FirmEM: Firmware Flashing Detection via Unintentional Electromagnetic Emissions

Elvan M. Ugurlu
Aether Argus Inc.
Atlanta, GA, US
elvan@aetherargus.com

Baki B. Yilmaz
Aether Argus Inc.
Atlanta, GA, US
baki@aetherargus.com

Angelos D. Keromytis
Aether Argus Inc.
Atlanta, GA, US
angelos@aetherargus.com

***Abstract*—Ensuring the integrity of firmware running on embedded systems—such as automotive Electronic Control Units (ECUs)—is critical to system reliability and security. This paper presents a lightweight, non-intrusive method for detecting firmware flashing events using electromagnetic (EM) side-channel emissions collected via low-cost software-defined radios (SDRs). By focusing on short, deterministic update phases (e.g., bootloader-based flashing), we avoid the overhead of continuous monitoring. We propose a kernel-based pattern matching framework that identifies flashing operations by comparing short-time spectral signatures to a reference. We validate this approach on SAMV71 development boards and show that appropriate frequency band selection is key to signal clarity. Cross-board experiments reveal challenges related to hardware variability, including missing components. These results highlight the feasibility of EM-based firmware inspection in real-world, resource-constrained environments.**

***Index Terms*—EM side-channels, firmware verification, flash detection, boot-load monitoring.**

I. Introduction

Modern embedded systems, including those found in vehicles, industrial machinery, and medical devices, rely on firmware—low-level software that interfaces directly with hardware—to execute mission-critical functions. The integrity of this firmware is therefore critical to ensuring reliable and secure system behavior. In the automotive domain, for example, electronic control units (ECUs) manage essential operations such as braking, steering, and powertrain control. Unfortunately, attackers have demonstrated the ability to modify embedded systems with malicious or outdated firmware, bypassing traditional runtime protections and compromising system functionality [1], [2].

While electromagnetic (EM) side-channel analysis has been shown to reveal meaningful information about device behavior, much of the prior work either relies on lab-grade equipment or is restricted to FPGA platforms and simulation models [3]. Other approaches for detecting firmware modifications either require intrusive instrumentation, proprietary vendor support, or high-cost side-channel measurement equipment. There remains a gap in the literature for solutions that work non-invasively, require no architectural support, and are viable for low-cost field deployment.

In this paper, we present a practical and resource-constrained EM monitoring framework for detecting firmware reflashing events and firmware variation on a real ARM-based microcontroller equipped on SAMV71 boards [4]. Our system focuses specifically on the bootloading phase, where we demonstrate the feasibility of pattern-based recognition in the time domain. This approach allows for fast detection and reduced processing latency that could also be used in run-time for continuous monitoring to detect certain events based on a reference model trained on the EM signatures of these events.

Using a simple software-defined radio (SDR) receiver and fixed-probe setups, we show that even constrained tools can provide high-confidence detection of firmware flashing. We then explore the extent of what more we can infer—including cross-device generalization and robustness against manufacturing variance. This work raises important questions about real-world deployment and also surfaces potential limitations such as hardware variation or sensitivity to minor changes in configuration. In summary, this paper makes the following key contributions:

- Demonstrate firmware flashing detection using near-field probes and low-cost SDRs (e.g., RTL-SDR, HackRF) with minimal processing and narrow-band capture.
- Propose a kernel-based pattern matching framework for fast, low-latency spectral signature detection.
- Identify deployment challenges due to hardware inconsistencies (e.g., missing components across boards).
- Provide evidence for broader applicability of the proposed framework using advanced SDRs (e.g., USRPs) for finer-grained runtime firmware differentiation.

The rest of this paper is organized as follows. Section II provides background on firmware security, existing EM-based monitoring approaches, and motivates the need for a lightweight and reliable firmware monitoring. Section III presents our methodology for detecting firmware events using EM emissions, including our kernel-based pattern matching approach. In Section IV we describe our experimental setup. Section V presents our experimental results, highlighting detection performance and cross-device generalization under hardware variability. Finally, Section VI concludes the paper.

II. Background and Motivation

A. Firmware Updates and Security

Embedded systems are pervasive in modern life, performing critical tasks across a wide range of applications, such as

ECUs in automobiles. ECUs serve as the intelligence backbone of vehicles, coordinating interactions among sensors, actuators, and communication interfaces. They rely on embedded firmware to execute real-time control functions and respond to safety-critical events. This firmware is typically stored in non-volatile memory (e.g., flash) and can be updated through diagnostic interfaces, either via on-board diagnostic (OBD) ports or remotely using over-the-air (OTA) mechanisms [5].

Firmware updates on such systems are typically deployed as compiled images and written over serial or debugging interfaces. Secure firmware updates often involve multiple layers: authentication of the update source, integrity checks on the firmware binary, secure boot mechanisms to verify firmware at runtime, and rollback protections.

In practice, however, many legacy systems or less critical ECUs lack full support for these measures. Moreover, even when secure boot is implemented, it is often not designed to detect physical-layer reflashing attempts or low-level firmware changes made through unauthorized access.

Most ECUs are updated regularly for feature enhancements, while others are tightly controlled. Despite these efforts, the diversity of update pathways and legacy support introduces a broad attack surface [6], [7]. There have been numerous demonstrations of firmware-level attacks on automotive systems including reverse-engineer firmware and inject malicious code via exposed diagnostic interfaces, bypassing OEM protections [8]. These attacks typically go unnoticed until runtime behavior changes or failures occur, which can be catastrophic in safety-critical systems. The issue is compounded by the lack of visibility into the bootloading phase, where reflashing often occurs and a malicious reprogramming event may leave no trace in runtime logs or digital attestation systems.

B. Defensive Use of Unintentional EM Emissions

The phenomenon of unintentional EM emissions from electronic devices has long been studied in the context of hardware security and side-channel analysis [9]. These emissions—byproducts of rapid switching activity within integrated circuits—can unintentionally leak sensitive information or reveal execution behavior.

In adversarial settings, researchers have demonstrated the feasibility of extracting cryptographic keys from FPGAs and embedded systems using EM side-channels, performing remote code execution inference on mobile devices, and even launching EM fault injection attacks to bypass security checks [10]–[13]. These studies exploit fundamental principles of electromagnetics and circuit theory, making the existence of such channels unsurprising—if not inevitable—given the physical structure of modern electronics.

However, beyond adversarial use cases, researchers have also explored EM emissions as a nonmalicious tool for passive software monitoring and system introspection. Nonmalicious EM-based software monitoring frameworks such as Spectral Profiling [14] and EDDIE [15] have demonstrated the viability of EM spectral analysis for software behavior detection, they typically operate at coarse-grained levels (e.g., loops, function calls, states). Further studies have shown instruction-level tracking techniques using high-fidelity spectrum analysis and signal processing pipelines [16], [17].

Multimodal analog side-channel analysis is a promising non-intrusive path forward for detecting state divergences and physical-layer anomalies in cyber-physical systems. Recent work by Kacmarcik et al. [18] further demonstrates that acoustic emissions, in addition to EM emissions, can also be used to identify abnormal activity in embedded systems, reinforcing the idea that a multi-channel monitoring approach can improve anomaly detection. This opens the door for firmware-level integrity checks using antennas or sensors placed near different components, particularly during sensitive phases like bootloading or reflashing.

Modern vehicles contain over a hundred modular ECUs with limited software complexity, making them well-suited for side-channel-based monitoring. Their periodic behavior—driven by clocks, buses, and control loops—produces distinct frequency-domain patterns. This structure enables lightweight, spectral techniques like ours to detect anomalies or unauthorized firmware changes with minimal overhead.

C. Motivation and Proposed Use

While many existing efforts leveraging the EM emissions target continuous monitoring of software execution, such approaches often require high-resolution signal acquisition, detailed system knowledge, or persistent observability—conditions that are difficult to meet in many real-world deployments. In contrast, this work focuses on a more practical and bounded target: the firmware flashing phase. Firmware flashing is a short-lived but security-critical operation, typically occurring during system reboots or intentional reprogramming (e.g., during updates or maintenance). These events present a unique opportunity for low-overhead integrity

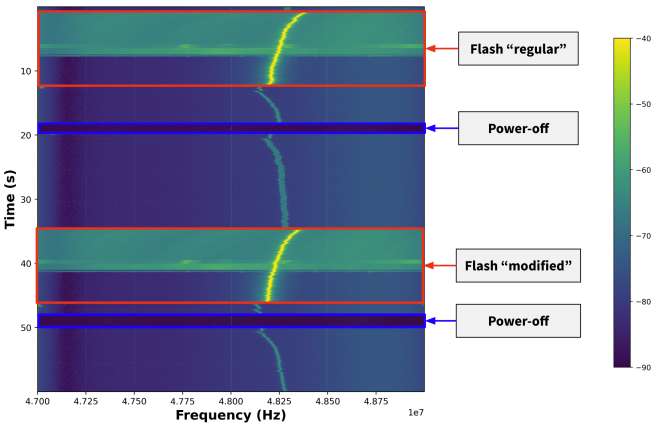

Fig. 1: Time–frequency spectrogram of EM emissions during two firmware flashing events. The first event (top red box) highlights regular firmware flashing, while the second event (bottom red box) corresponds to the flashing of a modified firmware.

checks, as they are deterministic, repeatable, and temporally isolated from normal operation.

Fig. 1 presents a spectrogram (time–frequency representation) of EM emissions collected from the SAMV71 [4] development board using an RTL-SDR [19] device over a 60-second interval.

The highlighted red box region corresponds to firmware flashing operations, which occur over just a few seconds, clearly exhibiting a distinct chirp-like spectral pattern. In this experiment, we flash two different images onto the microcontroller, one representing a regular/original firmware akin to what might be deployed on a production ECU, and another that is slightly modified to simulate a different firmware version. In our experiments, we utilized these two firmware versions to illustrate the applicability of this approach in scenarios such as the temporal identification of "authorized updates" or the detection of an "unauthorized modification."

Both events produce spectrally rich and visually distinct chirp-like patterns, suggesting that firmware flashing can be characterized using compact spectral signatures. This observation motivates our proposed detection framework, which performs lightweight, out-of-band verification of flashing events by identifying these spectral and temporal features—even in the presence of hardware diversity or resource constraints.

III. Methodology

Our methodology aims to detect firmware flashing events from EM emissions using a low-latency pattern matching approach. We divide the process into two main phases: training and testing.

A. Training Phase

In the training phase, we extract one or more spectral reference signatures that characterize the EM behavior of known flashing events. While a single averaged signature can suffice for simple detection tasks, our setup allows for capturing variations across firmware versions, boards, or experimental conditions. To enable automated detection, we translate the visually identifiable flashing patterns observed in the spectrogram into structured spectral representations through the following preprocessing and feature extraction steps.

1) We first identify and crop time segments from spectrograms where flashing occurs. These segments exhibit consistent spectral patterns that make them visually distinguishable.
2) The average duration of the cropped segments is used to define a fixed-length analysis window, or *frame size*, applied in both training and testing.
3) For each segment, we compute the average FFT magnitude across time to obtain a corresponding one dimensional *spectral profile*, as illustrated in Fig. 2.
4) These profiles can either be averaged into a single *reference signature*, or stored individually for more fine-grained, instance-based matching. This flexibility supports multiple detection strategies, such as using firmware-specific kernels or board-dependent references.

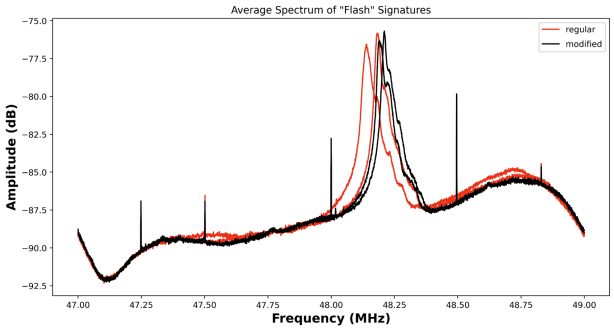

Fig. 2: *Spectral profile*s of four individual flashing events across two firmware versions ("regular" in red, "modified" in black).

This process yields one or more compact EM signatures for flashing events, serving as pattern-matching kernels for subsequent detection.

B. Testing Phase

During testing, the objective is to identify whether the incoming EM emissions match any known *reference signature*(s) generated during training.

1) The incoming EM signal is continuously transformed into its time-frequency representation using STFT.
2) A sliding window (with the *frame size* defined during training) is applied across the STFT output. For each window, we compute the corresponding 1D *spectral profile*, i.e., the average FFT magnitude across time.
3) For each window, we compute the Euclidean distance between its *spectral profile* and the *reference signature*(s) obtained during training. The inverse of this distance serves as a similarity score.
4) A flashing event is detected when the similarity score exceeds a predefined threshold. Optionally, the time corresponding to the peak similarity score can be used to estimate the onset of the flashing event.

This technique enables low-latency detection even on constrained hardware. The threshold can be calibrated empirically to balance detection sensitivity against false positives. By supporting both single-template and multi-template detection, the approach adapts to different operational scenarios — such as varying firmware versions, boards, or sampling conditions.

IV. Experimental Setup

A. Hardware Platform and Firmware Variants

Fig. 3 illustrates the experimental setup used to investigate firmware-level activities on a SAMV71 development board. These boards are commonly used in embedded systems prototyping and serve as a practical proxy for automotive ECUs.

At the core of each SAMV71 board is the ATSAMV71, an ARM Cortex-M7-based microcontroller, which we designate as the primary device under test (DuT) throughout our experiments. A near-field probe is positioned directly above this chip to monitor EM emissions during firmware operations.

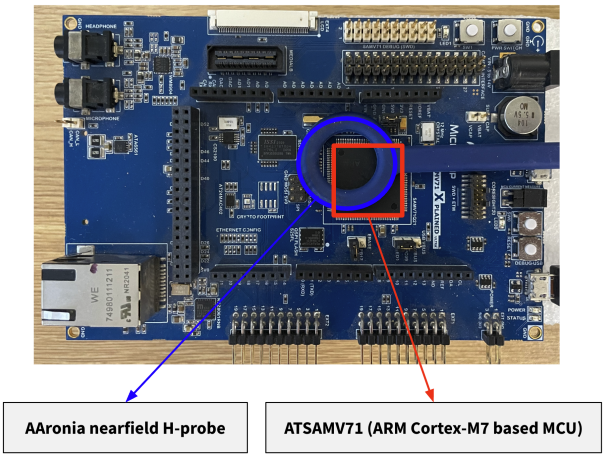

Fig. 3: Experimental setup for monitoring the embedded ARM-based microcontroller on SAMV71 Microchip Development Board.

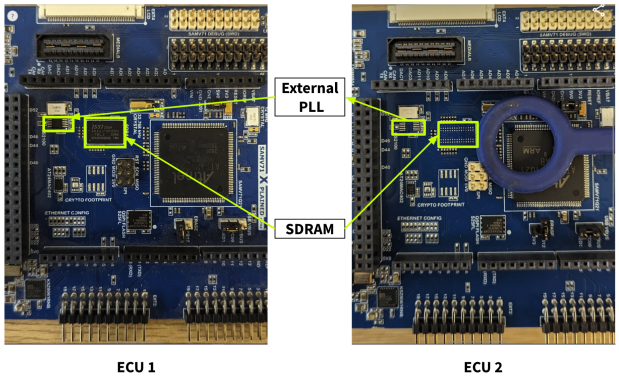

Fig. 4: Comparison of two SAMV71-based ECU boards used in this study. ECU 1 includes an external PLL and onboard SDRAM, while ECU 2 lacks both components. These discrepancies, highlighted in green, illustrate the hardware variability observed across boards acquired from different vendors.

Interestingly, we discovered notable differences between boards sourced from different vendors—some missing components like external PLL and SDRAM (as demonstrated in Fig. 4). These differences affect signal characteristics and reflect realistic inconsistencies one might encounter in field-deployed embedded systems and raise important questions:

- Would such component-level differences (missing or altered components) on the board affect EM-based detection performance?
- How robust are EM-based detection systems for these differences?
- Are there other risks such as tampering, poor manufacturing, counterfeits, hardware trojans, etc. that can intentionally or unintentionally change the board's EM emissions characteristics?

To investigate the generalizability of our detection approach and address these questions, we experimented on multiple SAMV71 boards with hardware variations, acquired from different vendors at different times.

In this study, we evaluate two firmware versions compiled for the same SAMV71 target: 1) "regular" firmware image and 2) "modified" firmware image. Both are valid *.hex* files that can be successfully flashed onto the microcontroller and execute without triggering any observable faults or errors. The modification introduced in the altered version is intentionally minimal. As a result, the run-time behavior and execution patterns of the two programs are virtually indistinguishable when observed through EM emissions during normal operation.

B. EM Probing and Signal Acquisition Setup

To observe and analyze the electromagnetic emissions during firmware flashing, we use a near-field magnetic (H-field) probe [20] positioned directly above the ATSAMV71 microcontroller. A visual depiction of the probe positioning relative to key components is shown in Fig. 3. Probe positioning is known to be a critical factor in EM signal locality [21].

High-fidelity signal acquisition platforms such as spectrum analyzers or advanced software-defined radios (e.g., Ettus USRP series) are well-suited for fine-grained EM analysis, such as instruction-level tracking and run-time program monitoring. Existing literature for instruction-level EM modeling such as ZOP [16] and PITEM [17] relies on such platforms to achieve high spatial and temporal resolution at the cost of increased complexity, size, and price.

In contrast, this work aims to explore how far low-cost and accessible tools can go, especially for practical applications like detecting firmware flashing events during the bootloading stage. We focus our experiments on the RTL-SDR v3 [19], a low-cost ($\sim$\$30 USD) receiver with limited bandwidth ($\sim$ 2.4 MHz usable) and no onboard amplification. We also evaluated the HackRF One [22] as a mid-range alternative and found that it provides similar results as well.

Choosing the appropriate center frequency is a critical step when using narrow-band SDRs like the RTL-SDR. While wideband systems such as USRPs can capture broad spectral regions and extract relevant features post-hoc, low-cost receivers are constrained to a limited view of the spectrum—typically just 2–3 MHz wide. This constraint reduces background noise and data volume, which is advantageous for real-time or embedded deployment, but introduces a key trade-off: one must carefully select a frequency band that consistently captures the event of interest while avoiding regions dominated by unrelated or high-power interference.

Fig. 5 illustrates this trade-off using spectrograms collected during three consecutive firmware flashing events. When centered at 48 MHz (left), the flashing signatures appear as distinct chirp-like structures, separated by quiet idle periods—making detection straightforward, even using basic time-domain methods. In contrast, the spectrogram centered at 50 MHz (right)—aligned with the microcontroller's system clock—is cluttered with persistent vertical lines. These clock-related artifacts and background modulations are present re-

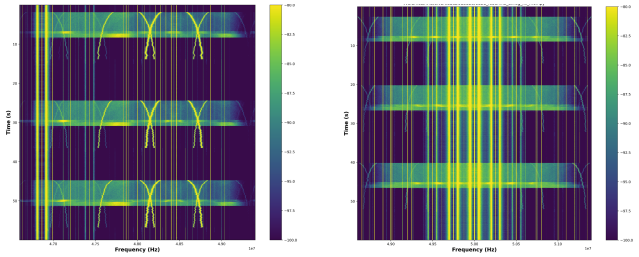

Fig. 5: Comparison of EM spectrograms captured during three consecutive firmware flashing events using RTL-SDR, with different center frequencies.

gardless of system state and obscure the true flashing signature, increasing the likelihood of false detections. This comparison highlights that in narrow-band monitoring, selecting the right slice of spectrum is critical: even a small shift in center frequency can dramatically affect signal clarity and overall detection robustness.

V. Experimental Results

A. Temporal Detection

To evaluate the temporal resolution and detection capability of our method, in Fig. 6 we visualize how similarity scores evolve over time during flashing events. For this purpose, we reorient the spectrograms to emphasize temporal alignment: the horizontal axis represents time, while the vertical axis shows frequency. Directly below the spectrogram, we display the corresponding similarity curves.

Each colored trace in the bottom plot of Fig. 6 corresponds to one of five reference signatures—four obtained from individual flashing instances and one computed as their average. To detect flashing events, we slide a fixed-length analysis window (equal to the defined *frame size*) across the input spectrogram. For each window, we compute its spectral profile by averaging the FFT magnitudes over time, efficiently leveraging the STFT output in a moving-window manner.

We then calculate the Euclidean distance between this spectral profile and each reference signature. The inverse of this distance is plotted over time, producing a similarity score curve for each reference. Peaks in these curves indicate strong correspondence between the observed signal and known flashing patterns. As seen in the figure, all reference signatures yield clear peaks during flashing intervals, with the averaged signature offering smoother but robust detection—illustrating a tradeoff between event-specific sensitivity and generalizability.

This variation across instances underscores the importance of accounting for hardware and environmental variability when designing detection systems. While some differences may arise from benign sources—such as slight changes in probe positioning or board layout—others may reflect more significant hardware discrepancies. Rather than attempting to distinguish between these causes directly, our goal is to evaluate how such variability impacts detection performance and whether the system remains reliable across different devices.

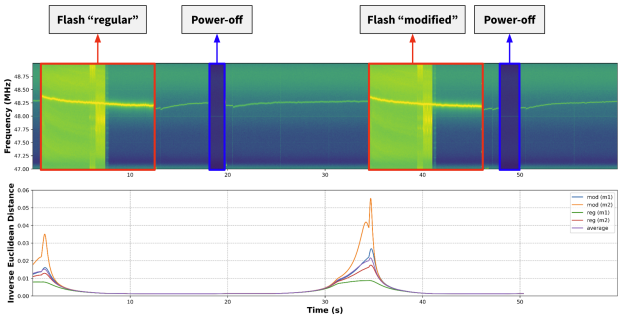

Fig. 6: Visualization of two flashing events using rotated (for temporal alignment) time–frequency spectrograms (top) and corresponding inverse Euclidean distance scores (bottom), which serves as similarity score to the corresponding *reference signature*s

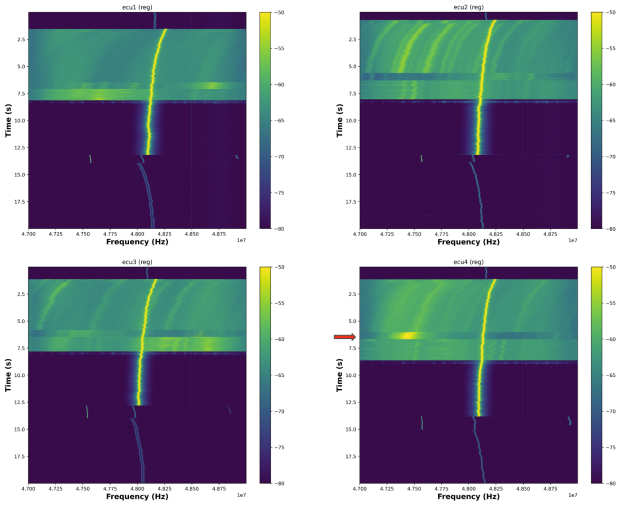

Fig. 7: Spectrograms of regular firmware flashing events across four ECU boards (ECU1–ECU4), recorded using RTL-SDR with a 48 MHz center frequency and 2 MHz sampling rate. Each spectrogram was collected using the same setup, although minor differences in probe positioning and board-level component configurations (e.g., missing passives or manufacturing variation) are possible.

B. Cross-board Testing

To assess cross-board generalization of the proposed approach, we evaluate the system on four separate SAMV71 ECU boards—i.e., different physical instances of the same microcontroller model, sourced from various vendors and batches. In this context, we use the term "board" to refer to variations in hardware configuration, manufacturing tolerances, or passive component presence, while keeping the underlying microcontroller (i.e., the device) the same.

While spectrograms recorded using RTL-SDR (visualized in Fig. 7) showed consistent flashing behavior across all boards, particularly in timing and overall structure, we observed slight variations in power levels (such as the one highlighted by the red arrow in Fig. 7), likely caused by hardware differences such as missing components or manufacturing tolerances.

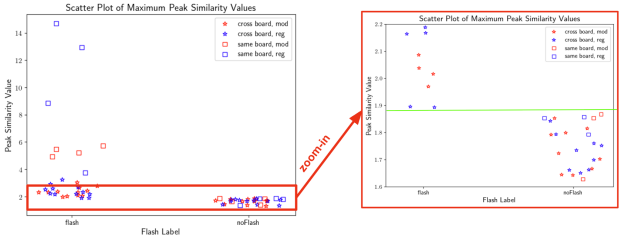

Fig. 8: Scatter plots of maximum similarity scores during flashing and non-flashing intervals across same-board and cross-board scenarios.

Specifically, we evaluate how well the system distinguishes between flashing and non-flashing intervals under both same-board and cross-board scenarios. Here, same-board refers to cases where both the *reference signature* and the test data originate from the same physical board. In contrast, cross-board refers to cases where the test data is acquired from a different physical board—i.e., a separate instance of the same microcontroller model—than the one used to construct the reference. For cross-board evaluations, the *reference signature* is computed by averaging the flashing signatures from the other three boards (excluding the test board). These are marked with square (same-board) and star (cross-board) symbols in Fig. 8. As expected, same-board scenarios yield higher peak similarity scores during flashing intervals, reflecting stronger alignment with their corresponding *reference signatures*. In contrast, cross-board scenarios show attenuated peak values, particularly for boards with different passive components or minor manufacturing variations.

The scatter plot includes labels for the two firmware types (regular and modified) in the legend, but these labels don't reflect meaningful similarity scores. Our experiments don't show a consistent trend where one firmware type yields higher or lower similarity. This supports our focus on detecting flashing behavior, regardless of firmware content. The zoomed-in view shows a promising trend: despite reduced peak magnitudes, a green decision boundary can effectively separate flash from no-flash intervals even under cross-board conditions. This suggests that cross-board detection is feasible, especially with improved training strategies like using representative subsets or augmented reference kernels. These findings highlight the potential for robust flashing detection across multiple hardware instances, despite structural or environmental differences.

C. Firmware Classification

To investigate whether different firmware versions yield distinguishable EM signatures during the flashing process, we conducted a comparative analysis using both RTL-SDR and HackRF receivers. Fig. 9 illustrates spectrograms for regular and modified firmware flashing captured by each receiver.

Despite the visual similarities in coarse-grained spectral structure observed in Fig. 1, this figure shows that there are clear temporal differences between the firmware versions. These timing differences, which are consistent across both RTL-SDR and HackRF, indicate that even firmware variants

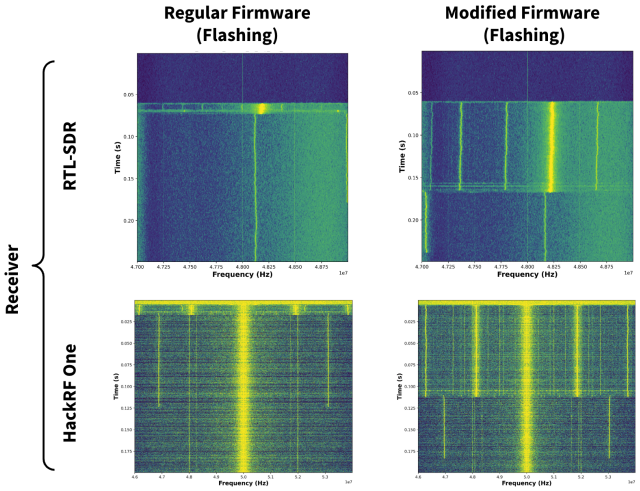

Fig. 9: Timing comparison of flash durations between firmware versions (collected with RTL-SDR and HackRF).

with similar signal shapes may differ in duration, suggesting minor but measurable behavioral variations in the flashing process.

Importantly, observing such subtle distinctions required targeted experimental design and precise alignment to isolate and zoom into the relevant timeframe within the broader flashing sequence. This level of granularity cannot be easily achieved through full-spectrum visual inspection alone, further justifying our use of time-domain pattern matching in the proposed methodology.

Additionally, the HackRF receiver appears to capture richer and more detailed features, potentially due to its higher sampling rate and wider instantaneous bandwidth. Nonetheless, our methodology remains agnostic to receiver type: while signal quality may vary, the core detection and classification techniques are portable across devices.

D. Runtime Monitoring

Beyond flashing-phase detection, we explore whether EM emissions collected during normal runtime operation can reflect differences between firmware versions. Fig. 10 presents spectrograms acquired using three different signal acquisition tools—a Siglent spectrum analyzer, a USRP B210, and a HackRF One—while the ECU was running either regular or modified firmware; the left column shows emissions from the regular firmware, and the right column shows those from the modified version. All recordings were taken at a center frequency of 50 MHz (clock frequency of DuT during runtime) with an 8 MHz sampling rate.

The ability to distinguish between firmware versions during runtime strongly depends on signal quality. With high-end equipment such as the Siglent SA [23], we observe clear differences: the regular firmware exhibits periodic wideband activity, while the modified firmware produces more burst-like, irregular emissions. The USRP B210 [24] captures similar trends but with reduced clarity. In contrast, the HackRF One

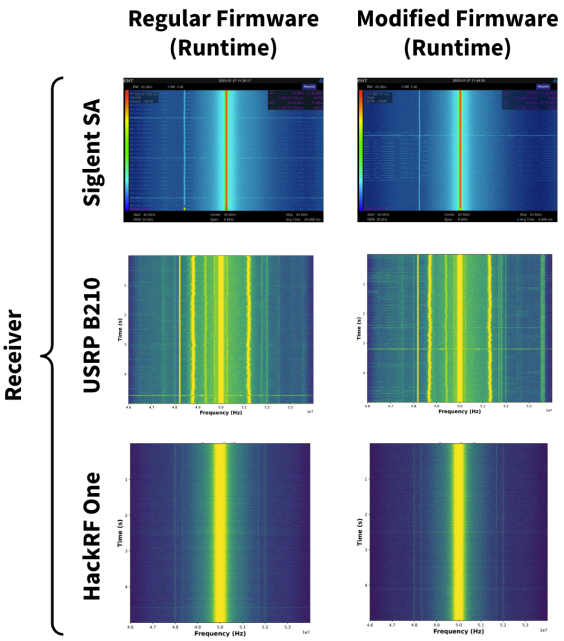

Fig. 10: Spectrograms captured during runtime operation of the same ECU using different receivers: Siglent Spectrum Analyzer [23], USRP B210 [24], and HackRF One [22].

[22] struggles to resolve these differences due to its limited resolution and dynamic range.

VI. Conclusion

This work presents a practical and cost-effective approach for detecting firmware flashing events in embedded systems using electromagnetic side-channel emissions. By targeting the bootloader phase and employing a kernel-based pattern matching technique, we demonstrate that low-cost SDRs can reliably detect reflashing operations without the need for intrusive instrumentation or vendor-specific support. Our experimental results on SAMV71 development boards show that careful frequency band selection and simple signal processing can yield high detection accuracy, even in the face of hardware variability across different manufacturers. While our method proves robust in controlled settings, it also highlights important considerations for deployment—such as sensitivity to missing components and potential configuration drift. Ultimately, this study underscores the feasibility of retrofitting security assurance into embedded systems through low-overhead EM-based monitoring. Such an approach would be particularly useful for enhancing the security of critical systems, such as ECUs, where maintaining firmware integrity is essential for operational safety and reliability.

References

[1] J. Van den Herrewegen, "Automotive firmware extraction and analysis techniques," Ph.D. dissertation, University of Birmingham, 2021.

[2] H. Mansor, K. Markantonakis, R. N. Akram, and K. Mayes, "Don't brick your car: firmware confidentiality and rollback for vehicles," in *2015 10th International Conference on Availability, Reliability and Security*. IEEE, 2015, pp. 139–148.

[3] A. Zajić and M. Prvulovic, Eds., *Understanding Analog Side Channels Using Cryptography Algorithms*. Cham: Springer International Publishing, 2023. [Online]. Available: https://doi.org/10.1007/978-3-031-38579-7

[4] Microchip, "Atsamv71q21," https://www.microchip.com/en-us/product/atsamv71q21.

[5] B. Li, W. Hu, L. Da, Y. Wu, X. Wang, Y. Li, and C. Yuan, "Over-the-air upgrading for enhancing security of intelligent connected vehicles: a survey," *Artificial Intelligence Review*, vol. 57, no. 11, p. 314, 2024.

[6] D. K. Nilsson, P. H. Phung, and U. E. Larson, "Vehicle ecu classification based on safety-security characteristics," in *IET Road Transport Information and Control-RTIC 2008 and ITS United Kingdom Members' Conference*. IET, 2008, pp. 1–7.

[7] F. Kohnhäuser, D. Püllen, and S. Katzenbeisser, "Ensuring the safe and secure operation of electronic control units in road vehicles," in *2019 IEEE Security and Privacy Workshops*. IEEE, 2019, pp. 126–131.

[8] S. Córdoba Pellicer, "Security assessment for automotive controllers using side channel and fault injection attacks," Master's thesis, Universitat Politècnica de Catalunya, 2018.

[9] A. Zajić and M. Prvulovic, *Using Analog Side Channels for Hardware Event Profiling*. Cham: Springer International Publishing, 2023, pp. 279–321. [Online]. Available: https://doi.org/10.1007/978-3-031-38579-7_10

[10] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side—channel (s)," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 29–45.

[11] M. Alam, H. A. Khan, M. Dey, N. Sinha, R. Callan, A. Zajic, and M. Prvulovic, "One&done: A single-decryption em-based attack on openssl's constant-time blinded rsa," in *Proceedings of the 27th USENIX Conference on Security Symposium*. USENIX Association, 2018, pp. 585–602.

[12] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics," *Digital Investigation*, vol. 29, pp. 43–54, 2019.

[13] M. A. Elmohr, H. Liao, and C. H. Gebotys, "Em fault injection on arm and risc-v," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 206–212.

[14] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic, "Spectral profiling: Observer-effect-free profiling by monitoring em emanations," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–11.

[15] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "Eddie: Em-based detection of deviations in program execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 333–346.

[16] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead profiling via em emanations," in *Proceedings of the 25th int symposium on software testing and analysis*, 2016, pp. 401–412.

[17] E. M. Ugurlu, B. B. Yilmaz, A. Zajić, and M. Prvulovic, "Pitem: Permutations-based instruction tracking via electromagnetic side-channel signal analysis," *IEEE Transactions on Computers*, vol. 71, no. 5, pp. 1156–1169, 2021.

[18] A. Kacmarcik and M. Prvulovic, "Securing cps through simultaneous analog side-channel monitoring of cyber and physical domains," *IEEE Access*, vol. 12, pp. 126 717–126 728, 2024.

[19] "RTL-SDR: Software Defined Radio," 2024, accessed: 2024-06-26. [Online]. Available: https://www.rtl-sdr.com/about-rtl-sdr/

[20] AARONIA PBS, https://www.tequipment.net/Aaronia/PBS1-5/Standard/Passive-Oscilloscope-Probes/?rrec=true.

[21] F. T. Werner, J. Dinkić, D. Olćan, A. Djordjević, M. Prvulović, and A. Zajić, "An efficient method for localization of magnetic field sources that produce high-frequency side-channel emanations," *IEEE Transactions on Electromagnetic Compatibility*, vol. 63, no. 6, pp. 1799–1811, 2021.

[22] HackRF One: Software Defined Radio, https://greatscottgadgets.com/hackrf/.

[23] Siglent SSA3000X-R Real-Time Spectrum Analyzer, https://siglentna.com/spectrum-analyzers/ssa3000x-r/.

[24] Ettus Research, "USRP B210: SDR Platform," https://www.ettus.com/all-products/ub210-kit/, 2024, accessed: 2024-06-26.