

When Faults Don't Vanish: Persistent Fault Injection and Key Recovery on MRAM-Backed AES

Brojogopal Sapui, Priyanjana Pal, Mehdi B.Tahoori

Institute of Computer Engineering, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

{brojogopal.sapui, priyanjana.pal, mehdi.tahoori}@kit.edu

Abstract—Spin-Transfer Torque MRAM (STT-MRAM) is gaining popularity as a leading non-volatile memory (NVM) for embedded, IoT, and automotive systems, owing to its low leakage, high endurance, and compatibility with CMOS processes. However, its magnetic nature and non-volatility feature introduce unique fault behaviors that differ fundamentally from conventional volatile memories such as SRAM and DRAM. In particular, faults injected during MRAM write operations may persist across power cycles, enabling attackers to exploit stable key corruptions. In this work, we present a persistent fault analysis (PFA) framework targeting AES implementations where the round-key schedule is stored in STT-MRAM. We demonstrate how carefully timed voltage glitches during MRAM write cycles can create reproducible, persistent bit flips that propagate through the AES key schedule. These persistent corruptions significantly reduce the ciphertext requirements for differential fault analysis (DFA) and enable statistical persistent fault analysis (SPFA) with only 12–17 faulty ciphertexts. These findings highlight that MRAM-based systems are exposed to a persistent-fault threat model different from transient faults in volatile memories, with direct implications for secure key storage and cryptographic implementations.

I. INTRODUCTION

Emerging non-volatile memories (NVMs) are increasingly considered as promising alternatives to conventional SRAM and DRAM in embedded, IoT, automotive, and storage applications. Among the available technologies, Spin-Transfer Torque Magnetic RAM (STT-MRAM) has gained particular attention due to its high endurance, fast read/write speeds, and near-zero standby power. Its compatibility with CMOS processes and scalability further make it attractive for both embedded and large-scale computing platforms. For cryptographic applications, MRAM is often chosen as a candidate for secure key storage because it combines the speed of SRAM with the non-volatility of Flash, while avoiding endurance limitations of other NVMs such as ReRAM or PCM. [1], [2].

Despite advantages, MRAM introduces critical security concerns with new attack surfaces. Prior works have examined side-channel leakage and reliability failures under environmental stress [3], [4], while fault analysis has mainly focused on volatile memories such as SRAM and DRAM [5], [6]. In these volatile memories, injected faults vanish after reset, whereas MRAM retains them magnetically, making injected faults persistent until overwritten. Similar fault injection studies on non-volatile flash memories have shown vulnerability to laser and voltage glitch attacks [7], [8], yet flash is constrained by block-level and slow writes. In contrast, STT-MRAM enables fast bit-level writes, making persistent faults particularly practical.

While MRAM's persistence is recognized as a risk, its impact on cryptographic fault analysis has not been systematically

studied. It remains unknown whether perturbation of carefully controlled memory-access cycles can produce reproducible, byte-localized faults that remain stable across power cycles. Moreover, the effect of STT-MRAM's switching behavior on fault characteristics has not been linked to cryptanalysis. Understanding how these device-level properties translate into exploitable faults in AES is the central motivation of this work.

To address this, we present a practical framework for persistent fault injection during MRAM write cycles in cryptographic key storage. By applying carefully timed voltage glitches during write operations, we induce deterministic faults in cells storing AES round keys. This method is enabled by electrically isolating the MRAM supply rail from the system core, ensuring the rest of the platform remains unaffected. We then show how these persistent corruptions propagate across the AES key schedule and can be exploited using both Differential Fault Analysis (DFA) and Statistical Persistent Fault Analysis (SPFA) to recover the secret key. While prior MRAM security studies have largely relied on simulation [9], [10] or device-level characterization that does not demonstrate cryptographic key recovery, our work is, to the best of our knowledge, the first experimental demonstration on a commercial MRAM device leading to complete AES-128 key recovery using fault analysis. Our contributions are twofold: (i) an experimental setup enabling targeted persistent MRAM fault injection, and (ii) a demonstration of AES-128 key recovery with fault analysis.

Our experiments reveal an asymmetry in MRAM write-cycle vulnerability: transitions from the parallel (P) to the anti-parallel (AP) state are significantly more fault-prone than AP→P transitions. This asymmetry highlights the physical origins of MRAM fault behavior and explains why induced errors can remain stable across multiple reads and power cycles. Leveraging this persistence, we demonstrate that AES-128 keys stored in MRAM can be recovered with only 12–17 faulty ciphertexts, compared to the tens of ciphers typically required in volatile-memory attacks [11], [12]. These findings confirm that persistent MRAM faults represent a practical and highly efficient attack vector for secure key storage.

The rest of the paper is as follows. Sec. II reviews technical background with related work. Sec. III and Sec. IV describe the threat model, the MRAM fault model, and the fault injection methodology. Sec. IV-C presents the proposed attack framework on the AES key schedule and details the mathematical propagation of persistent faults to recover secret keys. Sec. V reports the experimental results and complexity analysis with a detailed discussion. Finally, Sec. VI concludes the paper.

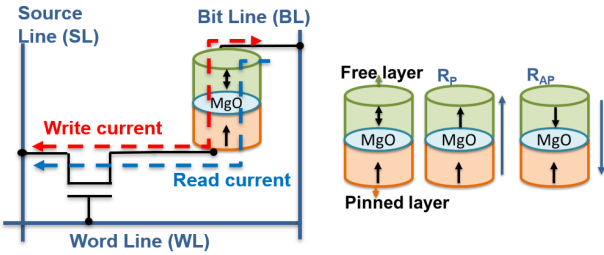


Fig. 1: STT-MRAM cell structure and operation principle, where data is stored in the relative magnetization of free and pinned layers.

II. BACKGROUND

A. STT-MRAM Basics

Spin-Transfer Torque MRAM (STT-MRAM) employs a magnetic tunnel junction (MTJ) as its core storage element [9], [10], [13]. An MTJ consists of two ferromagnetic layers separated by a thin oxide barrier, as shown in Figure 1. The pinned (reference) layer has fixed magnetization, while the free layer can switch its orientation under an applied write current. When the two layers are parallel, the cell exhibits low resistance (R_P); when anti-parallel, it shows high resistance (R_{AP}). A small sensing current during read distinguishes these states without altering the data.

A typical STT-MRAM array follows the 1T-1MTJ structure, where each cell integrates one MTJ with an access transistor. Peripheral CMOS circuitry supports reliable array-level operation. This hybrid design combines the non-volatility of MTJs with the scalability of CMOS, enabling dense, low-power memory arrays. Moreover, its compatibility with standard fabrication processes makes STT-MRAM a strong candidate for embedded memory storage applications.

B. AES and Fault Analysis

The Advanced Encryption Standard (AES) is the most widely used block cipher, standardized by NIST [14]. It encrypts 128-bit blocks using keys of 128, 192, or 256 bits, with rounds composed of *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. Differential Fault Analysis, introduced by Biham and Shamir and later adapted to AES [11], [15], exploits faulty ciphertexts to derive algebraic relations on subkeys, often recovering AES-128 with only a few faulty pairs. While DFA typically assumes *transient* faults injected per encryption, Persistent Fault Analysis [16] instead leverages *permanent* modifications in the algorithm or key schedule, allowing the adversary to observe many faulty encryptions after a single fault injection. Statistical extensions, SPFA [17], further exploit biases introduced by such faults, enabling efficient key recovery even under multiple or unknown persistent faults.

C. Related Work

Early studies on STT-MRAM focused on reliability and endurance under process, voltage, and temperature variations, largely through simulation, and reported error modes tied to write-current margins and retention under stress [4], [18]. Security-oriented works then examined information leakage in MRAM-based systems, such as access-pattern timing and

power correlations [3]. Hardware-based demonstrations have also shown that external magnetic fields can bias switching probabilities and corrupt stored data in commercial pMTJ chips [19], [20], confirming MRAM’s susceptibility to attacks.

Most prior MRAM fault injection studies therefore target the device level, relying on magnetic fields or environmental stress [18]–[20]. While effective, these methods require specialized equipment and close physical access, limiting their practicality in commodity board-level scenarios. Cryptographic studies with MRAM have instead focused on side-channel leakage, e.g., correlation power analysis [21], which recovers keys through observation but does not corrupt stored states.

PFA has been explored algorithmically and on SRAM devices [16], [17], but no prior work has demonstrated in hardware that MRAM write-cycle faults can persist and propagate through a cryptographic key schedule. Existing MRAM security research is either simulation-based or limited to magnetic disturbance and leakage, without establishing a practical, board-level adversary model. Our work fills this gap by presenting the hardware demonstration of board-level voltage-glitch fault injection during MRAM write cycles, inducing reproducible persistent faults that propagate through the AES key schedule and enable cryptanalysis.

III. THREAT MODEL

We consider a *non-invasive, board-level* adversary with physical access to a device where STT-MRAM stores AES keys. The attacker cannot (i) decapsulate or probe the MRAM, (ii) alter the AES/control logic, or (iii) directly read MRAM cells. They can, however, reset the board, interact with standard I/O (UART/DMA/AXI-Stream), and manipulate the MRAM power rail at PCB level.

A. Capabilities

- **Voltage glitching:** Inject transient droops on V_{MRAM} via an external switch, without disturbing the FPGA core.
- **I/O access:** Supply chosen plaintexts and observe ciphertexts through normal interfaces, without internal state visibility.
- **Timing:** Align glitches to MRAM writes using coarse external triggers (GPIO/LED/serial activity); no internal probes are assumed.

B. Limitations

- No EM, magnetic, thermal, or laser injection—only voltage-glitch-induced write faults are considered.
- No invasive MRAM readout or hardware modification.
- No prior knowledge of which MRAM cells fault; persistence reuses whatever corrupted pattern occurs.

C. Objective

The attacker aims to recover the AES-128 key by inducing *persistent* bit faults in the MRAM-resident expanded key schedule and reusing the stable corrupted key across encryptions. Success is defined as recovering the key via DFA/SPFA from correct/faulty ciphertexts, assuming persistence across resets until overwritten.

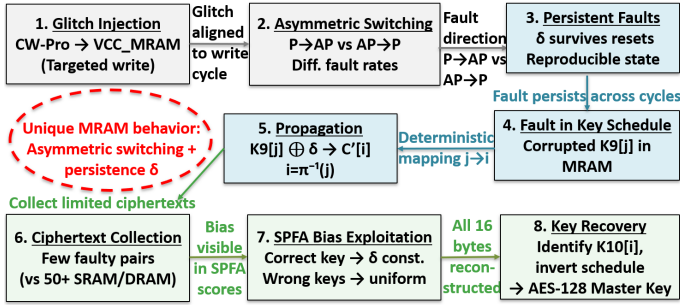


Fig. 2: Overview of the proposed Statistical Persistent Fault Analysis (SPFA) attack on AES-128 using STT-MRAM. A voltage glitch during write induces asymmetric switching with persistent faults, which propagate through the key schedule. Limited faulty ciphertexts are sufficient for bias exploitation, enabling recovery of all 16 key bytes and reconstruction of the AES-128 master key.

IV. ATTACK METHODOLOGY

A. Fault Attack

In our attack model, the AES-128 keys are stored in STT-MRAM, and faults are injected during write operations. Unlike transient state corruptions in the cipher, such faults directly alter round keys, which are then repeatedly used in all subsequent encryptions. A single corrupted round key therefore propagates deterministically through the AES rounds, producing a structured and reusable fault model. This persistence sharply contrasts with SRAM or DRAM, where injected faults vanish after reset and must be reinjected for each encryption. As a result, MRAM transforms even a single fault event into a long-lasting corruption that yields many faulty ciphers, significantly reducing the data complexity required for DFA and SPFA. The overall attack method is illustrated in Figure 2.

The critical requirement for exploiting this attack is precise control of the MRAM write process. As bit flips occur during magnetic switching of the MTJ free layer, an effective adversary must align fault injections with the narrow commit window of write operations. The following subsection details our methodology for achieving this alignment, including the timing model, trigger synchronization, and glitch parameterization.

B. Fault Injection Methodology

To realize the fault attack described above, the adversary must precisely synchronize voltage glitches with the MRAM write process. Unlike volatile memories, where timing violations or charge disturbance can flip bits during reads or refresh cycles, STT-MRAM is fault-sensitive only during writes, when the free layer of the MTJ undergoes magnetic switching. Figure 3 illustrates the timing of MRAM operations. The write cycle is longer than a read and includes a distinct *commit window* in which spin-transfer torque determines the final magnetic state. Perturbations outside this interval may disturb bus traffic, but do not alter the stored value.

We therefore model the write operation in three phases: (i) command and address transmission over the SPI bus, (ii) data byte latching into the input buffer, and (iii) magnetic commit. Only the third phase is fault-sensitive, as the MTJ orientation is being switched and is not yet stable. Glitches overlapping this window bias the switching probability, either suppressing or

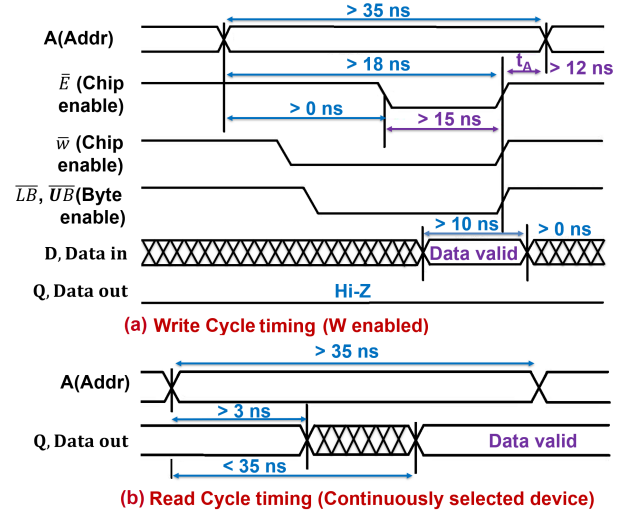


Fig. 3: Timing diagrams at the device interface, showing longer write cycle requirements compared to read access. The extended write window makes write operations more susceptible to precisely timed glitch injection, which is leveraged in our fault attack model.

accelerating the intended P \rightarrow AP or AP \rightarrow P transition, and thus provide a deterministic mechanism to inject persistent faults.

To exploit this property, our methodology aligns voltage glitches with sub-cycle precision relative to the commit window. We generate dedicated trigger markers at the start of each data write, enabling glitch offsets to be swept with 20–50 ns resolution. The offset parameter determines the alignment of the disturbance with the switching instant, the width controls how long the supply rail is suppressed, and the depth modulates the effective switching current seen by the MTJ. By systematically varying these parameters, we map the temporal fault vulnerability profile of the MRAM. This profiling reveals a narrow, tens-of-nanoseconds-wide window where faults occur with the highest probability. Once identified, this window can be repeatedly targeted to deterministically reproduce identical corrupted states across multiple executions, in contrast to transient or probabilistic errors in SRAM and DRAM.

The explored glitch parameters include offset relative to the trigger (–200 ns to +800 ns), width (5–200 ns), and depth (10–35% of V_{CC_MRAM}). Each configuration is applied over at least 1,000 repetitions to derive empirical fault probabilities. Separate campaigns are performed for P \rightarrow AP and AP \rightarrow P transitions, since STT-MRAM is known to exhibit asymmetric switching thresholds. By directly coupling this device-level asymmetry to the cryptographic fault model, our methodology extracts an attacker advantage that does not exist in prior SRAM or DRAM fault injection studies.

To confirm persistence, we define a valid magnetic-state fault using three criteria: (i) the same corrupted value must be observed across at least 100 consecutive reads, (ii) the corruption must remain stable across multiple full power cycles, and (iii) the fault must be cleared by a clean overwrite. These conditions ensure that the observed errors originate from non-volatile state corruption rather than bus-level disturbances.

Finally, we characterize persistent faults along two dimensions. In the spatial domain, glitches typically corrupt 1–4

contiguous bytes, with 1–3 flipped bits per byte, which in our AES mapping corresponds to multiple adjacent round-key entries. In the temporal domain, faults occur most frequently when glitches overlap the final switching transitions of a data byte in the commit phase. Notably, P→AP writes exhibit up to 3× higher susceptibility than AP→P, reflecting the underlying asymmetry of spin-transfer switching currents. This bias validates the physical origin of the faults and gives attackers a predictable handle to strengthen DFA and SPFA.

C. Persistent Fault Analysis

Once persistent faults are injected into the AES key schedule, the device continues to produce faulty ciphertexts with no further intervention. The attacker can therefore collect correct ciphertexts from the unmodified state and faulty ciphertexts from the corrupted state, with the guarantee that the same difference persists across all encryptions until explicitly overwritten. This persistence eliminates the need for precise timing in each run and drastically reduces the number of faulty pairs required compared to transient DFA on volatile memories.

1) Notation

We consider AES-128 with round keys K_0, \dots, K_{10} . Let S denote the AES S-box, S^{-1} its inverse, and π the permutation of byte indices induced by `ShiftRows`. All operations are over \mathbb{F}_2^8 , with XOR as addition. For a byte index i , we write $x[i]$ for the i -th byte of state x . A persistent fault corresponds to a fixed nonzero difference $\delta \in \mathbb{F}_2^8$ injected into some round key byte, present across all encryptions.

2) Case A: Fault in the final round key K_{10}

Suppose a persistent byte fault affects $K_{10}[i]$:

$$K'_{10}[i] = K_{10}[i] \oplus \delta, \quad \delta \neq 0.$$

Then for any plaintext P with correct ciphertext C and faulty ciphertext C' :

$$C'[i] = C[i] \oplus \delta, \quad C'[j] = C[j] \quad (j \neq i).$$

The difference $\Delta C[i] = \delta$ directly exposes the corrupted byte. Because the same δ appears in every encryption, an attacker can trivially confirm the difference across multiple plaintexts and recover $K_{10}[i]$ with standard last-round key tests. This case illustrates the simplest form of persistent DFA.

3) Case B: Fault in the second last round key K_9

More interestingly, suppose $K_9[j]$ is faulted:

$$K'_9[j] = K_9[j] \oplus \delta.$$

At the input to round 10,

$$u'[j] = u[j] \oplus \delta, \quad u'[t] = u[t] \quad (t \neq j).$$

After the final round, only the ciphertext byte $i = \pi^{-1}(j)$ is affected:

$$C'[i] = S(u[j]) \oplus K_{10}[i], \quad C'[i] = S(u[j] \oplus \delta) \oplus K_{10}[i].$$

Eliminating $u[j]$ gives the DFA test equation:

$$\begin{aligned} T_k(C[i], C'[i]) &= S^{-1}(C[i] \oplus k) \oplus S^{-1}(C'[i] \oplus k) \\ &= \delta \iff k = K_{10}[i]. \end{aligned} \quad (1)$$

Correct k makes T_k constant across pairs; wrong keys yield near-uniform values. Persistence guarantees that all faulty pairs share the same δ , enabling extremely low data complexity.

4) Recovery Algorithm

Given m correct/faulty pairs $(C^{(t)}, C'^{(t)})$, the attacker does:

- For each hypothesis $k \in \{0, \dots, 255\}$, compute $T_k(C^{(t)}[i], C'^{(t)}[i])$ for all t .
- Score k by the constancy of the resulting multiset.
- Output $\hat{K}_{10}[i]$ as the maximizer; δ is the constant difference.

Complexity: $\Pr[T_k \text{ constant across } m \text{ pairs}] \approx 2^{-8(m-1)}$ for a wrong key k . Thus $m = 2$ pairs leave only ≈ 1 false candidate, while $m = 3$ already makes errors negligible (2^{-16}). Persistence means each new plaintext provides another valid pair without requiring reinjection.

5) Multiple-Byte Faults and SPFA

If multiple bytes of K_9 are faulted (e.g., from a wider MRAM glitch), several ciphertext bytes differ simultaneously. The same per-byte test applies independently, but biases may overlap. In this case, SPFA provides robustness by scoring hypotheses by consistency over the faulty distribution:

$$\text{score}(k) = \sum_{t=1}^m 1 \left[T_k(C^{(t)}[i], C'^{(t)}[i]) \neq 0 \right],$$

optionally weighted by the physical likelihood of δ (e.g., P→AP dominates). Because all encryptions share the same persistent bias, even a small m suffices to reveal the correct key.

6) From K_{10} to the master key.

Once K_{10} is recovered bitwise, the AES-128 key schedule can be inverted to obtain K_0 . Write K_r as four 32-bit words $W_{4r}, W_{4r+1}, W_{4r+2}, W_{4r+3}$. For $r = 9, \dots, 0$,

$$W_{4r+3} = W_{4(r+1)+3} \oplus W_{4(r+1)+2},$$

$$W_{4r+2} = W_{4r+3} \oplus W_{4(r+1)+1},$$

$$W_{4r+1} = W_{4r+2} \oplus W_{4(r+1)+0},$$

$$W_{4r} = W_{4r+1} \oplus \text{SWord}(\text{RWord}(W_{4(r+1)+3})) \oplus \text{Rcon}_{r+1}.$$

This recurrence enables the recovery of the master key K_0 in $O(1)$ time. Here, `SWord`(\cdot) denotes the application of the AES S-box substitution to each byte of the word, and `RWord`(\cdot) denotes the one-byte cyclic rotation of a 32-bit word.

7) Practical Targeting

Targeting K_{10} provides trivial $\Delta C = \delta$ but limited algebraic checks. Targeting K_9 enables stronger equations such as (1), and in practice only few pairs suffice for full key recovery. MRAM persistence ensures reproducibility across encryptions, while its P→AP asymmetry further reduces the effective hypothesis space.

V. RESULTS AND DISCUSSION

A. Experimental Setup

1) Device Level

Experiments were conducted on a Pynq-Z1 FPGA (100 MHz fabric) connected to an external SPI-based STT-MRAM via a PMOD header. The MRAM supply (V_{MRAM}) was electrically isolated from the global 3.3 V rail (3V3_SYS) using a

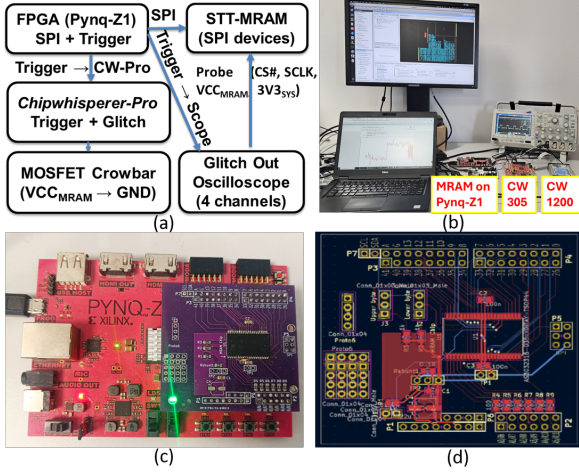


Fig. 4: Experimental setup for voltage glitching attacks on STT-MRAM. (a) System-level architecture showing FPGA (Pynq-Z1), ChipWhisperer-Pro, and oscilloscope connections for glitch injection and monitoring. (b) Laboratory setup with MRAM mounted on Pynq-Z1, ChipWhisperer CW305, and CW1200. (c) Hardware implementation with MRAM daughterboard interfaced to the Pynq-Z1 FPGA. (d) PCB layout of the custom MRAM daughterboard.

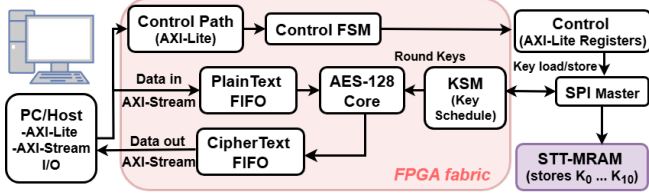


Fig. 5: FPGA-based AES-128 encryption architecture with STT-MRAM for secure key storage. The host communicates via AXI interfaces, while the FPGA fabric handles plaintext/ciphertext buffering, AES core execution, and key scheduling. Round keys are loaded and stored in STT-MRAM through an SPI master interface.

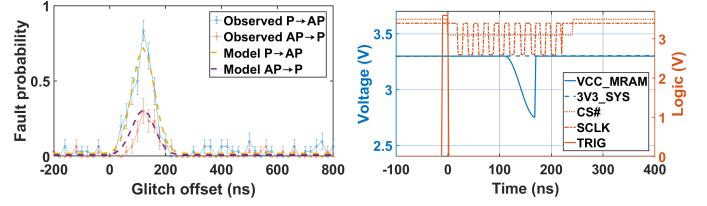
small series element (1–2 Ω resistor or ferrite bead). A low- $R_{DS(on)}$ MOSFET, controlled by the ChipWhisperer-Pro [22] glitch module, was placed between V_{MRAM} and ground to create transient supply droops. This isolation allows targeted perturbations of the MRAM without destabilizing the FPGA or other peripherals.

Local decoupling capacitors (100 nF + 10 nF) were retained near the MRAM package to preserve nominal stability while still enabling nanosecond-scale voltage glitches. All critical nodes (V_{MRAM} , 3V3_SYS, SPI CS#, SPI SCLK, trigger outputs) were monitored with a Tektronix MSO 2024B oscilloscope for timing validation. The whole setup is illustrated in Figure 4. This setup enables reproducible persistent faults during MRAM writes, unlike prior global-supply faulting studies which typically induced transient or unstable behavior.

2) Board Level

Figure 5 illustrates the AES system with STT-MRAM for key storage. Host communication occurs over AXI-Stream, with I/O FIFOs buffering plaintexts and ciphertexts. A lightweight Control FSM (through AXI-Lite) manages key loading, encryption, and data transfer. This modular structure separates data, key, and control paths, facilitating isolation of fault effects with minimal overhead. The AES core itself is unmodified.

To externalize the key schedule and expose a faultable



(a) Glitch window characterization: (b) CS#, SCLK, and TRIG traces measured fault probability as a function of glitch offset for P→AP and MRAM write commit window. All critical signals (V_{MRAM} , 3V3_SYS, and model fits are shown, highlighting SPI CS#, SPI SCLK, TRIG) are monitored using a Tektronix MSO 2024B oscilloscope to validate alignment.

Fig. 6: Empirical characterization and validation of optimal glitch injection timing in MRAM writes.

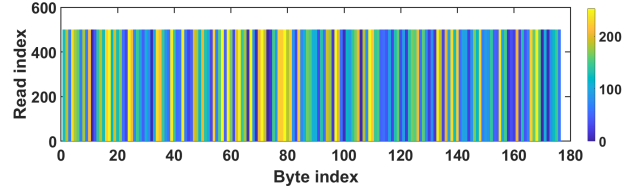


Fig. 7: Persistence verification: key schedule bytes across 100 reads and five power cycles. Persistent faults appear as stable stripes.

memory interface, all AES round keys are stored in STT-MRAM rather than on-chip BRAM. A Key-Schedule Manager (KSM) issues read/write requests to an SPI Master and forwards returned bytes to the AES core at round boundaries. This mirrors realistic deployments where MRAM serves as secure key storage, introduces only modest FPGA resource overhead, and adds a small latency dominated by SPI transactions. Critically, supply perturbations during MRAM write events create *persistent* key faults that survive resets and are reused across encryptions, enabling practical PFA.

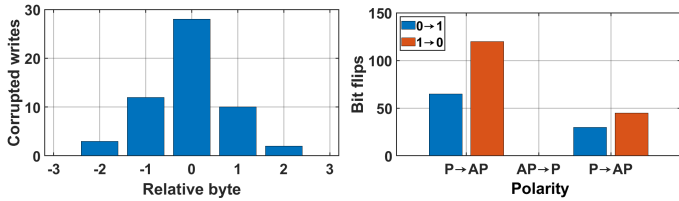
B. Attack Evaluation

1) Timing Verification

Scope captures (Figure 6a) confirm that V_{MRAM} can be selectively perturbed by 350–700 mV for 10–80 ns while the global 3V3_SYS rail remains stable within <20 mV. Faults only occur when glitches overlap the commit window of a write operation, validating the effectiveness of our rail isolation design. In contrast, prior SRAM/DRAM fault injection studies [23] report that perturbing the global supply often destabilizes the device, causing resets or non-reproducible errors. Our selective approach ensures that observed faults are intrinsic to MRAM write dynamics rather than generic board-level instability.

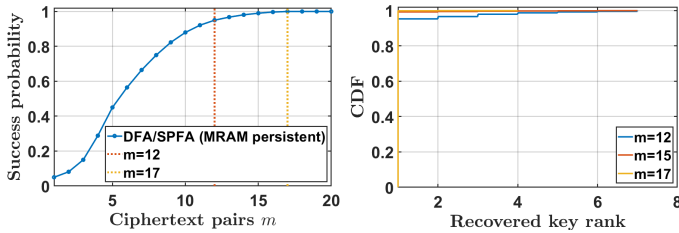
2) Glitch Window Characterization

The measured glitch window (Figure 6b) is narrow (\approx 40–60 ns wide) and centered around +120 ns after the trigger, aligning with the MRAM commit phase. Notably, P→AP transitions are more susceptible, with peak fault probabilities of 55–70% compared to 20–35% for AP→P. This asymmetry reflects the differing critical currents of spin-transfer torque switching. By comparison, SRAM/DRAM faults induced under voltage stress typically show near-symmetric distributions of 0→1 and 1→0 flips [24]. The observed bias therefore highlights a unique vulnerability of MRAM-based designs.



(a) Persistent faults: spatial locality. Relative index 0 marks the targeted byte, while ± 1 denotes 7510 immediate neighbors. Most faults occur at the target, with only minor spillover. (b) Persistent faults: bit-flip polarity. Logic ‘0’ maps to the P state and logic ‘1’ to the AP state. A strong bias toward $1 \rightarrow 0$ flips (AP \rightarrow P) is observed, reflecting asymmetric MTJ switching.

Fig. 8: Spatial/polarity characterization of persistent MRAM faults.



(a) DFA per-byte success probability shows that 12 pairs suffice for reliable recovery. (b) Key-rank CDF for 1-byte of K_{10} shows that 12 pairs suffice for reliable recovery. With $m=17$, rank-1 is obtained in $>99\%$ cases.

Fig. 9: Success probability and key-rank evaluation of DFA/SPFA attacks under persistent MRAM faults.

3) Persistence Verification

Figure 7 demonstrates that injected faults remain stable across 100 consecutive reads and at least five power cycles, until explicitly re-written. Out of 50 glitch campaigns, 42 produced persistent faults, with multi-byte bursts (2–4 bytes) in $\sim 6\%$ of cases. Thus, MRAM’s non-volatility enables adversaries to induce a fault once and reuse it across arbitrary encryptions. This persistence drastically reduces attacker effort by eliminating the need for repeated glitching campaigns.

4) Fault Morphology

Most persistent faults affect 1–3 bits in a single byte, with occasional spillover into adjacent bytes (Figure 8a). Bit-flip polarity analysis (Figure 8b) reveals a strong bias toward $1 \rightarrow 0$ flips, i.e., P \rightarrow AP transitions, consistent with asymmetric MTJ switching currents. In conventional DFA on SRAM/DRAM, bit flips are usually assumed to occur uniformly at random [11], [15], offering the attacker less structural information. Here, the polarity bias provides an exploitable advantage by reducing the effective hypothesis space.

5) AES Key Recovery Performance

Persistent faults in $K_9[j]$ propagate deterministically to ciphertext byte $i = \pi^{-1}(j)$. As shown in Figure 9a, two faulty ciphertext pairs suffice to identify the correct key byte with $\sim 95\%$ probability, while three pairs yield $>99.9\%$ success. The key-rank CDF in Fig. 9b confirms that false hypotheses vanish with very few pairs. By comparison, classical transient DFA on DRAM often requires tens of carefully timed fault injections to achieve similar recovery rates [23]. SPFA (Figure 10) further enhances robustness: the correct key k^* yields consistent scores, while wrong keys scatter near-uniformly, allowing recovery even under multiple or unknown persistent faults.

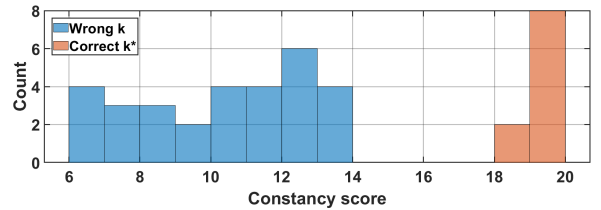


Fig. 10: SPFA score distribution: correct key k^* yields consistent δ ; wrong keys yield near-uniform scores.

TABLE I: Our work vs prior MRAM security literature

Works	Platform	Attack Class	Result (AES)
Reliability/Endurance of STT-MRAM [4], [18]	Simulation	Variation / stress analysis	N/A
Leakage in MRAM systems [3]	Device	Side-channel (timing/power)	N/A
Magnetic field bias on pMTJ MRAM [19], [20]	Device	Magnetic fault/bias	N/A
CPA on MRAM-backed crypto [21]	Device + board	Power side-channel (CPA)	key recovery 2k–5k traces
This work (MRAM write-cycle faults)	Device + board	Voltage-glitch during writes	key recovery 12–17 ciphers

6) Results Summary

Overall, these results demonstrate that persistent MRAM faults offer adversaries a qualitatively stronger primitive than transient faults in conventional volatile memories. The persistence across power cycles eliminates the need for repeated injections, the polarity bias simplifies analysis, and the deterministic propagation enables near-certain AES key recovery with only a handful of ciphertexts. Compared to SRAM/DRAM-based attacks, our results highlight the more severe risks of using MRAM. The comparative analysis with other MRAM-based security literature are summarized in Table I.

C. Countermeasures

Persistent MRAM faults demand defenses beyond transient SRAM/DRAM attacks. At the hardware level, isolating and decoupling the MRAM supply increases glitching difficulty, while rail monitors or write-verify schemes can detect or correct corruption. At the architecture level, recomputing round keys from the master key avoids persistence, while randomizing write timing or storing redundant copies with checksums hinders exploitation. In practice, a layered defense combining hardware monitors with lightweight redundancy is essential to secure against DFA/SPFA. A detailed exploration of such specialized countermeasures is left for future work.

VI. CONCLUSION

This work explored persistent fault analysis on AES when its key schedule is stored in STT-MRAM. By isolating the MRAM supply rail and using ChipWhisperer-Pro to inject timed voltage glitches, we induced reproducible faults during MRAM writes without destabilizing the FPGA core. Unlike transient faults in SRAM or DRAM, these corruptions persist across encryptions and power cycles, enabling efficient exploitation. We showed that such faults propagate through the AES key schedule and allow DFA/SPFA to recover the AES-128 key with only a few ciphertexts. This persistence reduces attack complexity and synchronization effort, underscoring MRAM as a uniquely powerful fault target and motivating the need for MRAM-specific countermeasures.

REFERENCES

- [1] J. S. Meena *et al.*, “Overview of emerging nonvolatile memory technologies,” *Nanoscale research letters*, vol. 9, p. 526, 2014.
- [2] M. N. I. Khan *et al.*, “Comprehensive study of side-channel attack on emerging non-volatile memories,” *JLPEA*, 2021.
- [3] J. Ahn *et al.*, “Security analysis of emerging nonvolatile memories: Case study on STT-RAM,” in *Proceedings of the 55th Annual Design Automation Conference (DAC)*, 2018, pp. 1–6.
- [4] X. Wang *et al.*, “Reliability of STT-MRAM under environmental stress,” in *IEEE International Reliability Physics Symposium (IRPS)*, 2018, pp. 2B.3–1–2B.3–6.
- [5] Y. Kim *et al.*, “Rowhammer: Reliability analysis and security implications,” *IEEE Micro*, vol. 36, pp. 114–122, 2016.
- [6] S. Skorobogatov, “Semi-invasive attacks – a new approach to hardware security analysis,” in *Cryptographic Hardware and Embedded Systems (CHES)*, 2005, pp. 1–29.
- [7] —, “Optical fault masking attacks,” in *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2010, pp. 23–29.
- [8] M. Hutter *et al.*, “The temperature side channel and heating fault attacks,” in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2013, pp. 219–235.
- [9] W. Zhao *et al.*, “Spintronics for low-power computing,” *Proceedings of the IEEE*, vol. 104, pp. 1782–1799, 2016.
- [10] —, “Spintronics for low-power computing,” *Proceedings of the IEEE*, vol. 104, pp. 1782–1799, 2016.
- [11] G. Piret *et al.*, “A differential fault attack technique against SPN structures, with application to the AES and KHAZAD,” in *Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2003, pp. 77–88.
- [12] C. Giraud, “Dfa on aes,” *Advanced Information Security*, vol. 2, pp. 27–41, 2004, also presented at FDTC 2004.
- [13] K. Wang *et al.*, “Compact modeling of STT-MRAM for reliability analysis and design exploration,” *IEEE Transactions on Electron Devices*, vol. 60, pp. 2226–2233, 2013.
- [14] National Institute of Standards and Technology (NIST), “FIPS 197: Advanced Encryption Standard (AES),” Federal Information Processing Standards Publication 197, 2001. [Online]. Available: <https://doi.org/10.6028/NIST.FIPS.197>
- [15] P. Dusart *et al.*, “Differential fault analysis on AES,” in *Applied Cryptography and Network Security (ACNS)*. Springer, 2003, pp. 293–306.
- [16] F. Zhang *et al.*, “Persistent fault analysis on block ciphers,” in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. Lecture Notes in Computer Science, vol. 1102. Springer, 2018, pp. 655–684.
- [17] —, “Statistical persistent fault analysis,” in *Advances in Cryptology – ASIACRYPT 2020*, ser. Lecture Notes in Computer Science, vol. 12491. Springer, 2020, pp. 87–115.
- [18] Z. Fang *et al.*, “Reliability issues of STT-MRAM and prospective solutions: A review,” *IEEE Transactions on Device and Materials Reliability*, vol. 18, pp. 3–15, 2018.
- [19] M. Barrera *et al.*, “Magnetic fault injection attacks on STT-MRAM in 40 nm technology,” in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2019, pp. 151–158.
- [20] L. Goubin *et al.*, “Fault attacks on commodity MRAM memories,” in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020, pp. 75–82.
- [21] A. Chakraborty *et al.*, “Power analysis attacks on STT-MRAM based cryptographic implementations,” in *Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2017, pp. 167–187.
- [22] A. Shylendra *et al.*, “Fault attack detection in aes by monitoring power side-channel statistics,” in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 219–224.
- [23] Z. Al-Ars *et al.*, “Transient faults in drams: Concept, analysis and impact on tests,” in *Proceedings 2001 IEEE International Workshop on Memory Technology, Design and Testing*. IEEE, 2001, pp. 59–64.
- [24] Y. Kim *et al.*, “Flipping bits in memory without accessing them: An experimental study of dram disturbance errors,” *ACM SIGARCH Computer Architecture News*, vol. 42, pp. 361–372, 2014.