of on-chip computation as a viable defense against secret extraction.

ACKNOWLEDGMENTS

The project depicted is sponsored by the Defense Advanced Research Projects Agency. The content of the information 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] Behnam Amelifard and Massoud Pedram. 2009. Optimal design of the power-delivery network for multiple voltage-island system-on-chips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 6 (2009), 888–900.

[2] Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Markus Rosenstihl, André Schaller, Sebastian Gabmeyer, and Stefan Katzenbeisser. 2018. Low-temperature data remanence attacks against intrinsic SRAM PUFs. In *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 581–585.

[3] Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Markus Rosenstihl, André Schaller, Sebastian Gabmeyer, and Stefan Katzenbeisser. 2019. Attacking SRAM PUFs using very-low-temperature data remanence. *Microprocessors and Microsystems* 71 (2019), 102864.

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

[5] Johannes Bauer, Michael Gruhn, and Felix C Freiling. 2016. Lest we forget: Cold-boot attacks on scrambled DDR3 memory. *Digital Investigation* 16 (2016), S65–S74.

[6] Cagla Cakir, Mudit Bhargava, and Ken Mai. 2012. 6T SRAM and 3T DRAM data retention and remanence characterization in 65nm bulk CMOS. In *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*. IEEE, 1–4.

[7] Ellick M. Chan, Jeffrey C. Carlyle, Francis M. David, Reza Farivar, and Roy H. Campbell. 2008. BootJacker: compromising computers using forced restarts. In *Proceedings of the 15th ACM conference on Computer and communications security - CCS '08*. ACM Press, Alexandria, Virginia, USA, 555. https://doi.org/10.1145/1455770.1455840

[8] Dawei Chu, Yuewu Wang, Lingguang Lei, Yanchu Li, Jiwu Jing, and Kun Sun. 2019. OCRAM-assisted sensitive data protection on ARM-based platform. In *European Symposium on Research in Computer Security*. Springer, 412–438.

[9] Patrick Colp, Jiawen Zhang, James Gleeson, Sahil Suneja, Eyal De Lara, Himanshu Raj, Stefan Saroiu, and Alec Wolman. 2015. 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*. 177–189.

[10] Computing, Trusted. 2008. TCG platform reset attack mitigation specification. https://trustedcomputinggroup.org/wp-content/uploads/Platform-Reset-Attack-Mitigation-Specification.pdf.

[11] RASPBERRY PI FOUNDATION. 2021. Raspberry Pi 3 Model B:Single-board computer with wireless LAN and Bluetooth connectivity. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[12] RASPBERRY PI FOUNDATION. 2021. Raspberry Pi 4. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[13] Behrad Garmany and Tilo Müller. 2013. PRIME: Private RSA Infrastructure for Memory-Less Encryption. In *Proceedings of the 29th Annual Computer Security Applications Conference* (New Orleans, Louisiana, USA) *(ACSAC '13)*. Association for Computing Machinery, New York, NY, USA, 149–158. https://doi.org/10.1145/2523649.2523656

[14] Le Guan, Jingqiang Lin, Bo Luo, and Jiwu Jing. 2014. Copker: Computing with Private Keys without RAM.. In *NDSS*. 23–26.

[15] Le Guan, Jingqiang Lin, Ziqiang Ma, Bo Luo, Luning Xia, and Jiwu Jing. 2016. Copker: a cryptographic engine against cold-boot attacks. *IEEE Transactions on Dependable and Secure Computing* 15, 5 (2016), 742–754.

[16] Peter Gutmann. 2001. Data Remanence in Semiconductor Devices.. In *USENIX Security Symposium*. 39–54.

[17] J Alex Halderman, Seth D Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A Calandrino, Ariel J Feldman, Jacob Appelbaum, and Edward W Felten. 2009. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM* 52, 5 (2009), 91–98.

[18] Michael Henson and Stephen Taylor. 2013. Beyond full disk encryption: Protection on security-enhanced commodity processors. In *International Conference on Applied Cryptography and Network Security*. Springer, 307–321.

[19] D. E. Holcomb, W. P. Burleson, and K. Fu. 2009. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Trans. Comput.* 58, 9 (Sept. 2009), 1198–1210. https://doi.org/10.1109/TC.2008.212

[20] Daniel E Holcomb, Amir Rahmati, Mastooreh Salajegheh, Wayne P Burleson, and Kevin Fu. 2012. DRV-fingerprinting: Using data retention voltage of SRAM cells for chip identification. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, 165–179.

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

[22] G Knight. 2017. Encrypt data using VeraCrypt. (2017).

[23] Arm Limited. 2016. ARM Cortex-A72 MPCore Processor Technical Reference Manual. https://developer.arm.com/documentation/100095/0003/.

[24] Simon Lindenlauf, Hans Höfken, and Marko Schuba. 2015. Cold boot attacks on DDR2 and DDR3 SDRAM. In *2015 10th International Conference on Availability, Reliability and Security*. IEEE, 287–292.

[25] Taizhi Liu, Chang-Chih Chen, Jiadong Wu, and Linda Milor. 2016. SRAM stability analysis for different cache configurations due to bias temperature instability and hot carrier injection. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*. IEEE, 225–232.

[26] Abhranil Maiti, Logan McDougall, and Patrick Schaumont. 2011. The impact of aging on an FPGA-based physical unclonable function. In *2011 21st International Conference on Field Programmable Logic and Applications*. IEEE, 151–156.

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

[28] Microsoft. 2019. Overview of BitLocker Device Encryption in Windows 10. https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-device-encryption-overview-windows-10.

[29] Praveen Mosalikanti, Chris Mozak, and Nasser Kurd. 2011. High performance DDR architecture in Intel® Core™ processors using 32nm CMOS high-K metal-gate process. In *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test*. IEEE, 1–4.

[30] Tilo Müller, Felix C. Freiling, and Andreas Dewald. 2011. TRESOR Runs Encryption Securely Outside RAM. In *Proceedings of the 20th USENIX Conference on Security* (San Francisco, CA) *(SEC'11)*. USENIX Association, USA, 17.

[31] Tilo Müller and Michael Spreitzenbarth. 2013. FROST: Forensic Recovery of Scrambled Telephones. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security* (Banff, AB, Canada) *(ACNS'13)*. Springer-Verlag, Berlin, Heidelberg, 373–388. https://doi.org/10.1007/978-3-642-38980-1_23

[32] NXP Semiconductors. 2021. i.MX535: Multimedia Applications Processors - HD Video, High-Performance, Low-Power, ARM Cortex®-A8 Core. https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-mature-processors/multimedia-applications-processors-hd-video-high-performance-low-power-arm-cortex-a8-core:i.MX535.

[33] Lois Orosa, Yaohua Wang, Mohammad Sadrosadati, Jeremie S. Kim, Minesh Patel, Ivan Puddu, Haocong Luo, Kaveh Razavi, Juan Gómez-Luna, Hasan Hassan, Nika Mansouri-Ghiasi, Saugata Ghose, and Onur Mutlu. 2021. CODIC: A Low-Cost Substrate for Enabling Custom in-DRAM Functionalities and Optimizations. In *Proceedings of the 48th Annual International Symposium on Computer Architecture* (Virtual Event, Spain) *(ISCA '21)*. IEEE Press, 484–497. https://doi.org/10.1109/ISCA52012.2021.00045

[34] Hulfang Qin, Yu Cao, Dejan Markovic, Andrei Vladimirescu, and Jan Rabaey. 2004. SRAM leakage suppression by minimizing standby supply voltage. In *International Symposium on Signals, Circuits and Systems. Proceedings, SCS 2003.(Cat. No. 03EX720)*. IEEE, 55–60.

[35] Amir Rahmati, Mastooreh Salajegheh, Dan Holcomb, Jacob Sorber, Wayne P Burleson, and Kevin Fu. 2012. {TARDIS}: Time and Remanence Decay in {SRAM} to Implement Secure Protocols on Embedded Devices without Clocks. In *21st {USENIX} Security Symposium ({USENIX} Security 12)*. 221–236.

[36] Alec Roelke and Mircea R Stan. 2016. Attacking an SRAM-based PUF through Wearout. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 206–211.

[37] Hoseok Seol, Minhye Kim, Taesoo Kim, Yongdae Kim, and Lee-Sup Kim. 2021. Amnesiac DRAM: A Proactive Defense Mechanism Against Cold Boot Attacks. *IEEE Trans. Comput.* 70, 4 (2021), 539–551. https://doi.org/10.1109/TC.2019.2946365

[38] SiFive, Inc. 2019. SiFive U54-MC Manual. https://simfive.cdn.prismic.io/sifive%2Fdc4980ff-17db-448b-b521-4c7ab26b7488_sifive+u54-mc+manual+v19.08.pdf.

[39] Patrick Simmons. 2011. Security through amnesia: a software-based solution to the cold boot attack on disk encryption. In *Proceedings of the 27th Annual Computer Security Applications Conference*. 73–82.

[40] Sergei Skorobogatov. 2002. *Low temperature data remanence in static RAM*. Technical Report. University of Cambridge, Computer Laboratory.

[41] TestEquity. 2021. Model 123H Temperature/Humidity Chamber. https://www.testequity.com/category/Environmental-Chambers-Ovens/Temperature-Humidity-Chambers/TestEquity-123H-Temperature-Humidity-Chamber-

North-America-Version-17267-1.

[42] Harrison Williams, Alexander Lind, Kishankumar Parikh, and Matthew Hicks. 2020. Silicon Dating. *arXiv preprint arXiv:2009.04002* (2020).

[43] Salessawi Ferede Yitbarek, Misiker Tadesse Aga, Reetuparna Das, and Todd Austin. 2017. Cold Boot Attacks are Still Hot: Security Analysis of Memory Scramblers in Modern Processors. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, Austin, TX, USA, 313–324. https://doi.org/10.1109/HPCA.2017.10

[44] Ning Zhang, Kun Sun, Wenjing Lou, and Y Thomas Hou. 2016. Case: Cache-assisted secure execution on arm processors. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 72–90.