Robust LFSR-based Scrambling to Mitigate Stencil Attack on Main Memory

GAURAV KUMAR, Electrical Engineering, Indian Institute of Technology Jammu, Jammu, India
KUSHAL PRAVIN NANOTE, Electrical Engineering, Indian Institute of Technology Jammu, Jammu, India
SOHAN LAL, Massively Parallel Systems Group, Hamburg University of Technology, Hamburg, Germany
YAMUNA PRASAD, Computer Science and Engineering, Indian Institute of Technology Jammu, Jammu, India
SATYADEV AHLAWAT, Electrical Engineering, Indian Institute of Technology Jammu, Jammu, India

Main memory plays a pivotal role in the storage of computational data in a wide range of applications, including highly sensitive assets such as banking transactions, cryptographic keys, and user credentials. However, memory systems remain vulnerable to advanced physical and side-channel attacks, including cold boot attacks that exploit residual data after power-down. To mitigate such risks, Intel's DDR3 memory scrambler uses a Linear Feedback Shift Register (LFSR)-based stream cipher to obscure memory contents. Nevertheless, this mechanism has been shown to be susceptible to stencil attack, a cold boot technique that reconstructs the scrambling key by leveraging the linear and periodic nature of the keystream. This article proposes a novel, lightweight, and secure scrambling architecture based on a generic LFSR designed to enhance the security of DDR3 memory against cold boot attacks. The proposed generic LFSR-based mechanism eliminates differential keystream periodicity by introducing an address- and seed-dependent LFSR structure, thereby rendering differential key recovery techniques computationally infeasible. Furthermore, unlike traditional AES-based memory encryption that incurs high latency and area overhead, the proposed approach achieves comparable security guarantees with low hardware complexity and zero access latency. The hardware implementation results on the Xilinx VCU118 FPGA show that the proposed scheme consumes only 252 LUTs, 256 registers and 104 slices, comparable to the Intel DDR3 scrambler, while offering superior resilience against the cold boot, warm boot, and probing attacks. These results demonstrate the practicality of the proposed scheme for secure memory systems in resource-constrained environments.

CCS Concepts: • **Security and privacy** → *Embedded systems security*; **Side-channel analysis and countermeasures**; • **Computer systems organization** → *Embedded hardware;*

Additional Key Words and Phrases: DRAM security, LFSR, memory scrambling, memory disclosure attacks, stencil attack

Authors' Contact Information: Gaurav Kumar, Electrical Engineering, Indian Institute of Technology Jammu, Jammu, jammu and kashmir, India; e-mail: gaurav.kumar@iitjammu.ac.in; Kushal Pravin Nanote, Electrical Engineering, Indian Institute of Technology Jammu, Jammu, jammu and kashmir, India; e-mail: 2023pvl0091@iitjammu.ac.in; Sohan Lal, Massively Parallel Systems Group, Hamburg University of Technology, Hamburg, HH, Germany; e-mail: sohan.lal@tuhh.de; Yamuna Prasad, Computer Science and Engineering, Indian Institute of Technology Jammu, Jammu, jammu and kashmir, India; e-mail: yamuna.prasad@iitjammu.ac.in; Satyadev Ahlawat, Electrical Engineering, Indian Institute of Technology Jammu, Jammu, jammu and kashmir, India; e-mail: satyadev.ahlawat@iitjammu.ac.in.

ACM 1539-9087/2025/09-ART102
https://doi.org/10.1145/3758321

ACM Reference Format:
Gaurav Kumar, Kushal Pravin Nanote, Sohan Lal, Yamuna Prasad, and Satyadev Ahlawat. 2025. Robust LFSR-based Scrambling to Mitigate Stencil Attack on Main Memory. *ACM Trans. Embedd. Comput. Syst.* 24, 5s, Article 102 (September 2025), 22 pages. https://doi.org/10.1145/3758321

1 Introduction

Dynamic Random Access Memory (**DRAM**) serves as a critical component in modern computing systems, offering high-speed and volatile storage. It supports a wide range of applications, from personal computing and mobile devices to enterprise-grade servers and cloud infrastructure. With DRAM being responsible for storing active computational data, it frequently holds highly sensitive information such as cryptographic keys, authentication credentials, and financial records [16, 40]. This makes DRAM an increasingly attractive target for attackers seeking to exfiltrate confidential information. In recent years, various attacks have been demonstrated on DRAM-based systems, including cold boot attacks [2, 15, 21, 47, 52, 53], warm boot attack [18], row hammer attack [20] and microarchitectural side-channel attacks [13, 23, 38, 46, 48, 50]. These attacks have proven effective in undermining the confidentiality, integrity, and availability of computing platforms [5, 9, 22]. For this, these attacks leverage physical access or microarchitectural leakage to circumvent traditional security mechanisms, thereby enabling unauthorized access to sensitive data. Among these threats, cold boot attacks present a particularly severe risk, especially in scenarios where adversaries gain physical access to the system immediately following a power-down or reset. Due to the inherent data remanence property of DRAM cells, the residual charge can persist for a brief period, allowing attackers to capture memory snapshots and perform offline analysis. This enables the reconstruction of critical information, such as cryptographic keys, authentication tokens, or confidential user data, thereby compromising system confidentiality.

To counteract such threats, various countermeasures [3, 10–12, 17, 25, 26, 32, 34, 36, 44, 45] have been proposed, including physical tamper detection circuitry, device authentication, user authentication, self-encrypting memory modules, and memory controller-integrated encryption engines. Among these, one of the most widely deployed lightweight protections is the DDR3 memory scrambler [7]. This scrambling mechanism is primarily intended to decorrelate data patterns by generating a pseudo-random bitstream using a **Linear Feedback Shift Register** (**LFSR**). The generated bitstream is then XORed with the data before it is stored in DRAM. This architecture is particularly well-suited for power-constrained and performance-constrained systems, as it introduces negligible access latency and exhibits minimal hardware complexity due to the simplicity of the XOR operation. Despite its efficiency, recent studies have identified fundamental vulnerabilities in this scrambling architecture. Notably, the Stencil attack [2], an advanced variant of the cold boot attack, leverages the deterministic and periodic nature of the LFSR-generated keystream to reconstruct scrambling sequences. By analyzing the memory snapshots, the attack enables reconstruction of the keystream and subsequent recovery of the original plaintext data. The feasibility of this attack undermines the security assurances provided by conventional scrambling, highlighting its insufficiency in defending against adversaries with physical access and memory forensics capabilities. Although integrating cryptographic primitives such as AES into memory controllers [1] significantly enhances security guarantees, these methods incur high area complexity, elevated power consumption, and substantial access latency, making them impractical for performance-critical, real-time, or resource-constrained embedded systems. These constraints underscore the necessity for a secure scrambling architecture that maintains the low-latency, low-area benefits of LFSR-based implementations. At the same time, it must eliminate keystream predictability and improve resilience against the cold boot, differential analysis, and other side-channel attacks.

In this work, we propose a novel enhancement to LFSR-based memory scrambling architectures aimed at mitigating cold boot attacks by addressing the fundamental vulnerability of keystream periodicity. The proposed architecture introduces dynamic keystream generation by integrating address-dependent and seed-dependent modifications into the LFSR structure. For this, a minimal set of additional AND gates is introduced into the fixed LFSR configuration, enabling non-linear transformations that disrupt the predictable linear recurrence of traditional scramblers. These modifications disrupt the predictable linear recurrence inherent in conventional LFSR-based scramblers and eliminate repeated whitening patterns associated with fixed memory addresses. By introducing non-linear structural transformations that depend on the memory address and session seed, the proposed scheme produces a keystream, that is, both address-dependent and seed-dependent. This design disrupts differential keystream analysis and renders stencil-based cold boot attacks computationally impractical. Despite its enhanced security properties, the proposed design retains the lightweight characteristics of conventional scrambling techniques, such as minimal hardware resource utilization and zero additional access latency. Specifically, the inclusion of only a few logic AND gates ensures negligible area and latency overhead, maintaining the lightweight characteristics essential for low-power, high-performance systems. This ensures compatibility with performance-constrained systems such as embedded platforms and IoT devices. Consequently, this approach offers a balanced trade-off between computational efficiency and security robustness, addressing the limitations of existing schemes without incurring the overhead associated with full-scale cryptographic implementations. The main contributions of the papers are as follows:

- We propose a lightweight generic LFSR-based memory scrambling scheme that incorporates address-dependent and seed-dependent non-linear variations, effectively eliminating keystream periodicity and predictability.
- The architecture achieves comparable security levels to existing encryption standards while incurring minimal area overhead and zero access latency, thus making it highly suitable for resource-constrained embedded and memory systems.
- The proposed scheme is systematically evaluated for its robustness against a wide spectrum of memory-based and bus-based side-channel attacks, including stencil attack, brute-force key recovery, store-and-replay attacks, traditional cold boot attacks, warm boot attacks, and probing-based attacks.
- The security of the proposed method is rigorously evaluated using both quantitative and qualitative analyses. The quantitative evaluation employs statistical metrics to assess statistical independence and randomness. In contrast, the qualitative analysis employs visual randomness assessments to demonstrate the scheme's effectiveness in obfuscating plaintext image data.

The rest of the article is organized as follows: the existing countermeasures are explained briefly in Section 2. The existing attack and proposed methodology are explained in detail in Sections 3 and 4, respectively. Subsequently, security analysis and experimental results are presented in Sections 5 and 6, respectively. Section 7 discusses the extension of the proposed scheme to support DDR4 and DDR5 memory architectures. Finally, Section 8 concludes the study.

2 Background

Memory disclosure attacks, such as cold boot, warm boot, row hammer and other remanence-based side-channel attacks [2, 13, 15, 18, 20, 21, 23, 38, 46–48, 50, 52, 53], pose a critical threat to the confidentiality of data stored in volatile memory systems. These attacks are primarily enabled by the persistence of original or partially degraded data in DRAM cells after power-off and by the presence of compromised or inadequately secured system components such as

memory controllers. To counter such threats, various defense strategies have been proposed in the literature. These countermeasures are classified into various types: (a) Memory initialization based countermeasures [37, 42], (b) Data encryption countermeasures [1, 8, 14, 49], and (c) Memory scrambling countermeasures [7, 30].

In memory initialization based countermeasures, memory contents are systematically cleared or overwritten during system power-on (Memory zeroization [42]) or reset sequences to eliminate residual data and mitigate the risk of cold boot attacks [37]. This approach effectively reduces the data remanence window by ensuring that sensitive information is not retained in memory after the system reboot. This operation is typically performed by dedicated hardware circuits at boot time to ensure that residual data from previous sessions is eradicated. However, given the substantial capacity of contemporary DRAM modules, complete zeroization can incur significant latency, often requiring several million clock cycles, thereby delaying system startup.

Another effective approach to mitigating memory disclosure vulnerabilities is the encryption of memory data. In [8], the authors propose RegKey, a lightweight and efficient register-centric framework for implementing **Elliptic Curve Cryptography** (**ECC**) signature algorithms. Unlike traditional encryption approaches that aim to secure the entire cryptographic execution path, RegKey selectively restricts critical security-sensitive computations to the CPU register file. This architectural design guarantees that secret information never leaves the register file, thereby eliminating its exposure in the main memory. In a related advancement toward securing the memory data, MemFHE [14] is introduced as the first full-stack accelerator architecture supporting both client-side and server-side execution for modern **Homomorphic Encryption** (**HE**) schemes. Notably, MemFHE incorporates a pipelined server-side architecture with flexible bootstrapping support. This design enables adaptability across different encryption modes and security levels depending on application-specific requirements. Later, the authors in [1] employ page-level AES encryption, enabling the selective encryption of memory regions using the C-bit (the most significant bit of the physical address) in page table entries. This design allows the operating system or hypervisor to manage encrypted memory spaces efficiently. It is particularly effective in multi-tenant environments such as virtualized systems, where each **Virtual Machine** (**VM**) is assigned a unique AES key to ensure data isolation. These methods offer robust security guarantees but introduce significant overhead in terms of area, power, and access latency, which limits their applicability in resource-constrained or real-time systems. Similarly, in [49], the **AES In-Memory** (**AIM**) encryption mechanism is implemented within the memory controller to provide transparent and hardware-enforced data confidentiality for DRAM. In AIM, a dedicated AES encryption engine performs on-the-fly encryption of data during memory write operations and decrypts it during reads, ensuring that plaintext data is never stored in external memory. The AES keys used for encryption are randomly generated at each system reset and securely retained in hardware registers that remain isolated within the **System-on-Chip** (**SoC**). These registers are designed to be inaccessible to software, mitigating the risk of key exposure due to software-level attacks.

Another class of countermeasures involves the scrambling of memory contents to obscure data patterns and mitigate information leakage. Memory scrambling techniques apply lightweight, deterministic transformations, typically implemented using LFSRs, to decorrelate the stored data from its original form. The primary objective of these methods is to eliminate statistical biases and reduce the predictability of memory contents, thereby making them more resistant to simple data remanence and pattern-based attacks [30]. The work presented in [30] introduces a data scrambling scheme for securing memory transactions between the CPU and main memory. This scheme utilizes a dedicated scrambler table to generate scrambling vectors based on the memory address. During a write operation, the data is scrambled by XORing it with a vector derived from the scrambler table corresponding to the transaction address. The resulting scrambled data is

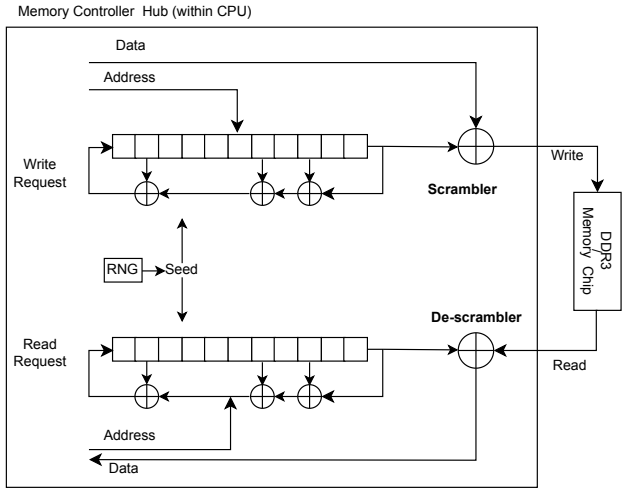


Fig. 1. Intel DDR3 scrambler structure [7].

then stored in memory along with a metadata bit, referred to as the *youth* flag, which assists in the descrambling process. During a read operation at the same address, the data is retrieved and descrambled using the previously stored youth flag in conjunction with the original address, thereby recovering the original plaintext. This approach enhances data confidentiality during transmission while maintaining low computational overhead. Later, Intel introduced an LFSR-based scrambling mechanism [7] integrated into DDR3 memory controllers to enhance memory data security with minimal performance and area overhead. This technique leverages an LFSR to generate pseudo-random sequences that are XORed with the memory data prior to storage in DRAM, thereby obfuscating the raw data patterns and reducing statistical biases. As illustrated in Figure 1, the scrambling process begins with the generation of a session-specific seed using a hardware-based **Random Number Generator** (**RNG**). This seed is then XORed with a subset of the **Least Significant Bits** (**LSBs**) of the memory transaction address (read or write) to derive the initial state of the LFSR. Once initialized, the LFSR operates using a fixed feedback polynomial, identical across all DDR3 devices, to produce a deterministic pseudo-random keystream. This keystream is subsequently XORed with the original data before transmission to the DRAM, thereby ensuring that the data is stored in a scrambled form. During read operations, the same procedure is repeated to reconstruct the original data by XORing the scrambled memory contents with the same LFSR-generated keystream using another similar LFSR. Due to its lightweight implementation, primarily consisting of XOR gates and shift registers, the Intel DDR3 scrambler introduces negligible area and latency overhead. This makes it well-suited for power-constrained and performance-constrained systems. However, the use of a fixed polynomial poses security limitations, which have been exploited in the stencil attack, explained in the subsequent section.

3 Stencil Attack

The scrambling mechanism in DDR3 memory aims to obfuscate memory content using pseudo-random sequences to mitigate data remanence and side-channel vulnerabilities. However, the security assurance provided by such mechanisms hinges on the unpredictability and entropy of the keystream. However, the authors in [2] outline a differential cold boot attack (stencil attack) that enables an adversary to recover memory contents by exploiting this inherent periodicity. The steps of stencil attack are explained below:

3.1 Memory Acquisition and Differential Keystream Extraction

To evaluate the scrambling behaviour of Intel DDR3 scrambler [7] and assess its vulnerability, the attacker conducts repeated cold boot acquisition. The process begins by powering off the system completely and allowing the DRAM to decay to a well-defined ground state. Upon reboot, the memory is immediately acquired, producing memory images that capture the scrambled data state. This procedure is repeated across multiple boot instances to collect independent memory snapshots, denoted I_0 and I_1. Each snapshot corresponds to the same underlying plaintext T, scrambled with distinct keystreams K_0 and K_1, respectively:

$$I_0 = K_0 \oplus T, \quad I_1 = K_1 \oplus T. \tag{1}$$

The bitwise XOR operation of the two snapshots eliminates the common plaintext, yielding a differential keystream (D):

$$D = I_0 \oplus I_1 = (K_0 \oplus T) \oplus (K_1 \oplus T) = K_0 \oplus K_1. \tag{2}$$

In the case of DDR3 memory, where scrambling relies on LFSR-generated keystreams, the result D typically exhibits a periodic structure, i.e., $D = S^x$, where S denotes a repeating subkey of period p, and S^x represents its replication across the address space.

3.2 Periodicity Detection and Subkey Reconstruction

To detect the periodicity p of the differential keystream D, the attacker segments D into chunks of varying sizes (e.g., powers of two from 32 to 1,024 bytes) and applies autocorrelation-based analysis. Two chunks X and Y are considered approximately equal if their normalized Hamming distance (H) remains below a pre-defined threshold ϵ:

$$X \approx Y \iff \frac{H(X \oplus Y)}{|X|} < \epsilon, \tag{3}$$

where $H(\cdot)$ denotes the Hamming weight of the XOR result and $|X|$ is the length of each bitstream. When multiple chunks satisfy this equivalence relation, they are grouped into an equivalence class:

$$\{C_0, C_1, \ldots, C_{n-1}\} \text{ such that } C_i \approx C_j, \ \forall i, j \in \{0, \ldots, n-1\}. \tag{4}$$

The correct identification of periodicity p leads to the formation of a dominant equivalence class containing several subkey candidates of length p, enabling the reconstruction of the subkey S. This reconstructed key can then be used to reverse the scrambling process and recover plaintext data, thereby breaking the DDR3 scrambling-based protection [7]. To concretely demonstrate the presence of repetition in the differential keystream, we consider a simplified illustrative example involving two different devices initialized with distinct 6-bit seeds and using two different 4-bit memory addresses, as shown in Figure 2. Although the seeds used in each session differ, the XOR of the resulting LFSR-based keystreams across devices yields an identical differential key. This repetition in the differential stream illustrates the deterministic and periodic nature of the LFSR-based scrambling mechanism. Consequently, the deterministic behaviour in the differential keystream severely undermines the security of the Intel DDR3 scrambling technique [7], enabling consistent and effective reconstruction of the original memory contents through stencil attack [2].

4 Proposed Methodology

In this section, a countermeasure against the stencil attack is presented. However, prior to the proposed countermeasure, it is essential to thoroughly examine the root cause of the stencil attack.

Device 1
Seed 1 = 110010
Address 1 = 1000
Derived Seed = 111010

Device 2
Seed 2 = 001011
Address 1 = 1000
Derived Seed = 00011

$$K_0 = 111101\ 111110\ 111111\ 011111\ 101111\ 010111\ 001011\ 100101$$
$$K_1 = 000001\ 100000\ 110000\ 111000\ 011100\ 101110\ 110111\ 111011$$
$$K_0 \oplus K_1 = 111100\ 011110\ 001111\ 100111\ 110011\ 111001\ 111100\ 011110$$

Device 1
Seed 1 = 110010
Address 2 = 1010
Derived Seed = 111000

Device 2
Seed 2 = 001011
Address 2 = 1010
Derived Seed = 00001

$$K_0 = 011100\ 101110\ 110111\ 111011\ 011101\ 001110\ 000111\ 000011$$
$$K_1 = 000001\ 100000\ 110000\ 111000\ 011100\ 101110\ 110111\ 111011$$
$$K_0 \oplus K_1 = 111100\ 011110\ 001111\ 100111\ 110011\ 111001\ 111100\ 011110$$

Fig. 2. Demonstration of differential key repetition in stencil attack [2].

4.1 Root Cause Analysis of Stencil Attack

The feasibility of the stencil attack [2] lies in the attacker's ability to extract structured and predictable differential keystreams across successive cold boot instances. This behaviour enables the adversary to analyze periodicity and subsequently reconstruct the scrambling subkey. This predictability arises from two core design characteristics of the scrambling architecture:

(1) **Static LFSR Structure:** The scrambling mechanism employs an LFSR with a fixed feedback polynomial. Although the LFSR seed is randomized during each boot through a hardware RNG, the fixed structural configuration of the LFSR leads to predictable transformations when subjected to identical transaction addresses.
(2) **Consistent Addressing Scheme:** Accessing identical memory addresses across multiple boot instances induces structural consistency in the scrambling transformations despite session-wise variation in the initial seed. This consistency arises due to the deterministic interaction between fixed address lines and the static configuration of the LFSR-based scrambler. As a result, the generated keystreams exhibit recurring patterns across sessions, leading to predictable differential keys.

The attacker leverages these vulnerabilities in the DDR3 scrambling technique [2] to determine the keystream's periodicity and reconstruct the subkey used during scrambling. This root cause underscores the need for scrambling designs that incorporate entropy not only in seeding but also within the structural behaviour of the transformation logic.

4.2 Proposed Generic LFSR

To mitigate the vulnerability exploited by the stencil attack, we propose a novel countermeasure that eliminates the predictable periodicity in the differential keystreams, thereby preventing key recovery through cold boot analysis. The central idea is to redesign the scrambling architecture by replacing the conventional fixed-structure LFSR with a generic LFSR. Unlike traditional designs that utilize a static polynomial LFSR [30] for all memory transactions, the generic LFSR dynamically adapts its feedback configuration based on the accessed memory address and the boot-time seed. At the same time, it retains fixed feedback connections at the two boundary positions of the LFSR,

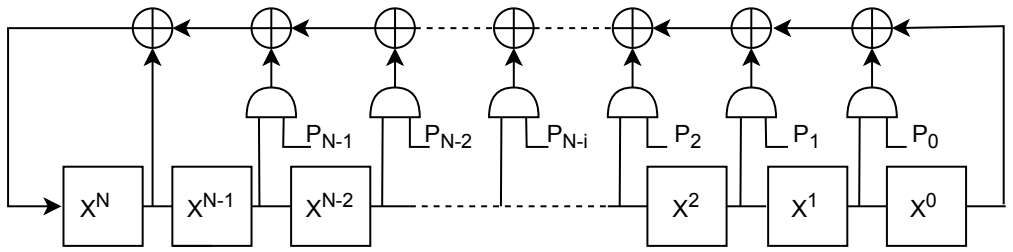


Fig. 3. Schematic of the proposed generic LFSR.

as shown in Figure 3. To achieve this, we define a configuration parameter P, computed as:

$$P = \text{Address} \oplus \text{Seed}. \tag{5}$$

The parameter P is employed to dynamically configure the internal feedback tap positions of the LFSR. However, unlike fully generic LFSR schemes, the proposed design fixes the first and last feedback positions to logical "1". This structural constraint enforces deterministic boundary behaviour while maintaining configurability in the intermediate tap positions. The feedback function of the proposed generic LFSR is defined as:

$$F(x) = x^0 \oplus \left(\bigoplus_{i=1}^{n-2} a_i \cdot x^i \right) \oplus x^{n-1}. \tag{6}$$

Here, x^n and x^0 (the first and last tap positions) are always active, i.e., $a_{n-1} = a_0 = 1$, while the coefficients $a_1, a_2, \ldots, a_{n-2} \in \{0, 1\}$ are determined dynamically using a subset of bits from the parameter P, as shown in Figure 3. This enables the flexible generation of LFSR structures without compromising structural stability. The dynamic nature of the intermediate feedback taps ensures that the generated keystream varies for each unique memory address and across different boot sessions.

For illustration, consider a 9-bit generic LFSR. If the memory address is `0x1A3` and the seed is `0x0F4`, the resulting P value becomes `000101010111`. Selecting the least significant 7-bits to configure internal taps yields the coefficient vector $[1, 0, 1, 0, 1, 1, 1]$, resulting in a feedback equation of the form:

$$F(x) = x_0 \oplus x_1 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8. \tag{7}$$

Similarly, with the same address `0x1A3` and a different seed `0x1F0`, $P =$ `000001010011`, and the internal taps become $[1, 0, 1, 0, 0, 1, 1]$, producing:

$$F(x) = x_0 \oplus x_1 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8. \tag{8}$$

In another scenario where both the address and seed are maximally different, such as `Address` = 0xFFF and `Seed` = 0x000, the resulting $P =$ `111111111111` yields a tap configuration $[1, 1, 1, 1, 1, 1, 1]$, and the corresponding feedback function becomes:

$$F(x) = x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8. \tag{9}$$

Finally, consider the degenerate case where the address and seed are identical, such that Address = Seed. In this case, $P =$ `000000000000`, and the dynamically selected internal coefficients become all zeros, i.e., $[0, 0, 0, 0, 0, 0, 0]$. Due to the boundary taps being fixed, the feedback function simplifies to:

$$F(x) = x_0 \oplus x_8. \tag{10}$$

In this minimal case, i.e., $P =$ `000000000000`, the generic LFSR remains functional, preserving the necessary non-linearity and variability at the fixed ends. These examples demonstrate the

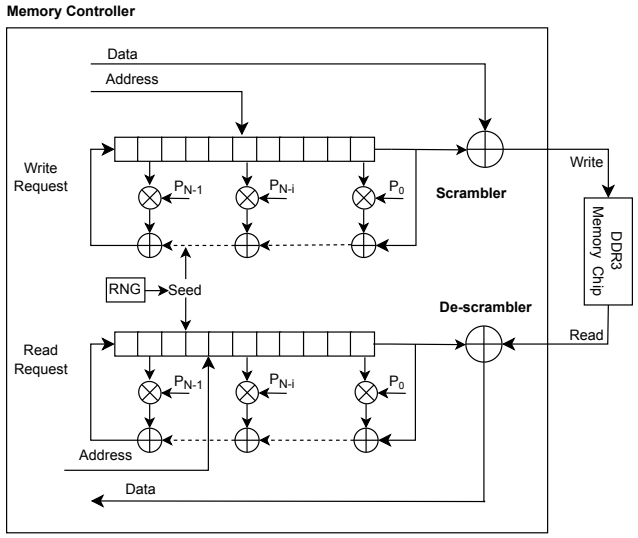


Fig. 4. Proposed architecture using generic LFSR.

strength of the proposed scheme in breaking predictable patterns by ensuring that the polynomial structure is dynamically varied across memory accesses and boot cycles, thereby strengthening resilience against cold boot attacks.

4.3 Proposed Memory Scrambling Architecture

The generic LFSR is integrated into the memory scrambling to replace the conventional LFSR-based scrambling technique used in DDR3 architectures, as shown in Figure 4. In the baseline design, data is XORed with a keystream generated by a fixed-polynomial LFSR, resulting in predictable whitening patterns that can be exploited through cold boot analysis, as explained in Section 3. However, the proposed architecture ensures secure memory access by dynamically adapting its structure for each memory transaction. This mechanism combines the structural stability of fixed feedback positions with the address-dependent and seed-dependent variability of dynamic taps to eliminate differential key periodicity across boots. This dynamic reconfiguration ensures that the generated keystreams exhibit high variability and unpredictability. Consequently, the traditional requirement of employing a primitive polynomial to guarantee maximum-length LFSR sequences becomes less critical. The core objective of using a primitive polynomial is to avoid short, repeating cycles in fixed LFSR configurations. However, in the proposed scheme, the LFSR parameters are dynamically selected, which inherently minimizes the probability of keystream repetition. The implications of using non-prime polynomials within this scheme are discussed in Section 6.3. The working of the proposed mechanism during write and read operations is explained below.

4.3.1 Write Mode Operation. During a write operation, the memory controller first obtains the full memory address and the boot-time seed. These two quantities are used to compute a configuration parameter P, as shown in Equation (5). This parameter dynamically determines the intermediate tap positions of the LFSR. This enables a distinct keystream to be generated for each memory address, even within the same boot session. Once the LFSR is fully configured, it generates a pseudo-random keystream K, which is then XORed with the plaintext data T to produce the

scrambled data T', computed as:

$$T' = T \oplus K. \tag{11}$$

The scrambled data T' is then stored in the corresponding memory location.

4.3.2 Read Mode Operation. In the read path, the same operations are performed in reverse to reconstruct the original data accurately. The memory controller retrieves the memory address and recomputes the configuration parameter P using the same seed from the current session. This ensures that the LFSR configuration is identical to what was used during the corresponding write operation. With this identical structure, the keystream K is regenerated using the same LFSR logic. The scrambled data T' fetched from memory is then XORed with the regenerated keystream to recover the original plaintext, as shown below:

$$\begin{aligned} T' &= T \oplus K \\ T' \oplus K &= T \oplus K \oplus K \\ T' \oplus K &= T. \end{aligned}$$

Since the LFSR configuration is dynamically derived from the seed and memory address, it guarantees that the same keystream is regenerated only under identical conditions, thus maintaining correctness while ensuring unpredictability across sessions. This mechanism effectively eliminates periodicity in the differential keystream (shown in the subsequent section), thereby mitigating the threat posed by cold boot attacks and rendering techniques like the stencil attack [2] infeasible.

5 Security Analysis of Proposed Scheme

This section presents a comprehensive security analysis of the proposed scrambling scheme, with particular emphasis on its effectiveness in preventing unauthorized access to external memory. The evaluation encompasses the scheme's resilience against a range of attack vectors, including brute-force key recovery attempts and memory disclosure attacks. Specifically, the analysis considers advanced attack scenarios such as the stencil attack, store and replay attack, cold boot attack, warm boot attack and probing attack, each of which exploits memory remanence or hardware-level leakage to compromise system confidentiality.

5.1 Stencil Attack Analysis

To formally assess the resilience of the proposed scrambling scheme against stencil attack, we consider an adversarial scenario where cold boot snapshots are extracted from two independent power cycles, denoted as session i and session j at time t. The attacker targets a specific memory address and computes the differential keystream ΔK_t as:

$$\Delta K_t = K_t^{(i)} \oplus K_t^{(j)}, \tag{12}$$

where $K_t^{(i)}$ and $K_t^{(j)}$ denote the keystreams generated at time step t in sessions i and j, respectively. In the proposed scheme, each keystream is produced using a generic LFSR with a session-specific configuration, characterized by a feedback function F_P derived from the XOR of the memory address and the boot-time seed:

$$F_P(x) = x^0 \oplus \left(\bigoplus_{k=1}^{n-2} a_k \cdot x^k \right) \oplus x^{n-1}. \tag{13}$$

Here, the coefficients $a_k \in \{0, 1\}$ for $1 \leq k \leq n-2$ are dynamically determined by a subset of bits from the configuration parameter $P = \text{Address} \oplus \text{Seed}$, while $a_0 = a_{n-1} = 1$ are fixed to ensure

structural stability. For two sessions i and j, the feedback functions will be:

$$F_{P_i}(x) = x^0 \oplus \left(\bigoplus_{k=1}^{n-2} a_k^{(i)} \cdot x^k \right) \oplus x^{n-1}, \tag{14}$$

$$F_{P_j}(x) = x^0 \oplus \left(\bigoplus_{k=1}^{n-2} a_k^{(j)} \cdot x^k \right) \oplus x^{n-1}. \tag{15}$$

By substituting the corresponding keystreams into Equation (12), the resulting expression yields the differential keystream as follows:

$$\Delta K_t = \bigoplus_{k=1}^{n-2} \left(a_k^{(i)} \oplus a_k^{(j)} \right) \cdot x_t^k, \tag{16}$$

where x_t^k denotes the LFSR state variable at time t for tap position k. Since $a_k^{(i)}$ and $a_k^{(j)}$ are derived from independent seeds, the XOR $c_k^{(i)} \oplus c_k^{(j)}$ yields a session-unique feedback structures. Consequently, ΔK_t exhibits pseudo-random behaviour without structural repetition. To detect any periodicity p in ΔK, the attacker performs an autocorrelation-based analysis. This involves segmenting the binary stream into chunks of length $|X| = |Y| = p$ and evaluating chunk similarity using the normalized Hamming distance, as shown in Equation (3). The segments satisfying equivalence are grouped into equivalence classes, as shown in Equation (4). If a dominant equivalence class emerges, the attacker can infer a candidate periodicity p, enabling subkey reconstruction. However, for the proposed scheme, due to the session-specific randomness in the LFSR configuration, we observe a normalized Hamming distance of approximately 0.5, which is far greater than ϵ:

$$\frac{H(X \oplus Y)}{|X|} \approx 0.5 \gg \epsilon. \tag{17}$$

This statistical independence prevents the formation of dominant equivalence classes and precludes any inference of a valid periodicity p. Thus, no exploitable structure emerges in ΔK, and the attacker cannot recover any deterministic subkey. To demonstrate the absence of repetition in the differential keystream, we revisit the same example presented in Section 3.2 for illustrating the stencil attack, now evaluated under the proposed defense, as shown in Figure 5. It can be observed from Figure 5 that the proposed scrambling scheme exhibits no observable repetition in the differential keystream. This contrasts with the fixed-pattern periodicity present in the DDR3 scrambling architecture shown in Figure 2.

5.2 Brute-force Attack Analysis

In this scenario, it is assumed that the adversary has no prior knowledge of the scrambling process. To successfully descramble the data transmitted across the memory interface bus, the attacker must perform an exhaustive brute-force search to determine the keystreams used in securing communication between the memory controller and the memory. In the proposed scrambling scheme, the keystream is dynamically generated based on both the memory address and a boot-time secret seed. This dynamic configuration ensures that the keystream varies across both memory addresses and boot sessions, thereby substantially increasing the complexity of any brute-force attack.

Let K denote the number of bytes written or read in a single memory transaction, and S be the length (in bytes) of the boot-time seed. An adversary attempting a brute-force attack must recover the correct keystream used during the scrambling process. We analyze the attack complexity in two scenarios:

Device 1	Device 2
Seed 1 = 110010	Seed 2 = 001011
Address 1 = 1000	Address 1 = 1000
Derived Seed = 111010	Derived Seed = 00011

$$K_0 = 011101\ 101110\ 010111\ 101011\ 010101\ 101011\ 110101\ 011010$$
$$K_1 = 000001\ 100000\ 110000\ 111000\ 111100\ 011110\ 001111\ 100111$$
$$K_0 \oplus K_1 = 011100\ 001110\ 100111\ 010011\ 101001\ 110101\ 111010\ 111101$$

Device 1	Device 2
Seed 1 = 110010	Seed 2 = 001011
Address 2 = 1010	Address 2 = 1010
Derived Seed = 111000	Derived Seed = 00001

$$K_0 = 011100\ 101110\ 110111\ 111011\ 111101\ 111110\ 011111\ 001111$$
$$K_1 = 100000\ 110000\ 111000\ 111100\ 111110\ 011111\ 001111\ 000111$$
$$K_0 \oplus K_1 = 111100\ 011110\ 001111\ 000111\ 000011\ 100001\ 010000\ 001000$$

Fig. 5. Demonstration of absence of differential key repetition using stencil attack [2].

5.2.1 Single-session Brute-force Complexity. In a fixed session, i.e., when the seed remains constant, the attacker aims to determine the keystream K used for a known memory address. Since the generic LFSR generates a pseudo-random keystream of K bytes per transaction, the adversary must perform an exhaustive trial over all possible keystreams of length K. Therefore, the total required number of brute-force attempts is:

$$\mathcal{C}_{\text{single}} = 2^{8K}, \tag{18}$$

where $8K$ represents the total number of keystream bits. This is equivalent to testing all possible pseudo-random patterns of the same length, assuming no structural knowledge of the LFSR configuration.

5.2.2 Cross-session Brute-force Complexity. In a more general and realistic scenario where the boot-time seed changes across sessions, the attacker cannot reuse the keystream or LFSR structure from a previous session. For each new session, the attacker must first guess the correct seed of length S, and for each seed candidate, repeat the 2^{8K} keystream trials. Thus, the total brute-force complexity across sessions becomes:

$$\mathcal{C}_{\text{cross}} = 2^{8S} \cdot 2^{8K} = 2^{8(S+K)}. \tag{19}$$

This exponential growth in complexity with respect to both the seed size and transaction width renders the attack computationally impractical for well-chosen parameters. The attack complexity for various values of S and K is shown in Table 1.

The analysis demonstrated in Table 1 proves that even under ideal conditions for the adversary (e.g., known plaintext and address), the search space quickly becomes infeasible to explore using brute-force. For instance, 8-byte seed and 32-byte key result in brute-force complexity of 2^{256} and 2^{320} for single-session and cross-session, respectively. Consequently, the proposed scrambling technique offers robust protection against brute-force key recovery attempts

5.3 Store and Replay Attack Analysis

In a store and replay attack, the adversary attempts to capture encrypted memory contents (i.e., scrambled data) during a legitimate execution session. Subsequently, the same ciphertext is re-injected, either within the same session or across different sessions. The attack is feasible if the

Table 1. Brute-force Attack Complexity for Varying Seed Length S and Data Size K

Seed Length (S) (bytes)	**Data Size (K) (bytes)**	**Complexity $\mathcal{C}_{\text{single}} = 2^{8K}$**	**Complexity $\mathcal{C}_{\text{cross}} = 2^{8(S+K)}$**
8	8	2^{64}	2^{128}
	16	2^{128}	2^{192}
	32	2^{256}	2^{320}
16	8	2^{64}	2^{192}
	16	2^{128}	2^{256}
	32	2^{256}	2^{384}
32	8	2^{64}	2^{320}
	16	2^{128}	2^{384}
	32	2^{256}	2^{512}

scrambling mechanism generates identical keystreams for the same memory address, enabling valid decryption upon replay. Traditional DDR3 scrambling architectures, which rely on fixed-polynomial LFSRs, are inherently susceptible to store and replay attacks due to their deterministic keystream generation. In such systems, a given memory address consistently produces an approximately identical scrambling sequence across sessions, thereby enabling an adversary to replay previously captured ciphertexts without detection or disruption of normal functionality.

In contrast, the proposed generic LFSR-based scrambling scheme integrates session-specific entropy and address dependency into keystream generation. The keystream K used for data scrambling is derived as:

$$K = \text{Generic\_LFSR}(\text{Address}, \text{Seed}), \tag{20}$$

where the configuration of the generic LFSR is dynamically determined using a bitwise XOR between the memory address and a boot-time secret seed. The configuration parameter P is used to select dynamic feedback taps of the LFSR, while the boundary taps are statically set to ensure structural stability. This design ensures that different sessions (i.e., different seeds) induce distinct keystreams even for the same memory address. Formally, consider a memory write operation (data T) to a specific address across two distinct sessions:

$$T_1 = T \oplus K_1, \quad \text{where } K_1 = \text{Generic\_LFSR}(\text{Address}, \text{Seed}_1), \tag{21}$$

$$T_2 = T \oplus K_2, \quad \text{where } K_2 = \text{Generic\_LFSR}(\text{Address}, \text{Seed}_2). \tag{22}$$

If an adversary replays T_1 in session 2, the memory controller attempts decryption using K_2, yielding:

$$T_{\text{replayed}} = T_1 \oplus K_2 = (T \oplus K_1) \oplus K_2 \neq T, \quad \text{since } K_1 \neq K_2. \tag{23}$$

The discrepancy in keystreams results in a corrupted plaintext T_{replayed}, thereby invalidating the attack. As the seed is randomly regenerated at each boot, the attacker cannot predict or replicate K across sessions without knowledge of the seed. Consequently, the proposed memory scrambling mechanism provides robust defense against the store and replay attack by tightly coupling the keystream generation with both memory address and session-specific seed. This dynamic structure ensures that scrambled data remains valid only within the session in which it was generated, rendering ciphertext reuse ineffective in subsequent sessions.

5.4 Traditional Cold Boot Attack Analysis

Traditional Cold boot attacks [15, 21, 47, 52, 53] exploit the data remanence property of DRAM to recover residual contents from memory after a system reboot. The traditional memory architectures,

where data is stored in plaintext or scrambled using predictable keystreams (e.g., fixed LFSRs), are inherently vulnerable to these cold boot attacks. In these attacks, an adversary may recover sensitive data by capturing the memory content immediately after power loss and analyzing static patterns. However, in the proposed generic LFSR-based scrambling scheme, data written to memory is pseudorandomly transformed using a dynamically generated keystream. This scrambling keystream is derived from a generic LFSR whose structure is configured using a session-specific seed and memory address. As a result, even if an adversary gains access to the raw contents of DRAM, the observed data will remain unintelligible. This is due to the requirement of knowing both the seed and the specific LFSR configuration used during that session. Moreover, since the seed is refreshed at each boot, the same memory address content results in entirely different ciphertexts across sessions. This behaviour effectively renders cold boot attacks infeasible under the proposed scheme.

5.5 Warm Boot Attack Analysis

Warm boot attack [18] exploits the persistence of DRAM data across system reboots that do not fully power down the memory. In such scenarios, the adversary forces a soft reboot of the system to bypass in-memory protections, intending to retrieve sensitive data before it is cleared or re-initialized. Unlike cold boot attacks, which involve complete power loss and often hardware intervention, warm boot attacks can be executed more stealthily and rapidly. This makes them a serious threat to systems with unprotected memory. In traditional DDR3 systems employing fixed-polynomial LFSR-based scrambling, the whitening process remains deterministic across reboots if the LFSR configuration and seed remain static. This allows an attacker to analyze and potentially reverse-engineer the whitening pattern using known plaintext structures or previously observed ciphertexts. The risk is heightened when the same memory address generates an identical keystream across reboots.

The proposed scheme, however, offers robust resistance against such attacks. The core security enhancement is the use of a boot-time random seed in conjunction with the memory address to dynamically configure the LFSR taps. The parameter P influences the intermediate tap positions of the LFSR, making the keystream K generated for each address both address-dependent and seed-dependent. As a result, even if an attacker successfully reboots the system and accesses residual data in memory, the retrieved content remains encrypted. This is because deciphering the ciphertext requires knowledge of the exact seed used in the preceding session, which is assumed to be unknown to the adversary. Since the seed is ephemeral and not stored in non-volatile memory, it becomes inaccessible post-reboot. Moreover, the non-repetitive keystream structure ensures that no meaningful patterns can be extracted through differential analysis or statistical methods. Consequently, even with physical access and rapid rebooting, an attacker is unable to retrieve or decrypt scrambled data residing in memory, thereby ensuring the confidentiality and integrity of sensitive information.

5.6 Probing Attack Analysis

The proposed generic LFSR-based scrambling scheme introduces an address-dependent and session-dependent transformation layer between the memory controller and DRAM memory. This effectively disrupts access-driven and communication-optimized cache attacks [23, 38, 50, 51], such as the enhanced Prime+Probe technique [38] targeting AES engines in SoCs. In traditional cryptographic hardware, cache-based attacks infer internal operations by exploiting correlations between memory accesses and secret-dependent control flows. However, in the proposed scheme, plaintext data is never written to or read from memory in unencrypted form. Instead, each memory transaction is scrambled using a pseudo-random keystream derived from a dynamically configured LFSR. The configuration of the generic LFSR depends on both the full memory address and a boot-time

Table 2. Area and Latency Overhead Analysis

Countermeasure	LUTs	Registers	Slice	Muxes	Latency (in clock cycles)
AES encryption [1]	395	1,448	167	0	10
Scrambling [30]	1,521	4,354	1,337	512	#read cyc.
Intel Scrambler [7]	243	256	104	0	0
Proposed	252	256	104	0	0

seed, ensuring that the whitening pattern changes across both spatial (address) and temporal (session) dimensions. From an adversary's perspective, the communication bus and cache hierarchy do not reveal meaningful patterns associated with the original plaintext or its encryption key. Even if a cache side-channel allows the attacker to observe memory access patterns or infer address-level activity, the contents transferred to or from DRAM remain obfuscated by a unique keystream. Therefore, the attacker cannot apply statistical techniques such as correlation analysis or autocorrelation to recover the key bytes.

Moreover, the proposed scheme eliminates keystream periodicity through its dynamic tap configuration, thus defeating differential attacks that rely on identifying repeating whitening sequences. Unlike traditional AES implementations targeted by Prime+Probe attacks, the scrambling process here is linear and stateless with respect to internal data dependencies. It also avoids memory-aligned substitution structures, such as **Look-Up Tables** (**LUTs**), which are commonly exploited in cache-based leakage attacks. Therefore, even advanced communication-aware cache attacks fail to provide leverage against the proposed memory scrambling scheme. The absence of static or data-dependent memory access patterns in the cryptographic path ensures that cache probing yields no exploitable correlation with the original or scrambled data. This analysis underscores the proposed scheme's robustness against both conventional and communication-enhanced implementation attacks.

6 Experimental Results

The proposed memory scrambling scheme was implemented in VHDL and synthesized using the AMD® Vivado design suite to evaluate its performance metrics and hardware resource utilization. For memory emulation, a simulation model of the Micron 64Mb SDRAM (MT48LC2M32B2) was employed to ensure compatibility and functional correctness. Although the current work focuses on modelling and simulation, we consider it both feasible and practical to implement the proposed generic LFSR-based scrambler on an FPGA and interface it with real memory (such as external SDRAM or on-chip SRAM). We plan to explore this as part of our future work.

6.1 Area and Latency Overhead Analysis

The hardware resource utilization of the proposed scheme was evaluated through a VHDL implementation on the VCU118 FPGA platform (device: xcvu9p-fpga21014-2L-e). The area overhead was quantified in terms of **Configurable Logic Blocks** (**CLBs**), including LUTs, Flip-Flops (registers), slices, and multiplexers. A comparative analysis was conducted against widely adopted security mechanisms such as AES encryption [1], Intel's DDR3 scrambling architecture [7], and data scrambling techniques based on XOR-logic and polynomial feedback [30]. As shown in Table 2, the data scrambling method proposed in [30] incurs the highest area overhead and a variable latency penalty that scales with the number of read cycles, rendering it impractical for real-time applications. In comparison, the AES-based solution imposes a substantial area and latency overhead, requiring 395 LUTs, 1,448 registers, and 167 slices, along with a latency penalty of 10 clock cycles. Since AMD's white paper [1] does not disclose the latency of its memory encryption engine, this latency estimate is based on the standard iterative AES implementation defined in NIST FIPS-197 [31]. However, this

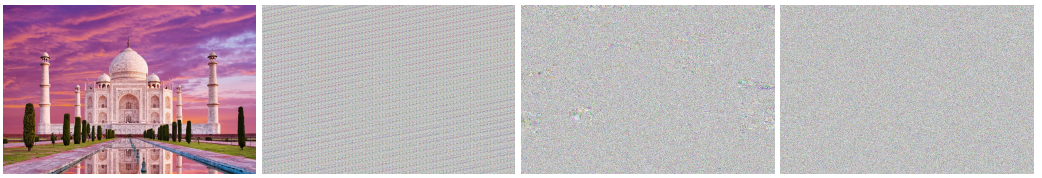
latency can be reduced to one clock cycle per block by utilizing a fully pipelined AES architecture, in which each of the ten AES rounds is implemented as a dedicated pipeline stage. During continuous memory operations, such as successive write (encryption) or read (decryption) operations, the pipeline remains fully utilized. This allows a new 128-bit data block to be processed every clock cycle, resulting in a sustained throughput of one encrypted or decrypted block per cycle. In contrast, for a single memory access, the pipeline must be completely filled before output is produced and subsequently flushed, leading to a fixed latency of 10 clock cycles corresponding to the pipeline depth. This performance improvement for continuous read and write operations comes at the cost of significantly increased resource consumption. The fully pipelined AES implementation replicates all ten AES rounds across the datapath, resulting in a tenfold increase in area utilization and greater critical path complexity. Additionally, the pipelined AES design exhibits considerably higher dynamic power consumption due to increased switching activity across the replicated S-boxes and MixColumns logic in each pipeline stage. Consequently, to mitigate performance degradation caused by this high latency, AMD's memory encryption architecture provides a selective bypass feature. This feature allows specific memory pages to remain unencrypted, thereby reducing access latency in performance-critical scenarios at the cost of reduced security for those pages.

In contrast, Intel's DDR3 scrambler achieves minimal hardware cost and zero latency overhead. However, it remains susceptible to stencil-based cold boot attacks due to its deterministic key regeneration process [2]. The proposed generic LFSR-based scrambling scheme addresses this security limitation while achieving a favourable trade-off between security and efficiency. Specifically, the proposed scheme requires 253 LUTs, 256 registers, and 104 slices. This represents only a marginal increase in area overhead compared to the Intel DDR3 scrambler. Although both the proposed scheme and Intel's DDR3 scrambler incur zero additional latency, their internal operations differ significantly. The proposed scheme employs simple XOR-based transformations for both scrambling and descrambling operations, similar to Intel's approach, thereby ensuring that the critical path remains unaffected. The key distinction lies in the dynamic computation of the feedback polynomial parameter P, as described in Section 4.2. Although the calculation of P introduces additional logic, it involves only lightweight bitwise operations. If this logic lies on the critical path, it may influence the timing constraints. However, this computation can be fully parallelized with the inherent read and write latencies of DRAM transactions [27, 28, 35]. In modern DRAM systems, both read and write operations are subject to fixed timing parameters, including **CAS Latency** (**CL**), **Additive Latency** (**AL**), and **Write CAS Latency** (**WCL**). These timing windows provide sufficient margin to accommodate the low-complexity polynomial generation logic without introducing additional delay. Consequently, the proposed scheme preserves the operational frequency compatibility with Intel's scrambler. At the same time, it offers substantially higher security by eliminating keystream periodicity and thwarting cold boot attacks.

6.2 Analysis of Effectiveness of the Proposed Scheme

To demonstrate the effectiveness of the proposed memory scrambling scheme, both qualitative and quantitative analyses have been performed. The qualitative evaluation includes visual inspection of encrypted images to assess perceptual randomness. On the other hand, quantitative analysis employs statistical metrics, such as correlation coefficients, mutual information, and divergence measures, to rigorously assess the statistical independence between original and scrambled data.

6.2.1 Qualitative Analysis. Figure 7 presents a comparative visual assessment of the scrambling efficacy of the proposed generic LFSR-based memory scrambling scheme. The original image, shown in Figure 7(a), serves as a reference for evaluating the degree of visual distortion [6]. Figure 7(b) illustrates the scrambled output using the conventional DDR3 scrambling technique [7], while



(a) Original (b) DDR3 Scrambler (c) Proposed (d) AES Encryption

Fig. 7. Images encrypted with various schemes.

Figure 7(c) depicts the result obtained through the proposed memory scrambling scheme. For reference, Figure 7(d) displays the encrypted output using the standard AES algorithm [1]. Notably, the proposed method achieves a level of perceptual randomness, that is, visually comparable to AES, effectively concealing the structural content of the original image. This qualitative evaluation aligns with the quantitative analysis presented in the subsequent subsection, thereby reinforcing the effectiveness of the proposed scheme in obfuscating image data.

6.2.2 Quantitative Analysis. To quantitatively assess the statistical independence between the original and encrypted images, multiple statistical measures are computed, as shown in Table 3. These metrics evaluate the presence of any residual structure, similarity, or dependency between the original image (7a) and scrambled images (7b, 7c, and 7d). It should be noted that these parameters were computed using Python libraries, including scipy.stats,[1] sklearn.metrics,[2] and scipy.spatial.distance.[3]

The values of the statistical parameters, as presented in Table 4, are either close to zero or statistically insignificant. The Pearson correlation coefficient of **0.0010** and Spearman's rank correlation of **0.00098** indicate an absence of linear and monotonic relationships between the original and encrypted images. Similarly, Kendall's tau value of **0.00066** confirms minimal ordinal association, supporting the claim of statistical independence. The Chi-Square test p-value **0.01253** of the proposed scheme rejects the null hypothesis that the input and output data are from the same distribution. The MI score of **0.0411** and its normalized counterpart (NMI = **0.0066**) further confirm that the ciphertext discloses very little about the plaintext. Additionally, the high MSE = **10868.85** indicates substantial pixel-level distortion introduced by the scrambling. Furthermore, the Jensen–Shannon divergence (JSD = **0.0093**) confirms a significant divergence in the statistical distributions of the two datasets. These results strongly indicate that the proposed memory scrambling scheme effectively disrupts inherent structures and correlations present in the original image. This disruption ensures a high degree of statistical independence between the plaintext and ciphertext domains.

6.3 Impact of Non-Primitive Polynomials on Proposed Scheme

To assess the impact of using non-primitive polynomials in the proposed scrambling scheme, an empirical analysis was conducted to measure the effective cycle lengths of randomly selected tap configurations. Although primitive polynomials yield maximal-length sequences of period 2^N-1, the proposed scheme does not enforce this constraint. Instead, security and functional correctness are achieved through architectural strategies, including per-access LFSR re-initialization, short-duration keystream generation, and address-dependent tap configuration. In the proposed scheme, the LFSR is re-initialized after every memory access, and the feedback polynomial is constructed using a hybrid approach. The highest-order term (x^N) and the lowest-order term (x) are always

[1] https://docs.scipy.org/doc/scipy/reference/stats.html
[2] https://scikit-learn.org/stable/modules/classes.html
[3] https://docs.scipy.org/doc/scipy/reference/spatial.distance.html

Table 3. Summary of Statistical Metrics Used for Evaluation

Metric	**Formula**	**Desired Value**
Pearson Correlation Coefficient (r_p) [33]	$r_p = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$	≈ 0
Spearman's Rank Correlation (ρ) [41]	$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}$	≈ 0
Kendall's Tau (τ) [19]	$\tau = \frac{C - D}{\frac{1}{2}n(n-1)}$	≈ 0
Chi-Square Test (χ^2) [4]	$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$	< 0.05
MI [39]	$I(X;Y) = \sum_x \sum_y p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$	≈ 0
Normalized Mutual Information (NMI) [43]	$\text{NMI}(X,Y) = \frac{2 \cdot I(X;Y)}{H(X) + H(Y)}$	≈ 0
Mean Squared Error (MSE) [29]	$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$	High
Jensen–Shannon Divergence (JSD) [24]	$\text{JSD}(P \parallel Q) = \frac{1}{2}D_{KL}(P \parallel M) + \frac{1}{2}D_{KL}(Q \parallel M)$, $M = \frac{1}{2}(P+Q)$	≈ 0

Table 4. Computed Statistical Parameters for Evaluating Encryption/ Scrambling Effectiveness

Metric	**AES encry. [1]**	**Scrambling [30]**	**Intel Scrambling [7]**	**Proposed**
Pearson Correlation Coefficient (r_p)	0.00041	−0.0044	0.0036	0.0010
Spearman's Rank Correlation (ρ)	0.00042	0.00414	0.0023	0.00098
Kendall's Tau (τ)	0.00031	−0.00272	0.00134	0.00066
Chi-Square Test p-value	0.00837	0.0 (Fail)	0.03457	0.01253
MI	0.010766	0.19439	0.0623	0.0411
NMI	0.002827	0.0351	0.0087	0.0066
MSE	1.9139×10^{76}	10802.19	11780.34	10868.85
JSD	0.0	0.0095	0.0093	0.0093

included as fixed tap points to ensure full-width feedback and prevent degenerate cycles. The remaining tap positions are deterministically derived through a bitwise XOR of the memory address and a session-dependent secret seed. Although the seed remains constant within a session to support consistent keystream regeneration during read and write operations, the memory address varies across accesses, producing a unique and deterministic feedback polynomial for each location. This address-dependent variation introduces spatial diversity into the scrambling process without requiring the strict use of primitive polynomials.

In the proposed scheme, the period of the LFSR is not fully utilized during scrambling operation. The length of the keystream required per access is limited by the size of the memory transaction. Specifically, a transfer of K bytes requires an $8 \times K$-bit keystream generated by running the LFSR for $8 \times K$ cycles. For example, a 64-byte cache line access requires only 512 bits of keystream. Due to the short and bounded execution window per access, the operation does not engage the full

Table 5. Average LFSR Cycle Length for 100 Randomly Selected Tap Configurations for Different LFSR Sizes

LFSR Size (bits)	Average Cycle Length	Maximum Cycle Length
16	12,178.85	65,535
20	269,006.76	1,048,575
24	3,818,782.20	16,777,215
28	42,995,910.89	268,435,455
32	575,089,930.20	2,147,483,647

period of the underlying polynomial, rendering its maximality irrelevant in practical scenarios. To quantify the effective cycle lengths for hybrid non-primitive configurations, 100 random tap vectors were generated for each LFSR size $N \in \{16, 20, 24, 28, 32\}$. For each configuration, the LFSR was initialized to a random non-zero state and executed until the internal state was repeated. The results are summarized in Table 5.

The empirical results show that for the proposed hybrid tap configurations, the average cycle lengths typically fall within the range of 2^{N-3} to 2^{N-2}, where N is the LFSR register size. This range is several orders of magnitude greater than the keystream lengths required per memory access. For example, with a 32-bit LFSR, the average cycle length exceeds 2^{29}, while only 512 bits are needed for a 64-byte memory transaction. This gap ensures that keystream reuse or observable periodicity remains practically infeasible under normal operating conditions. The source codes and tap configurations used to generate these results are publicly available at our GitHub repository[4].

7 Extension to DDR4 and DDR5 Memory Architectures

To evaluate the effectiveness of the proposed scrambling scheme against stencil attack, the experiments were conducted on a DDR3-based memory system. The motivation for initially targeting DDR3 stems from the fact that stencil attacks, the primary threat considered in this work, were originally demonstrated on DDR3 platforms [2]. To enable direct comparison and effective evaluation of the mitigation strategy, DDR3 was selected as the reference interface. However, the applicability of the proposed scrambling scheme is not limited to DDR3. The scrambling logic operates exclusively within the memory controller's data path, modifying data during write operations and restoring it during reads via a generic LFSR-based transformation. Since this transformation does not interfere with the DRAM's internal architecture, command protocol, or timing specifications, its integration into DDR4 or DDR5 requires no modification to the core memory interface or command scheduling logic. Both DDR4 and DDR5 preserve the burst-oriented memory transaction model introduced in DDR3, maintaining compatibility with the data access granularity assumed by the proposed scrambling scheme. Specifically, DDR3 and DDR4 utilize a fixed burst length of 8 (BL8) [28], meaning each read or write operation transfers eight consecutive data words per command. DDR5 advances this model by doubling the burst length to 16 (BL16) [35], thereby increasing throughput and improving data bus utilization. Despite the difference in burst size, the fundamental access pattern, deterministic, burst-aligned data transfers, remains consistent across all three standards and is fully compatible with the proposed scrambling scheme. Moreover, DDR4 and DDR5 memory controllers offer enhanced configurability and support for pipelined data operations, making them well-suited for integrating low-latency LFSR-based scramblers. These features facilitate seamless integration of the proposed scrambling scheme without violating timing constraints or impacting critical path requirements, enabling deployment across modern DRAM platforms.

[4] https://github.com/gkaggarwal/Generic_LFSR_Scheme

8 Conclusion

This work highlights the imperative need for robust protection mechanisms against memory disclosure attacks that target sensitive data in main memory systems. To this end, we introduced a novel generic LFSR-based secure scrambling architecture that establishes a lightweight yet effective communication framework between the memory controller and the memory device. The proposed scheme ensures both confidentiality and integrity of data traversing the memory bus. The experimental evaluations demonstrate that our approach offers security guarantees comparable to those of AES encryption and traditional scrambling methods while significantly reducing area overhead and eliminating access latency. Furthermore, the address- and seed-dependent LFSR structure effectively mitigates vulnerabilities associated with side-channel attacks, such as the stencil attack. The proposed scheme strikes a critical balance between hardware efficiency and security robustness, positioning it as a viable solution for next-generation memory protection. Future work will explore scalability in multi-user and multi-core environments, thereby broadening its applicability in secure computing systems.

References

[1] Advanced Micro Devices, Inc. 2021. *AMD Memory Encryption*. Technical Report. Advanced Micro Devices, Inc. Retrieved from https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/memory-encryption-white-paper.pdf

[2] Johannes Bauer, Michael Gruhn, and Felix C. Freiling. 2016. Lest we forget: Cold-boot attacks on scrambled DDR3 memory. *Digit. Investig.* 16, S (2016), S65–S74. DOI: 10.1016/j.diin.2016.01.009

[3] Siddhartha Chhabra and Yan Solihin. 2011. i-NVMM: A secure non-volatile main memory system with incremental encryption. In *Proceedings of the 2011 38th Annual International Symposium on Computer Architecture (ISCA)*. 177–188. DOI: 10.1145/2000064.2000086

[4] Frederick E. Croxton and Dudley J. Cowden. 1955. *Applied General Statistics* (3rd ed.). Prentice-Hall. Retrieved from https://archive.org/details/in.ernet.dli.2015.223541

[5] Daniel Dobkin, Nimrod Cever, and Itamar Levi. 2025. RAD-FS: Remote timing and power SCA security in DVFS-augmented ultra-low-power embedded systems. *ACM Trans. Embed. Comput. Syst.* 24, 2, Article 33 (Feb. 2025), 27 pages. DOI: 10.1145/3711836

[6] Elena-studio. n.d.. A mix of pollution, inadequate infrastructure and growing apathy is taking the shine off the Taj Mahal [Photograph]. Retrieved from https://skift.com/2022/06/14/taj-mahal-a-wonder-of-the-world-in-peril/. Getty Images.

[7] Maynard C. Falconer, Christopher P. Mozak, and Adam J. Norman. 2013. Suppressing Power Supply Noise Using Data Scrambling in Double Data Rate Memory Systems. Retrieved from https://patents.google.com/patent/US8503678B2/en

[8] Yu Fu, Jingqiang Lin, Dengguo Feng, Wei Wang, Mingyu Wang, and Wenjie Wang. 2023. RegKey: A register-based implementation of ECC signature algorithms against one-shot memory disclosure. *ACM Trans. Embed. Comput. Syst.* 22, 6, Article 97 (Nov. 2023), 22 pages. DOI: 10.1145/3604805

[9] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. 2018. Another flip in the wall of rowhammer defenses. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*. 245–261. DOI: 10.1109/SP.2018.00031

[10] Le Guan, Jingqiang Lin, Bo Luo, Jiwu Jing, and Jing Wang. 2015. Protecting private keys against memory disclosure attacks using hardware transactional memory. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*. 3–19. DOI: 10.1109/SP.2015.8

[11] Shay Gueron. 2016. A memory encryption engine suitable for general purpose processors. *IACR Cryptol. ePrint Arch.* 2016, 204 (2016), 1–14. Retrieved from https://eprint.iacr.org/2016/204

[12] Chandranshu Gupta, Gaurav Kumar, Manish Kumar, Satyadev Ahlawat, and Gaurav Varshney. 2025. *Poster Abstract: SRAM PUF-Based Logic Locking for Secure Authentication and IP Protection*. Association for Computing Machinery, New York, NY, USA, 660–661. DOI: https://doi.org/10.1145/3715014.3724057

[13] Kedar Gupta and Alastair Nisbet. 2016. Memory forensic data recovery utilising RAM cooling methods. In *Proceedings of the 14th Australian Digital Forensics Conference, Edith Cowan University, Perth, Australia*. 11–16. Retrieved from https://api.semanticscholar.org/CorpusID:65477698

[14] Saransh Gupta, Rosario Cammarota, and Tajana Šimunić. 2024. MemFHE: End-to-end computing with fully homomorphic encryption in memory. *ACM Trans. Embed. Comput. Syst.* 23, 2, Article 28 (March 2024), 23 pages. DOI: 10.1145/3569955

[15] 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 (May 2009), 91–98. DOI: 10.1145/1506409.1506429

[16] Mohamed Hassan, Anirudh M. Kaushik, and Hiren Patel. 2018. Exposing implementation details of embedded DRAM memory controllers through latency-based analysis. *ACM Trans. Embed. Comput. Syst.* 17, 5, Article 90 (Oct. 2018), 25 pages. DOI: 10.1145/3274281

[17] Kevin Hutto and Vincent Mooney. 2025. Implementing privacy homomorphism with random encoding and computation controlled by a remote secure server. *ACM Trans. Embed. Comput. Syst.* 24, 2, Article 27 (Jan. 2025), 27 pages. DOI: 10.1145/3651617

[18] Yichen Jiang, Shuo Wang, Renato Figueiredo, and Yier Jin. 2023. Warm-boot attack on modern DRAMs. In *Proceedings of the 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1–2. DOI: 10.23919/DATE56975.2023.10136888

[19] Maurice G. Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1-2 (1938), 81–93. DOI: https://doi.org/10.1093/biomet/30.1-2.81

[20] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *Proceedings of the 2014 ACM IEEE 41st International Symposium on Computer Architecture (ISCA)*. 361–372. DOI: 10.1109/ISCA.2014.6853210

[21] Gnanambikai Krishnakumar, Kommuru Alekhya REDDY, and Chester Rebeiro. 2019. ALEXIA: A processor with lightweight extensions for memory safety. *ACM Trans. Embed. Comput. Syst.* 18, 6, Article 122 (Nov. 2019), 27 pages. DOI: 10.1145/3362064

[22] Andrew Kwong, Daniel Genkin, Daniel Gruss, and Yuval Yarom. 2020. RAMBleed: Reading bits in memory without accessing them. In *Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP)*. 695–711. DOI: 10.1109/SP40000.2020.00020

[23] Dayeol Lee, Dongha Jung, Ian T. Fang, Chia che Tsai, and Raluca Ada Popa. 2020. An off-chip attack on hardware enclaves via the memory bus. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 487–504. Retrieved from https://www.usenix.org/conference/usenixsecurity20/presentation/lee-dayeol

[24] J. Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151. DOI: 10.1109/18.61115

[25] Stefanos Malliaros, Christoforos Ntantogian, and Christos Xenakis. 2016. Protecting sensitive information in the volatile memory from disclosure attacks. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*. 687–693. DOI: 10.1109/ARES.2016.75

[26] Brett Meadows, Nathan Edwards, and Sang-Yoon Chang. 2020. On-chip randomization for memory protection against hardware supply chain attacks to DRAM. In *Proceedings of the IEEE Security and Privacy Workshops (SPW)*. 171–180. DOI: 10.1109/SPW50608.2020.00044

[27] Micron Technology, Inc. 2010. *DDR3 SDRAM System-Power Calculator and Datasheet*. Retrieved from https://media-www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr3/4gb_ddr3_sdram.pdf. Micron DDR3 SDRAM Technical Documentation.

[28] Micron Technology, Inc. 2014. *DDR4 SDRAM*. Technical Report. Micron Technology, Inc. Retrieved from https://www.mouser.com/datasheet/2/671/Micron_05092023_8gb_ddr4_sdram-3175546.pdf

[29] Douglas C. Montgomery and George C. Runger. 2014. *Applied Statistics and Probability for Engineers* (6th ed.). Wiley. Retrieved from https://www.wiley.com/Applied+Statistics+and+Probability+for+Engineers%2C+7th+Edition-p-9781119400363

[30] Mădălin-Ioan Neagu and Liviu Miclea. 2014. Data scrambling in memories: A security measure. In *Proceedings of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics*. 1–6. DOI: 10.1109/AQTR.2014.6857847

[31] National Institute of Standards, Technology, Morris J. Dworkin, Meltem Sonmez Turan, and Nicky Mouha. 2023. *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication FIPS PUB 197-upd1. U.S. Department of Commerce. DOI: 10.6028/NIST.FIPS.197-upd1

[32] Hammond Pearce, Ramesh Karri, and Benjamin Tan. 2023. High-level approaches to hardware security: A tutorial. *ACM Trans. Embed. Comput. Syst.* 22, 3, Article 45 (April 2023), 40 pages. DOI: 10.1145/3577200

[33] Karl Pearson. 1895. Notes on regression and inheritance in the case of two parents. *Proc. R. Soc. Lond.* 58, 347-352 (1895), 240–242. DOI: https://doi.org/10.1098/rspl.1895.0041

[34] Rambus Inc. 2023. *IME-IP-340 Inline Memory Encryption Engine*. Retrieved from https://www.rambus.com/security/inline-memory-encryption/ime-ip-340/. Product Brief.

[35] Randall Rooney and Neal Koyle. 2019. *Micron® DDR5 SDRAM: New Features*. White Paper. Micron Technology, Inc. Retrieved from https://www.mouser.com/pdfDocs/ddr5_new_features_white_paper.pdf

[36] Jan-Peter Schat. 2021. Detection of a Cold Boot Memory Attack in a Data Processing System. Retrieved from https://eureka.patsnap.com/patent-US20210311823A1

[37] Hoseok Seol, Wongyu Shin, Jaemin Jang, Jungwhan Choi, Jinwoong Suh, and Lee-Sup Kim. 2017. In-DRAM data initialization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 11 (2017), 3251–3254. DOI: 10.1109/TVLSI.2017.2737646

[38] Johanna Sepúlveda, Mathieu Gross, Andreas Zankl, and Georg Sigl. 2021. Beyond cache attacks: Exploiting the bus-based communication structure for powerful on-chip microarchitectural attacks. *ACM Trans. Embed. Comput. Syst.* 20, 2, Article 17 (March 2021), 23 pages. DOI: 10.1145/3433653

[39] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x

[40] Sergei Skorobogatov. October, 2018. Hardware Security implications of Reliability, Remanence and Recovery in Embedded Memory. Journal of Hardware and Systems Security,Springer. DOI: 10.1007/s41635-018-0050-5

[41] Charles Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology* 15, 1 (1904), 72–101. DOI: https://doi.org/10.2307/1412159

[42] Ankush Srivastava and Prokash Ghosh. 2020. A novel approach of data content zeroization under memory attacks. *J. Electron. Test. Theory (JETTA), Springer* 36, 2 (2020), 147–167. DOI: 10.1007/s10836-020-05867-4

[43] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. 2000. Impact of similarity measures on web-page clustering. *Workshop on Artificial Intelligence for Web Search (AAAI)* (2000). Retrieved from https://www.ideal.ece.utexas.edu/papers/strehl_aaai00.pdf

[44] Salmin SultanaLi, ChenAbhishek BasakJason, and MartinJustin Gottschlich. 2017. Method and apparatus for preventing side channel attack on memory system and sensitive registers of SoC. In *Proceedings of the IP.com Disclosure Number: IPCOM000250953D*. Retrieved from https://patents.google.com/patent/US20190138719A1/en

[45] Soubhagya Sutar, Arnab Raha, Devadatta Kulkarni, Rajeev Shorey, Jeffrey Tew, and Vijay Raghunathan. 2017. D-PUF: An intrinsically reconfigurable DRAM PUF for device authentication and random number generation. *ACM Trans. Embed. Comput. Syst.* 17, 1, Article 17 (Dec. 2017), 31 pages. DOI: 10.1145/3105915

[46] Anna Trikalinou and Dan Lake. 2017. Taking DMA attacks to the next level: How to do arbitrary memory reads/writes in a live and unmodified system using a rogue memory controller. In *Proceedings of the Black Hat USA 2017*. Black Hat. Retrieved from https://www.blackhat.com/docs/us-17/wednesday/us-17-Trikalinou-Taking-DMA-Attacks-To-The-Next-Level-How-To-Do-Arbitrary-Memory-Reads-Writes-In-A-Live-And-Unmodified-System-Using-A-Rogue-Memory-Controller.pdf. Accessed: 2025-05-29.

[47] Yoo-Seung Won and Shivam Bhasin. 2021. Are cold boot attacks still feasible: A case study on raspberry pi with stacked memory. In *Proceedings of the 2021 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*. 56–60. DOI: 10.1109/FDTC53659.2021.00017

[48] Yuanzhe Wu, Grant Skipper, and Ang Cui. 2023. Cryo-mechanical RAM content extraction against modern embedded systems. In *Proceedings of the 2023 IEEE Security and Privacy Workshops (SPW)*. 273–284. DOI: 10.1109/SPW59333.2023.00030

[49] Mimi Xie, Shuangchen Li, Alvin Oliver Glova, Jingtong Hu, Yuangang Wang, and Yuan Xie. 2018. AIM: Fast and energy-efficient AES in-memory implementation for emerging non-volatile main memory. In *Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 625–628. DOI: 10.23919/DATE.2018.8342085

[50] Zhenyu Xu, Thomas Mauldin, Qing Yang, and Tao Wei. 2021. Runtime detection of probing/tampering on interconnecting buses. In *Proceedings of the 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 247–251. DOI: 10.1109/FCCM51124.2021.00038

[51] Zhenyu Xu, Thomas Mauldin, Zheyi Yao, Shuyi Pei, Tao Wei, and Qing Yang. 2020. A bus authentication and anti-probing architecture extending hardware trusted computing base off CPU chips and beyond. In *Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 749–761. DOI: 10.1109/ISCA45697.2020.00067

[52] 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 *Proceedings of the International Symposium on High Performance Computer Architecture*. 313–324. DOI: 10.1109/HPCA.2017.10

[53] Itamar Zimerman, Eliya Nachmani, and Lior Wolf. 2021. Recovering AES keys with a deep cold boot attack. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12955–12966. Retrieved from https://proceedings.mlr.press/v139/zimerman21a.html

Received 21 July 2025; revised 21 July 2025; accepted 29 July 2025