STORM — Small Table Oriented Redundancy-based SCA Mitigation for AES

Yaacov Belenky, Hennadii Chernyshchyk, Oleg Karavaev, Oleh Maksymenko, Valery Teper, Daria Ryzhkova, Itamar Levi, Osnat Keren and Yury Kreimer

FortifyIQ, Inc. | 12 Damonmill Square | Suite EB-1N | Concord, MA | 01742 USA
{belenky,chernyshchyk,karavaev,maksymenko,teper,ryzhkova,kreimer}@fortifyiq.com
{itamar.levi,osnat.keren}@biu.ac.il
https://www.fortifyiq.com/

Abstract. Side-channel-analysis (SCA) resistance with cost optimization in AES hardware implementations remains a significant challenge. While traditional masking-based schemes offer provable security, they often incur substantial resource overheads (latency, area, randomness, performance, power consumption). Alternatively, the RAMBAM scheme [BBA+22] introduced a redundancy-based approach to control the signal-to-noise ratio, and achieves exponential leakage reduction as redundancy increases. This method results in only a slight increase in area and in power consumption, and a significant decrease in the amount of randomness needed, without any increase in latency. However, it lacks a formal security proof.

In this study, we introduce a scheme, denoted STORM, that synergizes RAMBAM's methodology with the utilization of look-up-tables (LUTs) in memory (ROM/RAM) in a redundant domain. STORM, like RAMBAM, is as fast as a typical unprotected implementation and has the same latency, but has a significantly higher maximal clock frequency than RAMBAM, and consumes less than half the power. RAMBAM and STORM are code-based schemes in the sense that their set of representations is a code in the vector space $GF(2)^{8+d}$. RAMBAM requires a richer structure of a ring on $GF(2)^{8+d}$ and a ring homomorphism whereas STORM utilizes a simple vector space. In code-based-masking (CBM) [WMCS20], as in all masking schemes, non-interference based notions (t-S/NI) are fundamental for establishing provable security. RAMBAM and STORM diverge from this approach. While [WMCS20] (Section 6) employs codes in vector spaces over $GF(2^8)$ for AES protection, RAMBAM and STORM use codes over $GF(2)$ without the need for t-S/NI-gadgets, leaving them both smaller and more efficient.

Independence in security proofs typically means that in each individual computation (in a clock-cycle), at least one share does not participate. This approach does not work for RAMBAM where several field multiplications are executed sequentially in a cycle. However, in STORM no multiplications are performed due to its memory based tables, leaving only (independent) bitwise-XORs. Therefore, the reasoning necessary for proving security is different and STORM, unlike RAMBAM, enjoys provable security. We consider two distinct scenarios, *both with provable security*: (1) STORM1 — "leakage-free" memory reads, demonstrating (1,1,0)-robustness for LUTs with redundancy 2 in the 1-probe model and for LUTs with redundancy 6 in the 2-probe model, and (2) STORM2 — leaky memory reads, where additional protection mechanisms and a notion of memory-read robustness are introduced.

STORM can be implemented not only in HW, but in SW as well. However, this paper and the proofs in it relate to STORM's HW implementations.

Keywords: AES · DPA · Homomorphism · Leakage · Masking · Memory · Randomization · RAMBAM · Redundancy · Rings · Side-channel · SCA · STORM · SIFA-1 · LUT

1 Introduction

Side-channel analysis attacks (SCAs) [KJJ99] pose a significant challenge to the development of secure cryptographic algorithms in real-world, non-black-box scenarios without sacrificing performance or increasing electronic costs substantially. The cost of implementing protection against these attacks continues to be a major issue, making SCAs a persistent threat in the industry, particularly for resource-constrained devices. Consequently, efficient protection against SCAs remains a crucial area of academic research. Most recent protection schemes proposed in academia come with security proofs, ensuring resistance against some d^{th}-order side-channel leakage, contingent upon critical assumptions; namely, the independence of leakage across different sections or shares of an integrated circuit or sufficient noise. There is abundant literature on the challenges relating to signal 'glitches' [MMSS19], memory-recombination [MPG05, MPO05, CGP+12, BGG+14], and *compositional* issues. These can be found and corrected by verification tools, such as FullVerif [CGLS20, CS21], MaskVerif [BBC+19], and SILVER [KSM20]. At times some of these challenges can be simulated over circuit-simulation tools and solved before circuit tape-out, as demonstrated by the PROLEAD tool [MM22]. Moreover, in [DCEM18, LBS19, GGB+22] it was shown that such assumptions are approximations which hold in most cases, but there are cases in which (e.g.,) t-tests breach theoretical guarantees. Nevertheless, provable security provides a valuable guarantee of protection against SCA, albeit up to a certain number of traces. One common model of d-probe security was suggested in [ISW03]. However, this model does not account for certain physical effects, such as glitches, transitions, or couplings. An enhanced version of this model which takes these effects into account, dubbed (g, t, k)-robust d-probing security [FGDP+18] is used in our work because of its generality. Despite these advances, the challenge persists, as a result of the growing need to 'handle' all these hurdles in a provably-secure manner, which entails huge electronic costs and resource utilization.

The Rijndael cipher, which was adopted as the Advanced Encryption Standard (AES) by NIST in 2001, is one of the most widely employed block ciphers across hardware and software implementations. While AES exhibits no known cryptographic weaknesses, its naïve implementations are known to be highly prone to SCA. Therefore, protecting AES against SCA is an important example of the more general problem of protecting cryptographic algorithms against SCAs. Several provably-secure schemes of AES protection have been suggested, such as Threshold Implementations (TI) [NRR06], Domain Oriented Masking (DOM) [GMK16], Hardware-Private-Circuits (HPC) [CGLS20], Code-Based Masking (CBM) presented in [WMCS20], and LUT-based Masked Dual-Rail with Precharge Logic (LMDPL) [SBHM20].

However, provably-secure AES comes at a notable price, which counters NIST's design goals that date back to the 1997 AES competition. The selection of Rijndael as AES reflected these design goals, and resulted in fast and compact unprotected AES implementations. In TI, DOM, HPC and various other schemes, the proofs necessitate the execution of nonlinear transformations over several clock cycles to limit glitch propagation and compositional issues. These generally turn into either very high latency or alternatively high area or energy designs.

In 2022, a new AES protection called RAMBAM was introduced [BBA+22]. RAMBAM opts for a security parameter ("redundancy") that *controls the signal-to-noise ratio* rather than asserting provable security. *It reflects a significant paradigm shift.* This parameter has no impact on latency and performance, only slightly affects gate count, and allows for a significant reduction of randomness requirements. However, it causes an approximately exponential decrease in side-channel leakage as redundancy increases. This design explicitly does not adhere to provable security, typically achieved by restricting building blocks to be (strongly)-non-interfering (through various t-S/NI or PINI-gadgets), which entails high costs. The security parameter paradigm proves very practical in such a scenario, as it has in

the case of RSA. Although there is no proof, and there cannot be a proof, of the impossibility of factorization for large modulus sizes in RSA, there is a subexponential growth of the complexity of factorization, which reflects RSA's strength. Only the polynomial complexity of Shor's quantum factorization algorithm makes RSA insecure. RAMBAM proposes a similar approach: side-channel leakage decreases approximately exponentially with redundancy. Therefore, even in the face of powerful new attacks, a moderate increase in the redundancy (and, hence, a moderate increase in area) will uphold the scheme's security.

In this paper, we introduce a scheme, denoted STORM, that can be thought of as a combination of RAMBAM's methodology with the utilization of look-up-tables (LUTs) in memory (ROM or RAM) in a *redundant domain*. RAMBAM and STORM are code-based schemes in the sense of the definition of "code-based" in CBM, i.e. the set of the representations is a code in the vector space $GF(2)^{8+d}$. However, RAMBAM deviates from CBM in terms of its algebraic-requirements which necessitate both a richer structure of a ring on $GF(2)^{8+d}$ and a ring homomorphism. Moreover, both STORM and RAMBAM are significantly different from CBM — not in the general sense of being code-based, but with respect to the primitives utilized: in CBM, as in all masking schemes, the t-NI and t-SNI notions are crucial, since otherwise there cannot be provable security. CBM applied to AES [WMCS20, Section 6] uses codes in vector spaces over $GF(2^8)$, and the size of a masked representation is an integer number of bytes. In contrast, in RAMBAM and STORM the code is in a ring of dimension $8 + d$ over $GF(2)$, and the redundant-byte[1] consists of $8 + d$ bits, without using t-S/NI-gadgets. As a practical consequence of the underlying schemes, RAMBAM is much faster and requires substantially fewer random bits than code-based masking. The source of this efficiency stems from the fact that independence in security-proofs typically involves separation such that in each individual computation (in a clock-cycle), at least one share does not participate, or some form of randomness-*refreshing* takes place. However, this approach does not work for RAMBAM where various field multipliers can be executed sequentially in a clock cycle. This could be extremely counter-intuitive to practitioners. The security of RAMBAM is based not on probing but on a security parameter that affects the Signal-to-Noise Ratio (SNR). This approach results in an approximately exponential decrease in side-channel leakage as redundancy increases. Therefore, the security indication is not a binary True/False-leakage result in a d^{th}-order t-test, as is typical in probing security.

Two recent papers have discussed RAMBAM — [LMMS23] and [LK24]. In [LMMS23] the authors have analyzed RAMBAM and completed the picture by evaluating RAMBAM in a theoretical (probing security based) tool (PROLEAD) for the first time. They indeed found that RAMBAM is not provably-secure. Inherently, RAMBAM's construction provides asymptotic security, i.e., exponential reduction of leakage as the security parameter grows, with no probing security. In [LK24] the authors show that RAMBAM can be considerably expanded and generalized to a far broader range of encodings, can employ field isomorphisms, and can use simpler multiplication gadgets. This opens up new avenues and room for further research.

However, the main difference between RAMBAM and STORM is that STORM *indeed has* **provable security** stemming from the utilization of memory-based LUTs, which in turn implies that all non-linear operations are performed via memory reads, and no multiplications are done in logic, leaving only (independent) bitwise XORs. Therefore, proofs are possible, although the reasoning differs from that used for conventional NI-gadgets. STORM, unlike RAMBAM, provides provable security, as discussed below. Furthermore, STORM achieves the same latency as RAMBAM and is low-power compared

[1]Throughout the paper, in both RAMBAM and STORM, *redundant-byte* means a data chunk of $8 + d$ bits which represents a regular 8-bit byte in the *redundant domain*.

to RAMBAM. In the next sections, we consider two distinct scenarios, *both with provable security*:

1. **STORM1 — "leakage-free" memory reads:** demonstrating (1,1,0)-robustness for LUTs with redundancy 2 in the 1-probe model and for LUTs with redundancy 6 in the 2-probe robust model.
2. **STORM2 — leaky memory reads:** where additional protection mechanisms that preserve the scheme's efficiency in comparison to conventional paradigms, and a notion of memory-read robustness are introduced.

The theoretical security proofs are complemented by experimental data validating the effectiveness of the scheme.

STORM can be implemented not only in HW, but in SW as well. However, this paper and the proofs in it relate to the HW implementations.

1.1 Our Contributions

In this paper, we present STORM, a novel AES protection scheme which innovates in several ways: it combines a distinctive algebraic structure underpinning a code-based approach with a redundancy-based security-parameter embedded with tailored use of memory-based lookup tables (LUTs) similar to [GLZ12], but harnessing an algebraic-structure as discussed. The contributions provided by the scheme are:

- A new SCA-protection scheme for AES promoting a redundancy security-parameter that combines the redundant representations with LUT randomization in memory.
- Provably-secure schemes in the following two settings:
  1. **STORM1.** No leakage from memory reads exists (*leakage-free*), i.e., constant power consumption of memory reads or an otherwise protected memory is utilized.[2] For the digital logic in this variant of STORM, we prove (1,1,0)-robustness in the 1-probing model (starting from redundancy 2) and in the 2-probing model (starting from redundancy 6) as defined in [FGDP+18]. *To the best of our knowledge, this is the first validation of d-probe security for a scheme that is not share-based.*
  2. **STORM2.** Leaky memory reads. To protect against leakage from memory reads, we present two additional protections: LUT randomization and dummy reads. We introduce a notion of memory-read robustness and prove the existence of an instance of STORM2 with redundancy 6 which is both (1,1,0)-robust in the 2-probe model and memory-read robust. The number of random bits needed for LUT randomization is negligible (less than one bit per AES encryption). $8+d$ random bits per dummy read (= per S-box) are used to generate a random

[2] Side-channel attacks on SRAM memory exploit physical leakages to infer sensitive data from the memory. The prior art in side-channel secured SRAM design has focused on techniques such as embedded data masking, randomized compute-in-memory (CIM) SRAMs and logic encryption [SKM+23, AMZ+24, HJP+21, RDV12], power randomization and balancing [SNS24, GSNP24, HCKG19, GVL+18, CO19, GKF18, NMS23, ZWW16, WGC+21]. These methods primarily aim to protect the stored data by obfuscating the relationship between the actual data values and the observed side-channel leakage. Not all solutions may be effective in blocking address leakage from memory access patterns, which can reveal information about the addresses being accessed, or in preventing leakage which combines different address bits, different addresses or different data sections. However, this extremely active field is very new, and recent state-of-the-art research has only begun to address these limitations by exploring access-based countermeasures or expanding on previous solutions. These include randomized address mapping, dummy accesses, differential-privacy tools [LGD24], in addition to more traditional balancing and masking of the decoding logic. We believe that as part of the "leveled implementation" paradigm for SCA-security typically observed in NIST candidates for authenticated encryption schemes, such assumptions and primitives may have great value and reduce the efforts needed to SCA-protect other parts of the system. In its second setting (see below), STORM does not rely on leakage-free memory. However, if a leakage-free memory is available, it improves the parameters and most importantly, (1) no random bits are used beyond the initial randomization, (2) ROM can be used instead of RAM, (3) the amount of required memory decreases.

address, which is significantly lower than in most other protection schemes. There are two flavors of dummy reads, providing a tradeoff between performance (one round per clock cycle, as in unprotected implementations) and the compact LUTs.
- The providing of theoretical analysis, various examples and experimental analysis with measurement results.
- More versatility in primitive sets. Since the digital logic layer in STORM is very thin, the power consumption and the maximal clock frequency are determined mostly by memory reads. As a result, compared to alternative schemes, the power consumption is much lower, and the maximal frequency is much higher, at the cost of utilizing memory.
- STORM preserves inherent protection against statistical ineffective fault attacks SIFA-1 [DEK+18] from RAMBAM. At redundancy 4 (each byte is represented by 12 bits), STORM achieves protection against two simultaneous faults, and at redundancy 5 (each byte is represented by 13 bits), it achieves protection against three simultaneous faults. In contrast to share-based schemes, 3 shares (each byte is represented by 24 bits) are needed to protect against two simultaneous faults, and 4 shares (each byte is represented by 32 bits) are needed to protect against three faults.

In summary, STORM, like RAMBAM, is a non-share-based scheme with a security parameter (redundancy). It effectively solves the long-standing challenge of combining high security against SCA with high performance, low gate count, low latency, low power consumption, and low randomness utilization by offering a different tradeoff (memory utilization) that may be preferable to RAMBAM in many real-world cases. Unlike RAMBAM, for which it was experimentally shown in [BBA+22] that leakage rapidly decreases as redundancy grows, and the intuition behind it is explained, but without a security proof, we prove the lack of leakage from both digital logic and memory reads for STORM and show that the scheme is provably secure.

1.2 Organization of this Paper

The remainder of this paper is organized as follows:

- In Section 2, we provide an overview of previous algorithms, including plain AES, LUT-based AES, and RAMBAM. We also briefly introduce the d-probe security notions according to [ISW03] and [FGDP+18].
- Section 3 presents the STORM algorithm, including implementation notes. We establish the need of redundancy 4 or more for 2-probe security, even in the classic model from [ISW03]. Additionally, we introduce notation used throughout the paper and outline the requirements for the LUTs essential to the security proof. We also show how the level of protection against SIFA-1 depends on the redundancy d and compare it against RAMBAM and against share-based schemes.
- Section 4 presents the security proof of STORM1 (in the absence of side-channel leakage from memory reads), assuming that the LUTs comply with the requirements outlined in Section 3.
- In Section 5 we prove that redundancy 2 is enough for (1,1,0)-robustness of STORM1 in the 1-probe security model, and redundancy 6 is enough in the 2-probe security model, by building LUT suites that comply with the requirements from Section 3.
- In Section 6, we define a notion of memory-read robustness and present the additional protections and a security proof for STORM2 (leaky memory reads).
- Section 7 presents the experimental results in the STORM2 model.
- Section 8 promotes a discussion and draws conclusions.

2 Background

Basic notations: The following notations are used in this section:

- Algebraic structures: $F = GF(2)[x]/(P_0)$ denotes the finite field $GF(2^8)$. When clear from the context, we refer to F as the vector space Z_2^8. R denotes the vector space Z_2^{8+d}, and $\hat{F}$ denotes a subspace of R which is isomorphic to F and thus can be regarded as a finite field.
- Array indexing: indicated by a subscript. For example, c_j denotes byte j of the ciphertext.
- Expanded key indexing: k_j denotes byte (redundant-byte) j of the expanded key.
- $\|$ denotes the concatenation operator.
- $[\cdot]$ denotes the size of an array (in elements).
- $f(x)$ represents either a function operating on input x, or the result of a memory-based Look-Up Table (LUT) access in address x.

In Section 3 we introduce additional notations used thereafter.

2.1 Plain AES

The AES cipher is defined in [Nat01]. In this paper, we will use an equivalent definition for the standard AES presented below in Algorithm 1.

Remarks regarding Algorithm 1:

Algorithm 1: Plain AES encryption

Function $AesEnc(Nr, k[16(Nr+1)], p[16])$
  Input : $Nr \in \mathbb{N}$ — the number of rounds
    $k[16(Nr+1)] \in F^{16(Nr+1)}$ — the expanded key
    $p[16] \in F^{16}$ — the plaintext
  Output : $c[16] \in F^{16}$ — the ciphertext
  `/* First AddRoundKey */`
  $$\begin{pmatrix} x_0 \\ \vdots \\ x_{15} \end{pmatrix} = \begin{pmatrix} p_0 \\ \vdots \\ p_{15} \end{pmatrix} + \begin{pmatrix} k_0 \\ \vdots \\ k_{15} \end{pmatrix}$$
  `/* Regular rounds */`
  for $r = 1$ **to** $Nr - 1$ **do**
    $x = ShiftRows(x)$ `/* Permutation of bytes */`
    for $i = 0$ **to** 3 **do**
      $$\begin{pmatrix} x_{4i+0} \\ x_{4i+1} \\ x_{4i+2} \\ x_{4i+3} \end{pmatrix} = M \begin{pmatrix} A(x_{4i+0}^{254}) \\ A(x_{4i+1}^{254}) \\ A(x_{4i+2}^{254}) \\ A(x_{4i+3}^{254}) \end{pmatrix} + \begin{pmatrix} k_{16r+4i+0} \\ k_{16r+4i+1} \\ k_{16r+4i+2} \\ k_{16r+4i+3} \end{pmatrix}$$
    end for
  end for
  `/* Final round */`
  $x = ShiftRows(x)$
  $$\begin{pmatrix} c_0 \\ \vdots \\ c_{15} \end{pmatrix} = \begin{pmatrix} A(x_0^{254}) \\ \vdots \\ A(x_{15}^{254}) \end{pmatrix} + \begin{pmatrix} k_{16 \cdot Nr+0} \\ \vdots \\ k_{16 \cdot Nr+15} \end{pmatrix}$$
end

- We assume that the key has already been expanded, and therefore the only difference between AES128, AES192, and AES256 is in the number of rounds Nr and the size of the expanded key ($16(Nr+1)$ bytes).
- Following [Nat01], we see each byte as an element of the Galois field $GF(2^8)$ represented as a polynomial in $F = GF(2)[x]/(P_0)$, where $P_0 = x^8 + x^4 + x^3 + x + 1$.
- Byte additions and multiplications in Algorithm 1 below are the Galois field additions (XORs) and multiplications.
- We represent $Sbox(x)$ as $A(x^{254})$, where A is a specific affine transformation in the field seen as a vector space over $GF(2)$.
- We denote the `MixColumns` transformation matrix by: $M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$.

Note that the following set of byte functions is sufficient to implement the AES encryption:

- Field addition (XOR)
- Field multiplication (in the calculation of x^{254} and multiplications by constants in the matrix multiplication in line 8 of Algorithm 1)
- The affine transformation A

2.2 AES Protected with RAMBAM

The RAMBAM algorithm is described in [BBA+22] and is intended to protect AES implementations against SCA. Here, we sketch the general idea behind the RAMBAM algorithm:

- Define a *redundant domain* R larger than F and a homomorphism $H : R \to F$. We call the elements of the redundant domain *redundant-bytes*.
- Replace each byte of the plaintext and of the expanded key with its **randomly chosen** preimage under H.
- Perform the AES algorithm in the redundant domain R, with proper adjustments of the transformations used in it.
- After the last round, apply H to each redundant-byte of the result.

More concretely, since the transformations used in Algorithm 1 involve field addition and multiplication, the *redundant domain* must be a ring. Specifically, in RAMBAM the ring $R = GF(2)[x]/(PQ)$ is used, where P is an irreducible polynomial of degree 8 over $GF(2)$ (not necessarily P_0), and Q is a polynomial of degree d (called *redundancy*). The homomorphism $H : R \to F$ is defined as $H(x) = L^{-1}(x \bmod P)$, where $L : F \to R$ is an invertible linear transformation that maps the standard AES representation of any byte to the redundant representation of the same byte in the basis $\langle 1, t, \ldots, t^7 \rangle$ with d leading zeros prepended, where t is one of the roots of the polynomial P. Algorithm 2 below represents the RAMBAM algorithm, with the following remarks:

- P, Q, d, L are the parameters of the algorithm.
- Addition and multiplication are in the sense of R rather than in the sense of F.
- The algorithm $AesEnc^*$ is identical to $AesEnc$ (Algorithm 1), with the following changes:
  - It is executed in R instead of F.
  - Instead of $M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$, $M^* = \begin{pmatrix} L(2) & L(3) & 1 & 1 \\ 1 & L(2) & L(3) & 1 \\ 1 & 1 & L(2) & L(3) \\ L(3) & 1 & 1 & L(2) \end{pmatrix}$ is used.
  - Instead of A, A^* is used such that $(\forall x \in R)(H(A^*(x)) = A(H(x))$.

- $\pi \in R^{16}$ (line 2), $\kappa \in R^{16(Nr+1)}$ (line 3) and $\sigma \in R^{16}$ (line 4) are arrays of redundant bytes representing the plaintext, expanded key, and ciphertext, respectively.

The RAMBAM algorithm, as detailed in [BBA+22], includes additional intricacies, such as the re-randomization of intermediate results during exponentiation. However, for the scope of this paper, these details are not essential.

Algorithm 2: RAMBAM Protected AES Encryption

1 **Function** $ProtectedAesEnc(Nr, k[16(Nr+1)], p[16], r^k[16(Nr+1)], r^p[16])$
Input : $Nr \in \mathbb{N}$ — the number of rounds
$k[16(Nr+1)] \in F^{16(Nr+1)}$ — the expanded key
$p[16] \in F^{16}$ — the plaintext
$r^k[16(Nr+1)] \in (\mathbb{Z}_2^d)^{16(Nr+1)}$ — $16(Nr+1)$ d-bit random numbers for key randomization, seen as elements of $GF(2)[x]$
$r^p[16] \in (\mathbb{Z}_2^d)^{16}$ — 16 d-bit random numbers for data randomization, seen as elements of $GF(2)[x]$
Output : $c[16] \in F^{16}$ — the ciphertext

2 $$\begin{pmatrix} \pi_0 \\ \vdots \\ \pi_{15} \end{pmatrix} = \begin{pmatrix} L(p_0) \\ \vdots \\ L(p_{15}) \end{pmatrix} + \begin{pmatrix} r_0^p P \\ \vdots \\ r_{15}^p P \end{pmatrix}$$

3 $$\begin{pmatrix} \kappa_0 \\ \vdots \\ \kappa_{16\cdot Nr+15} \end{pmatrix} = \begin{pmatrix} L(k_0) \\ \vdots \\ L(k_{16\cdot Nr+15}) \end{pmatrix} + \begin{pmatrix} r_0^k P \\ \vdots \\ r_{16\cdot Nr+15}^k P \end{pmatrix}$$

4 $\sigma = AesEnc^*(Nr, \kappa, \pi)$

5 $$\begin{pmatrix} c_0 \\ \vdots \\ c_{15} \end{pmatrix} = \begin{pmatrix} H(\sigma_0) \\ \vdots \\ H(\sigma_{15}) \end{pmatrix}$$

6 **end**

2.3 LUT-based AES Encryption

Algorithm 3 is a LUT-based algorithm of AES encryption. Such algorithms were widely used for fast SW implementations of AES encryption before AES accelerating instructions (e.g., Intel's AES-NI) were introduced (see, for example, in [GLZ12]). This algorithm as such has nothing to do with protection against SCA. It only replaces the calculations related to `SubBytes` and `MixColumns` with accessing a memory-based LUT T with 256 entries. Each 16-bit entry $T[x]$ is $S(x)\|DS(x)$, where

$$S(x) = Sbox(x) \tag{1}$$

$$D(x) = 2 \cdot x \tag{2}$$

$$DS(x) = D \circ S(x) = 2 \cdot Sbox(x) \tag{3}$$

Besides LUT accesses, the only byte operations in this algorithm are XORs.

Note that although the matrix M used for `MixColumns` contains three different entries (1, 2 and 3), it is enough to keep in the LUT only $S(x)$ and $DS(x)$ because of the identity

$$3 \cdot Sbox(x) = (1+2)Sbox(x) = S(x) + DS(x) \tag{4}$$

Algorithm 3 below represents the LUT-based AES encryption algorithm, with the following remarks:

- Unlike Algorithms 1 and 2, here all four AES transformations, including `ShiftRows`, are merged into a one per-round operation.
- The addition of the indices of array x is modulo 16.

Algorithm 3: LUT-based AES Encryption

Function $AesEncLUT(Nr, k[16(Nr+1)], p[16])$
Input : $Nr \in \mathbb{N}$ — the number of rounds
$k[16(Nr+1)] \in F^{16(Nr+1)}$ — the expanded key
$p[16] \in F^{16}$ — the plaintext
Output : $c[16] \in F^{16}$ — the ciphertext
`/* First AddRoundKey */`

$$\begin{pmatrix} x_0 \\ \vdots \\ x_{15} \end{pmatrix} = \begin{pmatrix} p_0 \\ \vdots \\ p_{15} \end{pmatrix} + \begin{pmatrix} k_0 \\ \vdots \\ k_{15} \end{pmatrix}$$

`/* Regular rounds */`
for $r = 1$ **to** $Nr - 1$ **do**
for $i = 0$ **to** 3 **do**

$$\begin{pmatrix} x_{4i+0} \\ x_{4i+1} \\ x_{4i+2} \\ x_{4i+3} \end{pmatrix} = \begin{pmatrix} DS(x_{4i}) + DS(x_{4i+5}) + S(x_{4i+5}) + S(x_{4i+10}) + S(x_{4i+15}) + k_{16r+4i} \\ DS(x_{4i+5}) + DS(x_{4i+10}) + S(x_{4i+10}) + S(x_{4i+15}) + S(x_{4i}) + k_{16r+4i+1} \\ DS(x_{4i+10}) + DS(x_{4i+15}) + S(x_{4i+15}) + S(x_{4i}) + S(x_{4i+5}) + k_{16r+4i+2} \\ DS(x_{4i+15}) + DS(x_{4i}) + S(x_{4i}) + S(x_{4i+5}) + S(x_{4i+10}) + k_{16r+4i+3} \end{pmatrix}$$

end for
end for
`/* Final round */`

$$\begin{pmatrix} c_0 \\ \vdots \\ c_{15} \end{pmatrix} = \begin{pmatrix} S(x_{5\cdot 0}) + k_{16\cdot Nr+0} \\ \vdots \\ S(x_{5\cdot 15}) + k_{16\cdot Nr+15} \end{pmatrix}$$

end

2.4 Probing Models

In this paper, we will prove the security of STORM protection using the robust n-probing model introduced in [FGDP+18]. In this section, we describe the classic n-probing model introduced in [ISW03] and the robust n-probing model.

Although it is customary to denote the order of the probing model as d, in this paper we use n instead, since d is reserved for redundancy.

2.4.1 The Original ("Classic") Probing Model

According to the classic n-probing model introduced in [ISW03], a circuit is called n-secure if every n-tuple of its intermediate variables is independent of any sensitive variable. Here "intermediate variables" mean stable values of some wires, each at one clock cycle. This model does not take into account transitions and glitches.

2.4.2 The Robust n-probing Model

The robust n-probing model introduced in [FGDP+18] suggests extensions of the classic n-probing model, in order to take into account glitches, transitions, and couplings, by

replacing probes with extended probes.

For the transition extension, each probe on a wire W at the clock cycle t is extended by implicitly adding another probe on the same wire W at the clock cycle $t+1$.

For the glitch extension, each probe on a wire W at the clock cycle t in a combinational circuit is extended by implicitly adding probes at the same clock cycle t on all wires that affect, directly or indirectly, the value at W.

In this paper, we do not deal with couplings and do not describe the coupling extension.

A combinational circuit is called secure in the $(g,t,0)$-robust n-probing model if for any set of n probes, after the transition extension if $t=1$ and after the glitch extension if $g=1$, the set of values on the extended set of probes is independent of any sensitive variable. The $(0,0,0)$-robust n-probing model is therefore the classic n-probing model.

3 STORM Description

The STORM algorithm combines the redundant domain approach of RAMBAM with the use of LUTs. The primary goal of using LUTs in STORM is the replacement of all non-linear operations (which are the key sources of side-channel leakage) with memory accesses. At the same time, since the only remaining operations are memory accesses and XORs, there is an additional advantage of low power consumption and a high maximal clock frequency compared to other protection schemes.

In the following subsections, we describe the algorithm, prove a necessary condition for its security against classic probing attacks, formulate requirements to the LUT that must hold for the security proofs to work, and discuss STORM's immunity against statistical ineffective fault attacks.

Starting from this section, we make use of the following notations (all variables below are redundant-bytes, i.e., elements of R):

- p_j — redundant-byte j of the plaintext (after the initial randomization).
- ${}^{r}k_j$ — redundant-byte j of the round key of round r (the first round key used in the initial `AddRoundKey` has $r=0$).
- ${}^{r}x_j$ — redundant-byte j of the internal state before round r.
- c_j — redundant-byte j of the ciphertext (before the final de-randomization).

For bit indices in redundant-bytes we will use right upper indices, e.g., ${}^{r}x_j^i$ is bit i of redundant-byte j of the internal state before round r.

3.1 The Algorithm

A LUT-based implementation eliminates the need for multiplication over a finite field. That is, the redundant domain does not need to support multiplication, and the vector space $R=\mathbb{Z}_2^{8+d}$, rather than a ring, can be used as the redundant domain. Its elements are represented by $(8+d)$-bit redundant-bytes.

We denote by $\hat{F}$ the set of all $x\in R$ such that the d leading bits of x are zeros, and denote by $J:F\to\hat{F}$ the natural isomorphism between F and $\hat{F}$: $J(s)=0^d\|s$ for any $s\in F$. Due to this isomorphism, we will regard $\hat{F}$ as another representation of $GF(2^8)$. For example, the multiplication of two elements in $x_1,x_2\in\hat{F}$ is conducted as $x_1\cdot x_2=J(J^{-1}(x_1)\cdot J^{-1}(x_2))$.

A linear map $H:R\to\hat{F}$ is defined. Unlike the original RAMBAM scheme, where there were reasons for using $P\neq P_0$, here we require H to be a projection (i.e., $H^2=H$). It is easy to see that the function $G(x)=x-H(x)$ is a projection of R onto $Ker(H)$, and we can see R as a direct sum of its subspaces $\hat{F}$ and $Ker(H)$. Each $x\in R$ can be uniquely represented as $x=H(x)+G(x)$, where $H(x)\in\hat{F}$ is the *clear component* of x,

and $G(x) \in Ker(H)$ is its *random component*. A XOR with a randomly chosen element of $Ker(H)$ is used for the randomization of the input data and of the key.

If two elements $x_1, x_2 \in Ker(H)$ have the same d most significant bits, they must be equal, since $x_1 + x_2 \in Ker(H) \cap \hat{F} = \{0\}$. Therefore the most significant d bits of the 2^d elements of $Ker(H)$ assume all 2^d possible values. We sort all elements of $Ker(H)$ in lexicographic order and then remove the d most significant bits from each element. We denote the resulting array by K. K consists of 2^d 8-bit records. It is easy to see that the image of the function $V : \mathbb{Z}_2^d \to R$, defined as $V(s) = s \| K[s]$, is $Ker(H)$. We use this array, typically implemented in gates rather than in the memory, to randomly choose an element of $Ker(H)$.

Algorithm 4 shown below implements STORM, with the following remarks:

- The redundancy d, the projection H, and the array K used to generate elements of $Ker(H)$ as described above are the parameters of the algorithm.
- Addition is in the sense of R rather than of F.
- The algorithm $AesEncLUT^*$ is identical to $AesEncLUT$ (Algorithm 3), with the following changes:
  - It is executed in R rather than in F.
  - Instead of functions $S : F \to F$ and $DS : F \to F$ and table T, functions $S^* : R \to R$ and $DS^* : R \to R$ and table T^* are used, where the table contains the values of $S^*(x)$ and of $DS^*(x)$. The requirements for these functions are listed in Section 3.3.
- $\pi \in R^{16}$ (line 2), $\kappa \in R^{16(Nr+1)}$ (line 3) and $\sigma \in R^{16}$ (line 4) are arrays of redundant bytes representing the plaintext, expanded key, and ciphertext, respectively.

Note that, unlike the original RAMBAM algorithm, there is no re-randomization during the execution.

Algorithm 4: STORM — LUT-based Protected AES Encryption

1 **Function** $ProtectedAesEncLUT(Nr, k[16(Nr+1)], p[16], r^k[16(Nr+1)], r^p[16])$
Input : $Nr \in \mathbb{N}$ — the number of rounds
$k[16(Nr+1)] \in R^{16(Nr+1)}$ — the expanded key
$p[16] \in R^{16}$ — the plaintext
$r^k[16(Nr+1)] \in (\mathbb{Z}_2^d)^{16(Nr+1)}$ — $16(Nr+1)$ d-bit random numbers for key randomization
$r^p[16] \in (\mathbb{Z}_2^d)^{16}$ — 16 d-bit random numbers for data randomization
Output : $c[16]$ — the ciphertext

2 $$\begin{pmatrix} \pi_0 \\ \vdots \\ \pi_{15} \end{pmatrix} = \begin{pmatrix} J(p_0) \\ \vdots \\ J(p_{15}) \end{pmatrix} + \begin{pmatrix} (r_0^p \| K[r_0^p]) \\ \vdots \\ (r_{15}^p \| K[r_{15}^p]) \end{pmatrix}$$

3 $$\begin{pmatrix} \kappa_0 \\ \vdots \\ \kappa_{16 \cdot Nr+15} \end{pmatrix} = \begin{pmatrix} J(k_0) \\ \vdots \\ J(k_{16 \cdot Nr+15}) \end{pmatrix} + \begin{pmatrix} (r_0^k \| K[r_0^k]) \\ \vdots \\ (r_{16 \cdot Nr+15}^k \| K[r_{16 \cdot Nr+15}^k]) \end{pmatrix}$$

4 $$\begin{pmatrix} \sigma_0 \\ \vdots \\ \sigma_{15} \end{pmatrix} = AesEncLUT^*(Nr, \kappa, \pi)$$

5 $$\begin{pmatrix} c_0 \\ \vdots \\ c_{15} \end{pmatrix} = \begin{pmatrix} J^{-1} \circ H(\sigma_0) \\ \vdots \\ J^{-1} \circ H(\sigma_{15}) \end{pmatrix}$$

6 **end**

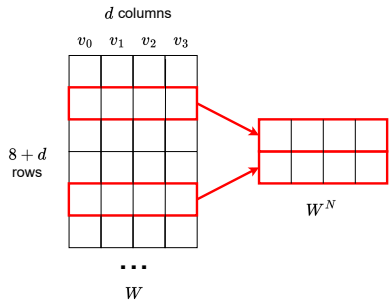


Figure 1

3.2 A Necessary Condition on Linear Map H for its Security Against Classic Probing Attacks

In Section 4 we will prove, under certain assumptions, the security of the scheme against SCA in (1,1,0)-robust 1-probing and 2-probing models introduced in [FGDP+18]. As an initial step toward this objective, we examine the necessary (but not necessarily sufficient) conditions for the linear map H to ensure the security within the classic n-probing model, as introduced in [ISW03].

Definition 1. If $N = \langle i_0, \ldots, i_{n-1} \rangle$ is an n-tuple of bit positions, then P^N is the $n \times (8+d)$ matrix in which only n bits P^N_{t,i_t} $(0 \le t < n,\ 0 \le i_t < 8+d)$ are 1, and all other bits are 0.

It is easy to see that P^N represents the linear transformation that maps a vector $x \in R$ to a vector in $\mathbb{Z}_2^n$ consisting of only the bits of x that are in the bit positions from N.

Definition 2. If $F_0 : \mathbb{Z}_2^t \to \mathbb{Z}_2^{u_0}$ and $F_1 : \mathbb{Z}_2^t \to \mathbb{Z}_2^{u_1}$ (where t, u_0, u_1 are arbitrary natural numbers) are linear maps, then the *direct sum* $F_0 \oplus F_1 : \mathbb{Z}_2^t \to \mathbb{Z}_2^{u_0+u_1}$ is defined as $F_0 \oplus F_1(x) = \langle F_0(x), F_1(x) \rangle$, where we represent any vector from $\mathbb{Z}_2^{u_0+u_1}$ as a pair consisting of a vector from $\mathbb{Z}_2^{u_0}$ and a vector from $\mathbb{Z}_2^{u_1}$.

Let $\{v_0, \ldots, v_{d-1}\}$ be an arbitrary basis of $Ker(H)$, and let W be the $(8+d) \times d$ matrix columns of which are the vectors of the basis. For an arbitrary n-tuple N of bit positions $(n \le d)$, let W^N be the $n \times d$ matrix formed by the rows of W corresponding to the bit positions in N.

Example 1. Figure 1 illustrates matrices W and W^N for $d = 4$ and $|N| = 2$.

Definition 3. A linear map $H : R \to F$ is called n-uniform if for any n-tuple N of bit indices $P^N(Ker(H)) = \mathbb{Z}_2^n$.

Lemma 1. *Let N be an n-tuple of bit positions where $n \le d$. Then H is n-uniform if and only if $rank(W^N) = n$.*

Proof. It is easy to see that the columns of W^N are the vectors $P^N(v_0), \ldots, P^N(v_{d-1})$, and the dimension of the space spanned by these vectors (which is $P^N(Ker(H))$) equals $rank(W^N)$. □

Lemma 2. *n-uniformity of H is a necessary condition for n-probing security of algorithm $AesEncLUT^*$ in the classic model.*

Proof. If H is not n-uniform, then for some n-tuple $N = \langle i_0, \ldots, i_{n-1} \rangle$ of bit indices $P^N(Ker(H)) \ne \mathbb{Z}_2^n$. Let $X = \langle x_0, \ldots, x_{n-1} \rangle$ be any vector from $\mathbb{Z}_2^n \setminus P^N(Ker(H))$. If the probes are set at the inputs to bits $i_0, \ldots, i_{n-1}$ of any redundant-byte x of the state, and

the vector of values measured on their probes is X, then the clear component $H(x)$ of the redundant-byte cannot be 0, so the measurements provide information regarding a byte of the state. □

The following two lemmas are rather trivial, but are convenient to refer to.

Lemma 3. *H is 1-uniform if and only if there are no all-zero rows in matrix W.*

Proof. If $|N| = 1$, then W^N is one row of W. Clearly, its rank is less than 1 if and only if it is all-zeros. □

Lemma 4. *H is 2-uniform if and only if there are no all-zero rows and no matching rows in matrix W.*

Proof. If $|N| = 2$, then W^N is a matrix with two rows out of W. It has rank less than 2 if and only if at least one of the rows is all-zeros, or the rows are identical — otherwise, the rows are linearly independent, and the rank is 2. □

Theorem 1. *If $d < 4$, then the algorithm $AesEncLUT^*$ using H cannot be robust in the classic 2-probing model.*

Proof. The matrix W has $8+d$ rows, d bits in each. There are 2^d possible bit combinations in each row. If $d < 4$, then $2^d < 8 + d$, and there exists a pair of identical rows, therefore by Lemma 4 H is not 2-uniform, and by Lemma 2 the algorithm $AesEncLUT^*$ using H cannot be robust in the classic 2-probing model. □

3.3 LUT functions: Definitions, Requirements for Entropy Preservation and Uniformity

Two functions of $AesEncLUT^*$ are implemented by LUTs, S^* and DS^*. The requirements for these functions are classified into two categories:

- Requirements that guarantee correctness and entropy preservation
- Requirements that guarantee certain uniformity properties

These groups of requirements, along with some definitions, lemmas and notes, are listed in the following subsections.

3.3.1 Requirements Related to Correctness and Entropy Preservation

Requirement 1. *$S^*(x)$ can be represented as*

$$S^*(x) = Sbox(H(x)) + \Delta_S(H(x)) + \Lambda_S(G(x)) \tag{5}$$

where $\Delta_S : \hat{F} \to Ker(H)$ is an arbitrary function and $\Lambda_S : Ker(H) \to Ker(H)$ is an invertible linear function.

Note that the clear component of $S^*(x)$ is

$$H(S^*(x)) = Sbox(H(x)) \in \hat{F} \tag{6}$$

and its random component is

$$G(S^*(x)) = \Delta_S(H(x)) + \Lambda_S(G(x)) \in Ker(H) \tag{7}$$

Requirement 2. *DS^* can be represented as $D^* \circ S^*$, where*

$$D^*(x) = 2 \cdot H(x) + \Delta_D(H(x)) + \Lambda_D(G(x)) \tag{8}$$

where $\Delta_D : F \to Ker(H)$ is an arbitrary function and $\Lambda_D : Ker(H) \to Ker(H)$ is an invertible linear function.

Denoting

$$\Delta_{DS} = \Delta_D \circ Sbox + \Lambda_D \circ \Delta_S \tag{9}$$

$$\Lambda_{DS} = \Lambda_D \circ \Lambda_S \tag{10}$$

we have

$$\begin{aligned} DS^*(x) &= 2 \cdot Sbox(H(x)) + \Delta_D(Sbox(H(x))) + \Lambda_D(\Delta_S(H(x))) + \Lambda_D(\Lambda_S(G(x))) \\ &= 2 \cdot Sbox(H(x)) + \Delta_{DS}(H(x)) + \Lambda_{DS}(G(x)) \end{aligned} \tag{11}$$

and therefore the clear component of $DS^*(x)$ is

$$H(DS^*(x)) = 2 \cdot Sbox(H(x)) \in \hat{F} \tag{12}$$

and its random component is

$$G(DS^*(x)) = \Delta_{DS}(H(x)) + \Lambda_{DS}(G(x)) \in Ker(H) \tag{13}$$

Note that in each coset of H the first two addends in equations (5), (8) and (11) are constant, so the functions S^*, D^*, DS^* are affine on any coset of H, where the linear part is the same for all cosets, and only the free term varies.

Lemma 5. *If Requirements 1 and 2 hold, then AesEncLUT* correctly performs AES encryption.*

Proof. Taking into account (6) and (12), it is easy to prove by induction on the rounds that the clear part of the redundant-bytes of the state in $AesEncLUT^*$ is identical up to the isomorphism J to the corresponding bytes in $AesEncLUT$. □

Lemma 6. *Assuming that Requirements 1 and 2 hold, for any fixed values of the expanded key ${}^r k_j$ and of the clear components of the plaintext $H(p_j)$ in algorithm $AesEncLUT^*$, the correspondence between the values of the plaintext p_j and of the state at any round r ${}^r x_j$ is one-to-one (where $0 \le r \le Nr,\ 0 \le j < 16$).*

Proof. The transformation in the inner loop of the regular rounds limited to the random component of the redundant-bytes can be represented as

$$\begin{pmatrix} G({}^{r+1}x_{4i}) \\ G({}^{r+1}x_{4i+1}) \\ G({}^{r+1}x_{4i+2}) \\ G({}^{r+1}x_{4i+3}) \end{pmatrix} = Z \begin{pmatrix} \Lambda_S(G({}^r x_{4i})) \\ \Lambda_S(G({}^r x_{4i+5})) \\ \Lambda_S(G({}^r x_{4i+10})) \\ \Lambda_S(G({}^r x_{4i+15})) \end{pmatrix} + \begin{pmatrix} G({}^r k_{4i}) \\ G({}^r k_{4i+1}) \\ G({}^r k_{4i+2}) \\ G({}^r k_{4i+3}) \end{pmatrix} + {}^r C_i \tag{14}$$

where

$$Z = \begin{pmatrix} \Lambda_D & \Lambda_D + I & I & I \\ I & \Lambda_D & \Lambda_D + I & I \\ I & I & \Lambda_D & \Lambda_D + I \\ \Lambda_D + I & I & I & \Lambda_D \end{pmatrix} \tag{15}$$

I is the $(8+d) \times (8+d)$ identity matrix, and ${}^r C_i \in Ker(H)$ is a constant vector depending on the clear parts of ${}^r x_{4i+5}$, ${}^r x_{4i+5}$, ${}^r x_{4i+10}$, ${}^r x_{4i+15}$.

Λ_S is invertible by Requirement 1, and the invertibility of Z is proven in Lemma 17 (Appendix A). Therefore transformation (14) is bijective, and by induction on the round index, the correspondence is one-to-one for all regular rounds. For the last round, we have

$$G(c_j) = \Lambda_S(G({}^{Nr}x_{5 \cdot j})) + G({}^{Nr}k_j) \tag{16}$$

Due to the invertibility of Λ_S, the transformation in the last round is also bijective. □

Lemma 7. *If the random parts of the plaintext redundant-bytes $G(p_j)$ are distributed independently and uniformly in $Ker(H)$, then so are the random parts of the redundant-bytes of the state $G({}^r x_j)$ for any round index r, regardless of the expanded key.*

Proof. Trivially follows from Lemma 6. □

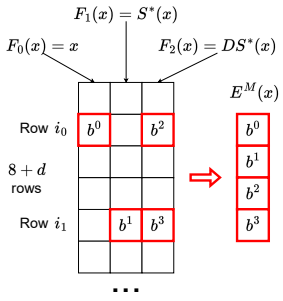


Figure 2

3.3.2 Requirements Related to the Uniformity

According to Requirements 1 and 2, the functions S^*, D^* and DS^* are fully defined by the quintuple $\langle H, \Lambda_S, \Delta_S, \Lambda_D, \Delta_D \rangle$. We will call such a quintuple a *LUT suite*.

Definition 4. Let $\langle H, \Lambda_S, \Delta_S, \Lambda_D, \Delta_D \rangle$ be a LUT suite with redundancy d. Let's consider an arbitrary $M = \{\langle t_0, i_0 \rangle, \dots, \langle t_{|M|-1}, i_{|M|-1} \rangle\} \subseteq \{0, 1, 2\} \times \{0, \dots, 7 + d\}$. Denoting $F_0 = I$ (the identity function), $F_1 = S^*$, $F_2 = DS^*$, we define $E^M(x) : R \to \mathbb{Z}_2^{|M|}$ as the function that maps any vector $x \in R$ to the vector with coordinates $(F_t(x)^i : \langle t, i \rangle \in M)$.

In other words, E^M maps any redundant-byte $x \in R$ to a subset of bits of x, $S^*(x)$, $DS^*(x)$, where each pair $\langle t, i \rangle \in M$ defines one bit out of this subset. Namely, t defines from which redundant-byte (x, $S^*(x)$, $DS^*(x)$) the bit is taken, and i defines its index in the redundant-byte.

Note that since both S^* and DS^* are affine at each coset of H, with the same linear part (Λ_S and Λ_{DS}, respectively) and different free terms, the same applies to E^M.

For $M = \{\langle t_0, i_0 \rangle, \dots, \langle t_k, i_k \rangle\}$, we'll denote the set of all bit indices represented in M

$$N(M) = \{i \mid \exists t : \langle t, i \rangle \in M\} = \{i_0, \dots, i_{|N(M)|-1}\} \tag{17}$$

where the indices in $N(M)$ are numbered in any predefined order, e.g., in ascending order.

We'll also denote

$$M^j = \{t \mid \langle t, i_j \rangle \in M\} \neq \varnothing \tag{18}$$

for $i < |N(M)|$.

Example 2. (Illustrating the M, $N(M)$ and M^j)

In Figure 2, $M = \{\langle 0, i_0 \rangle, \langle 1, i_1 \rangle, \langle 2, i_0 \rangle, \langle 2, i_1 \rangle\}$, $N(M) = \{i_0, i_1\}$, $M^0 = \{0, 2\}$, $M^1 = \{1, 2\}$. The bits of $E^M(x)$ are $b_0 = x^{i_0}$, $b_1 = (S^*(x))^{i_1}$, $b_2 = (DS^*(x))^{i_0}$, $b_3 = (DS^*(x))^{i_1}$.

Definition 5. E^M is called *uniform* if $E^M(C_1) = E^M(C_2)$ for any cosets C_1, C_2 of H.

Requirement 3. *For (1,1,0)-robustness in the 1-probing model, the linear map H must be 1-uniform, and E^M must be uniform for any M such that $|N(M)| = 1$ and $\{0, 1\} \nsubseteq M^0$.*

(The requirement $\{0, 1\} \nsubseteq M^0$ means that the x^i and $(S^*(x))^i$ are never present in M together.)

Requirement 4. *For (1,1,0)-robustness in the 2-probing model, the linear map H must be 2-uniform, and E^M must be M-uniform for any M such that $|N(M)| = 2$, $\{0, 1\} \nsubseteq M^0$, and $\{0, 1\} \nsubseteq M^1$, and for any M such that $|N(M)| = 1$.*

In other words, the sets M that contain two different bit indices are restricted to those in which the bit from x and from $S^*(x)$ are not present together for any of the indices. For the sets that contain only one bit index, there is no such restriction.

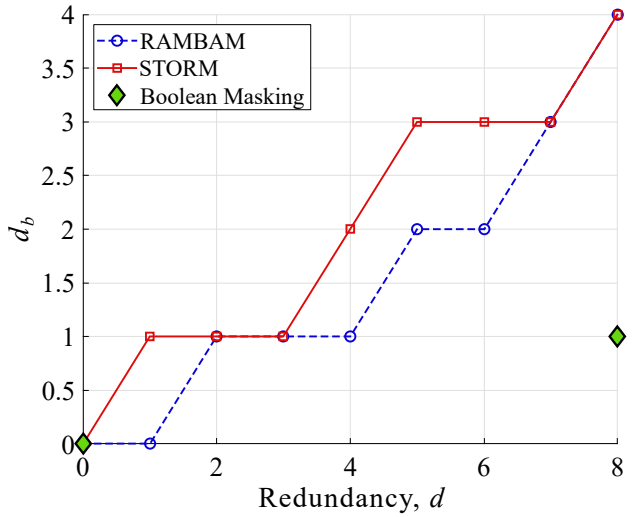


Figure 3: d_b versus the redundancy d for RAMBAM, STORM, and (conventional) Boolean Masking

3.4 Additional metrics and protection against SIFA-1

In [LMMS23] the authors build a bijection between RAMBAM with $d = 8$ and a representation in two shares and state that this bijection proves the equivalence between these two schemes. However, while this bijection is instrumental for analyzing RAMBAM with probing-theory related tools (such as PROLEAD) which requires a representation in shares as its input, it cannot be seen as an equivalence as suggested in [LMMS23]. For such an equivalence to be meaningful, it must preserve security metrics. As an example, it must preserve bit-level security order d_b which is defined as the maximal number such that the simultaneous probing of any set of d_b bits provides no information regarding clear values (see for example [BCC+14, KP21]). This security metric is not preserved under the above-mentioned bijection, as illustrated in Figure 3 which shows the dependency of d_b on the number d of additional bits in the representation of one byte for RAMBAM, for STORM, and for conventional Boolean masking.

Protection against SIFA-1 follows naturally; In SIFA-1, the attacker provides random plaintexts, and for each plaintext performs two AES encryptions — one without faults, the other with simultaneous faults (e.g., force to 0) into several bits of the internal state after the 8-th round (for AES128). Then, he deduces the last round key by analyzing the subset of ciphertexts that are identical, with and without faults (the "ineffective faults" which occur when the bits forced to 0 are already 0). If the set of bits in which the faults are injected bears no information regarding sensitive values, clearly such a deduction is impossible. Therefore, if the number of bits that the attacker can simultaneously force to 0 is $\leq d_b$, SIFA-1 is impossible. For masking over $GF(2^8)$ with the minimal possible number of shares, 2, the codeword size is 16 bits, and $d_b = 1$. On the other hand, for RAMBAM with redundancy 7 (a 15-bit codeword) and for STORM with redundancy 5 (a 13-bit codeword) with a carefully chosen ring or vector space $d_b = 3$, and with redundancy 8 for RAMBAM or STORM (a 16-bit codeword) $d_b = 4$, so RAMBAM and STORM are significantly more protected against SIFA-1 than masking schemes with an equal codeword size.

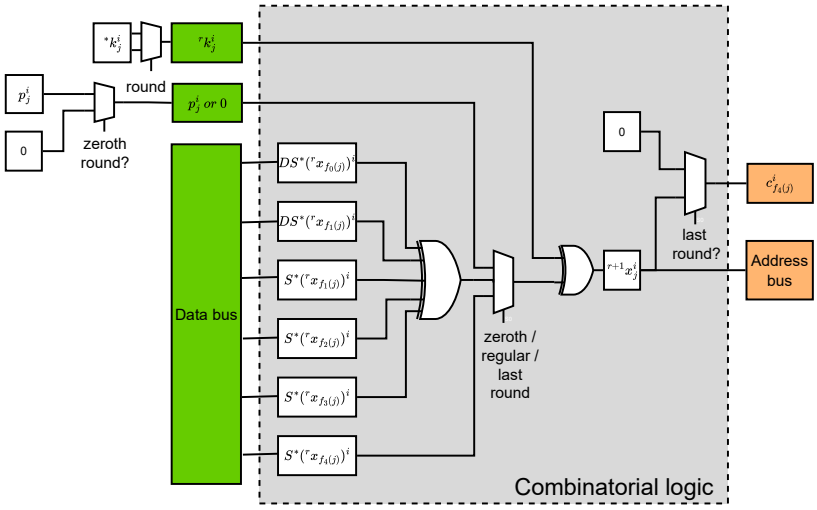


Figure 4

4 Proof of Security of STORM1

Since STORM uses memory, its total power consumption is the sum of the power consumption in the gates and the power consumption in the memory. In this section, we will prove STORM's security in the $(1, 1, 0)$-robust 1-probing and 2-probing models assuming that the power consumption of memory accesses does not reveal the accessed address and data, and therefore can be disregarded (which is the setting for STORM1).

For the purpose of the proof, we represent the data path by a combinational circuit that consists of many instances of a circuit functionally equivalent to the circuit shown in Figure 4 using the notation introduced in Section 3.3. Each such instance calculates a single bit x_j^i of the internal state at all rounds r.

We assume a parallel implementation with 16 instances of the table T^*, each one in its own memory, so that it is possible to calculate a round per clock cycle. In this implementation, no data-containing registers are needed. At each clock cycle the data from the data buses is processed, and the resulting data is sent to the address buses. The inputs of the combinational part of the scheme are the data buses and the input lines (randomized plaintext and key). Its outputs are the address buses and the output lines (ciphertext). The clock cycles are numbered as follows:

- Clock cycle 0 — the initial `AddRoundKey` (the "zeroth round")
- Clock cycles $1 - (Nr - 1)$ — the regular rounds
- Clock cycle Nr — the last round

The functions f_0, f_1, f_2, f_3, f_4 represent the relationships between the redundant-byte indices of the inputs and of the output. f_0, f_1, f_2, f_3 are consistent with Algorithm 3.

Note that according to the definition of f_4, after the last round the ciphertext is rotated by one redundant-byte. This tweak is crucial for Lemma 8 below (which is necessary for our security proof) to hold.

Lemma 8. *For* $u \in \{1, 2, 3, 4\}$, $f_u(j) \neq j$.

Proof. Trivially follows from the table definition of the functions (Table 1). □

To define a probe in the $(g, t, 0)$-robust n-probing model, we need to define both the wire and the clock cycle (= round).

Table 1: Relationships between Redundant-byte Indices in the Round Transformation

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
f_0	0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11
f_1	5	10	15	0	9	14	3	4	13	2	7	8	1	6	11	12
f_2	10	15	0	5	14	3	4	9	2	7	8	13	6	11	12	1
f_3	15	0	5	10	3	4	9	14	7	8	13	2	11	12	1	6
f_4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0

When $g = 1$ (which is the case in our proof), when choosing the basic probe set, it is enough to take into account only the output wires ${}^{r+1}x^i_j$ ($r \geq 0$) for the basic probe set, because they provide maximal extended sets of probes.

For the transition-extension, for each probe ${}^{r+1}x^i_j$ in the basic probe set we add another probe ${}^{r+2}x^i_j$ at the same wire, one clock cycle later.

According to the definition of the glitch-extension in [FGDP+18], for each probe ${}^{r+1}x^i_j$ from the basic or transition-extended probe set we need to *add* probes at all wires which affect ${}^{r+1}x^i_j$. Instead, we *replace* it with probes on all the input wires in Figure 4, namely:

- If $r > 0$, then we replace ${}^{r+1}x^i_j$ with ${}^{r}k^i_j$, $DS^*({}^{r}x_{f_0(j)})^i$, $DS^*({}^{r}x_{f_1(j)})^i$, $S^*({}^{r}x_{f_1(j)})^i$, $S^*({}^{r}x_{f_2(j)})^i$, $S^*({}^{r}x_{f_3(j)})^i$, $S^*({}^{r}x_{f_4(j)})^i$. (We do not need to list p^i_j, as these input lines contain zeros at all clock cycles except 0.)

- If $r = 0$, then we replace ${}^{r+1}x^i_j$ with ${}^{r}k^i_j$ and p^i_j. (The inputs at the data bus are irrelevant when $r = 0$, as there were no memory accesses yet.)

We will denote the basic set of probes (consisting of one or two probes) as Q, the same set after the transition extension as Q^t, and the same set after both the transition extension and the glitch replacement as Q^{tg}.

Note. In the following, notations ${}^{r}x^i_j$, $S^*({}^{r}x_j)^i$, $DS^*({}^{r}x_j)^i$, ${}^{r}k^i_j$ are used in two different senses. When we write, e.g., ${}^{r}x^i_j \in Q^t$, or $S^*({}^{r}x_j)^i \in Q^{tg}$, we mean that a specific wire at a specific round is in the set of probes or extended probes, regardless of the specific instance of the AES encryption. In other cases, ${}^{r}x^i_j$ and $S^*({}^{r}x_j)^i$ mean the values (0 or 1) at these wires in a specific instance of AES encryption. The meaning in each case should be clear from the context. We preferred this ambiguity to a further complication of the notation.

We prove that the joint distribution of the values at these probes is independent of the clear components of the plaintext and of the extended key. This implies that they bear no information regarding the sensitive values. For any output of the synthesizer, the tree of the wires which affect ${}^{r+1}x^i_j$ is functionally equivalent to Figure 4 and the values at all wires of this tree are logical functions of the input lines in Figure 4. Therefore, if the values at the input lines jointly bear no information regarding the sensitive values, it applies to all wires of the tree as well.

Lemma 9. *Suppose that:*

1. *For all round indices r and for all redundant-byte indices j, the clear components $H(p_j)$ and $H({}^{r}k_j)$ have arbitrary fixed values, and the random components $G(p_j)$ and $G({}^{r}k_j)$ are distributed independently and uniformly.*
2. *The values of all probes from Q^{tg} up to round r_0 are arbitrarily fixed, where r_0 is an arbitrary fixed round index.*

We denote $N^t_j = \{i : {}^{r_0+1}x^i_j \in Q^t\}$ for any redundant-byte index j. Then:

- *The distribution of ${}^{r_0+1}x_j$ is uniform over some coset of the linear map $H \oplus P^{N^t_j}$.*
- *The distributions of the random components $G({}^{r_0+1}x_j)$ ($0 \leq j < 16$) are independent.*

Proof. Let us arbitrarily fix, in any way consistent with the fixed values of the probes from Q^{tg}, the following random components:

- $G(p_j)$ for $0 \le j < 16$
- $G({}^{r}k_j)$ for $0 \le j < 16$ and $r < r_0$

Each redundant-byte ${}^{r_0+1}x_j$ is a sum of several redundant-bytes related to round r_0, namely:

- ${}^{r_0}k_j + p_j$ for $r_0 = 0$ (the initial `AddRoundKey`)
- ${}^{r_0}k_j + DS^*({}^{r_0}x_{f_0(j)}) + DS^*({}^{r_0}x_{f_1(j)}) + S^*({}^{r_0}x_{f_1(j)}) + S^*({}^{r_0}x_{f_2(j)}) + S^*({}^{r_0}x_{f_3(j)})$ for the regular rounds
- ${}^{r_0}k_j + S^*({}^{r_0}x_{f_4(j)})$ for the last round

Clearly, fixing the clear components of the plaintext and of the expanded key fixes the clear components of all state redundant-bytes $H({}^{r}x_j)$ at all rounds up to r_0 as well, so in all cases, all addends except for the first one (${}^{r_0}k_j$) are fixed. Since for any bit index i the only instance of Figure 4 in which ${}^{r_0}k_j^i$ appears is the instance with output ${}^{r_0+1}x_j^i$, we have $N_j^t = \{i : {}^{r_0}k_j^i \in Q^{tg}\}$, so the value of ${}^{r_0}k_j$ is uniformly distributed in a coset of $H \oplus P^{N_j^t}$. Since there are probes from Q^t at the bits ${}^{r_0+1}x_j^i$ with bit indices from the same set N_j^t, their values are fixed, and the set of the possible values of ${}^{r_0+1}x_j$ is limited by a coset of $H \oplus P^{N_j^t}$. The sum of the variable ${}^{r_0}k_j$ uniformly distributed in a coset of $H \oplus P^{N_j^t}$ with a constant, is also a variable uniformly distributed in a coset of $H \oplus P^{N_j^t}$, which is the set of all the possible values of ${}^{r_0+1}x_j$.

The random component $G({}^{r_0+1}x_j)$ of each redundant-byte ${}^{r_0+1}x_j$ depends on the random component $G({}^{r_0+1}k_j)$ of only one redundant-byte of the expanded key, and these random components are distributed independently. Therefore, the random components $G({}^{r_0+1}x_j)$ of the redundant-bytes are distributed independently as well. □

Lemma 10. *In the $(1,1,0)$-robust 1-probing model ($|Q| = 1$), it cannot happen that ${}^{r}x_j^i \in Q^t$ and $S^*({}^{r}x_j^i) \in Q^{tg}$.*

Proof. From $|Q| = 1$ it follows that $|Q^t| = 2$, and the two probes in Q^t differ only by the round index, but have the same redundant-byte and bit indices.

Suppose that ${}^{r}x_j^i \in Q^t$ and $S^*({}^{r}x_j^i) \in Q^{tg}$. It means that $S^*({}^{r}x_j^i) \in Q^{tg}$ is one of the input lines in some instance of Figure 4 with the output line ${}^{r+1}x_{j'}^i \in Q^t$, i.e., for some $u \in \{1,2,3,4\}$ and for some $0 \le j' < 16$, $j = f_u(j')$. By Lemma 8, $j' \ne j$, and ${}^{r}x_j^i$ and ${}^{r+1}x_{j'}^i$ are two probes from Q^t with different redundant-byte indices, a contradiction. □

Lemma 11. *In the $(1,1,0)$-robust 2-probing model ($|Q| = 2$), if the bit indices of the two probes in Q differ, then it cannot happen that ${}^{r}x_j^i \in Q^t$ and $S^*({}^{r}x_j^i) \in Q^{tg}$.*

Proof. If $|Q| = 2$ and the bit indices i_0, i_1 of the two probes in Q are different, then in Q^t there are two probes with bit index i_0 and two probes with bit index i_1, and in each pair of probes both probes have the same redundant-byte index. Therefore, it cannot happen that two probes from Q^t have the same bit index, but different redundant-byte indices.

Similarly to Lemma 10, suppose that ${}^{r}x_j^i \in Q^t$ and $S^*({}^{r}x_j^i) \in Q^{tg}$. It means that $S^*({}^{r}x_j^i) \in Q^{tg}$ is one of the input lines in some instance of Figure 4 with the output line ${}^{r+1}x_{j'}^i \in Q^t$, i.e., for some $u \in \{1,2,3,4\}$ and for some $0 \le j' < 16$, $j = f_u(j')$. By Lemma 8, $j' \ne j$, and ${}^{r}x_j^i$ and ${}^{r+1}x_{j'}^i$ are two probes from Q^t with different redundant-byte indices, a contradiction. □

Theorem 2. *Assuming that all requirements from Section 3.3 are met, except for Requirement 4, STORM1 is secure in the (1,1,0)-robust 1-probing model. If Requirement 4 is also met, then STORM1 is secure in the multivariate (1,1,0)-robust 2-probing model.*

Proof. We denote the number of probes in the basic set as n. Note that the number of different bit indices in the extended probe set Q^{tg} is at most n.

Due to the initial randomization, the random components $G(p_j)$ and $G({}^{r}k_j)$ are distributed independently and uniformly. We prove that the distribution of the values at the probes in the extended set Q^{tg} does not depend on the clear components of the plaintext and of the expanded key. The proof is by induction by the round index.

At round 0, let's consider the probes from Q^{tg}. They are of the form p_j^i and ${}^{0}k_j^i$ for some redundant-byte indices j and bit indices i. We split these bits into groups according to their sources, i.e., to the redundant-bytes to which they pertain. Since all these redundant-bytes are distributed independently, the distributions of the bits from different groups are also independent of each other. Let's further consider a single group consisting of $n_j \leq n$ bits at the set N_j^{tg} of bit positions. Note that the value of the source redundant-byte given the values at these probes is uniformly distributed in a coset of $H \oplus N_j^{tg}$. H is by assumption n-uniform, and therefore n_j-uniform as well. By Lemma 1, $P^{N_j}(Ker(H)) = \mathbb{Z}_2^{n_j}$, therefore 2^{n_j} cosets of $H \oplus P^{N_j}$ cover $Ker(H)$, so 2^{n_j} cosets of $H \oplus P^{N_j}$ cover any coset of H as well. Since all cosets of $H \oplus P^{N_j}$ have the same size, all 2^{n_j} combinations of the values of the probes in the group have the same probability.

At round $r_0 + 1 > 0$, we assume that at all the earlier rounds the values at the probes from Q^{tg} are fixed, and we will prove that the joint distribution of the values at the probes from Q^{tg} at round $r_0 + 1$ does not depend on the clear components.

We denote:

- $F_0 = I$, $F_1 = S^*, F_2 = DS^*$ as in Definition 4
- $N_j^t = \{i : {}^{r_0+1}x_j^i \in Q^t\}$ as in Lemma 9
- $M_j = \{\langle 0, i\rangle : {}^{r_0+1}x_j^i \in Q^t\} \cup \{\langle t, i\rangle : t \in \{1, 2\}, F_t({}^{r_0+1}x_j)^i \in Q^{tg}\}$

Clearly, $N_j^t \subseteq N(M_j)$, therefore $|N_j^t| \leq |N(M_j)| \leq n$.

We split the probes from Q^{tg} at round $r_0 + 1$ into groups according to their sources, where ${}^{r_0+1}x_j$ is regarded as the source of $S^*({}^{r_0+1}x_j)^i$ and of $DS^*({}^{r_0+1}x_j)^i$. Since each group of bits is a function of a different source, and since by Lemma 9 the sources are distributed independently, what is left to prove is that the distribution of the bit values inside each group is jointly uniform and does not depend on the clear components.

$E^{M_j}({}^{r_0+1}x_j)$ represents the values at all probes from Q^t which pertain to ${}^{r_0+1}x_j$ and all probes from Q^{tg} whose source is ${}^{r_0+1}x_j$.

In the $(1, 1, 0)$-robust 1-probing model, according to Lemma 10 it cannot happen that ${}^{r_0+1}x_j^i \in Q^t$ and $S^*({}^{r_0+1}x_j^i) \in Q^{tg}$, therefore $\{0, 1\} \nsubseteq M_j^0$, and by Requirement 3 E^{M_j} must be uniform.

In the $(1, 1, 0)$-robust 2-probing model, there are two cases.

If the two probes of Q have the same bit index i, then all probes of Q^{tg} have the same bit index i, $|N(M_j)| = 1$, and by Requirement 4 E^{M_j} must be uniform.

If the two probes have different bit indices i_0, i_1, then by Lemma 11 for each one of them it cannot happen that ${}^{r}x_j^i \in Q^t$ and $S^*({}^{r}x_j^i) \in Q^{tg}$, therefore $\{0, 1\} \nsubseteq M_j^0$ and $\{0, 1\} \nsubseteq M_j^1$, and by Requirement 4 E^{M_j} must be uniform.

Summarizing, E^{M_j} is uniform in all cases. Due to this uniformity, for any coset C of H, $Y = E^{M_j}(C)$ does not depend on the choice of the coset, i.e., on the clear components. According to Lemma 9, given the values at the probes from Q^{tg} up to round r_0, ${}^{r_0+1}x_j$ is distributed uniformly in a coset $C' \subseteq C$ of $H \oplus P^{N_j^t}$. $Y' = E^{M_j}(C') \subseteq Y$ is the subset of Y with fixed values of bits ${}^{r_0+1}x_j^i$ for $i \in N_j^t$, and this subset does not depend on the clear components as well. Due to the affinity of E^{M_j}, and to the uniformity of the distribution of ${}^{r_0+1}x_j$ in C', the distribution in Y' (i.e., the distribution of bits from N_j^{tg}) is also uniform. □

Example 3. (Illustrating the last paragraph of the proof)

Let Figure 2 represent the case with $M_j = \{\langle 0, i_0\rangle, \langle 1, i_1\rangle, \langle 2, i_0\rangle, \langle 2, i_1\rangle\}$. In this case, if ${}^{r_0+1}x_j \in Ker(H)$, then Y is the set of all combinations of the four bits x^{i_0}, $S^*(x)^{i_1}$, $DS^*(x)^{i_0}$, $DS^*(x)^{i_1}$ for $x \in Ker(H)$, and due to the uniformity of E^{M_j}, the same set of combinations corresponds to any coset of H. In this case, $N_j^t = \{i_0\}$, and Y' is the set of all possible combinations of bits $S^*(x)^{i_1}, DS^*(x)^{i_0}, DS^*(x)^{i_1}$ consistent with a specific value of x^{i_0}. Naturally, this set does not depend on the clear component (the coset of H from which ${}^{r_0+1}x_j$ is chosen) as well.

5 Construction of LUT Suites which are (1,1,0)-Robust in 1-Probing and 2-Probing Models in STORM1

In Section 4 we prove the security of STORM1, assuming that the requirements listed in Section 3.3.1 hold. In this section we describe how to construct LUT suites of redundancy 2 for security in the (1,1,0)-robust 1-probing model, and of redundancy 6 in the (1,1,0)-robust 2-probing model.

In the following, we assume an arbitrary LUT suite $\langle H, \Lambda_S, \Delta_S, \Lambda_D, \Delta_D\rangle$ and an arbitrary fixed basis $\{v_0, \dots v_{d-1}\}$ of $Ker(H)$.

Definition 6. $\Xi^M(x) : Ker(H) \to \mathbb{Z}_2^{|M|}$ is the function that maps any vector $x \in Ker(H)$ to the vector with coordinates $(\Phi_t(x)^i : \langle t, i\rangle \in M)$, where $\Phi_0 = I$ (the identity function), $\Phi_1 = \Lambda_S$, $\Phi_2 = \Lambda_{DS}$.

Lemma 12. *Ξ^M is linear, and at any coset of H, $E^M(x) = \Xi^M(G(x)) + const$.*

Proof. Trivially follows from the formulae for S^* (5) and for DS^* (11), and from the definition of E^M (Definition 4) and of Ξ^M (Definition 6). □

Definition 7. Ξ^M is called *degenerate* if $dim(\Xi^M(Ker(H))) < |M|$, and *non-degenerate* otherwise.

Lemma 13. *If Ξ^M is non-degenerate, then E^M is uniform.*

Proof. If Ξ^M is non-degenerate, then $dim(\Xi^M(Ker(H))) = |M|$, and therefore $\Xi^M(Ker(H)) = \mathbb{Z}_2^{|M|}$, i.e., Ξ^M assumes all possible values, therefore at any coset of H, $E^M(x) = \Xi^M(x) + const$ assumes all possible values as well. □

Definition 8. Ξ^M is called *minimal degenerate* if it is degenerate, but for any proper subset $M' \subset M$, $\Xi^{M'}$ is non-degenerate.

Definition 9. For any $K \subseteq \{0, 1, 2\}$, the $(8 + d) \times d$ matrix W^K is defined as follows: $W^K_{ij} = (\sum_{k\in K} \Phi_k(v_j))^i$, where $\Phi_0 = I$ (the identity function), $\Phi_1 = \Lambda_S$, $\Phi_2 = \Lambda_{DS}$.

Note that for $K = \{0\}$, $K = \{1\}$ and $K = \{2\}$, the columns of W^K are v_j, $\Lambda_S(v_j)$ and $\Lambda_{DS}(v_j)$, respectively, and matrix W introduced in Section 3.2 is $W^{\{0\}}$ as defined by Definition 9, so Lemmas 3 and 4 are applicable to $W^{\{0\}}$.

In the following, to specify a LUT suite including a basis of $Ker(H)$, we provide matrices $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$ and functions Δ_S, Δ_{DS}. Indeed, $W^{\{0\}}$ defines $Ker(H)$ and thus H, $W^{\{0\}}$ and $W^{\{1\}}$ together define Λ_S, and $W^{\{0\}}$ and $W^{\{2\}}$ together define Λ_{DS}.

Lemma 14. *For any $K \subseteq \{0, 1, 2\}$, $W^K = \sum_{t\in K} W^{\{t\}}$.*

Proof. Trivially follows from Definition 9. □

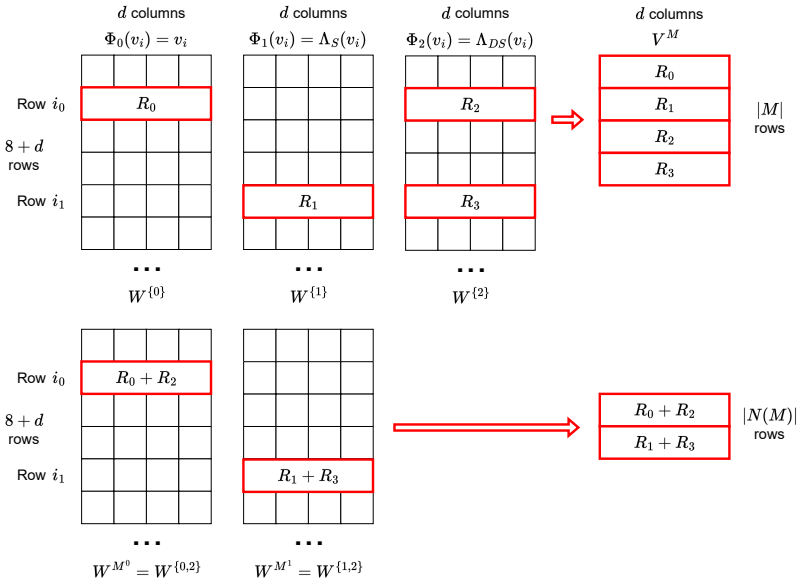


Figure 5

Lemma 15. *If Ξ^M is minimal degenerate, then*

$$\sum_{\langle t,i\rangle \in M} W_i^{\{t\}} = \sum_{j<|N(M)|} W_{i_j}^{M^j} = 0 \tag{19}$$

where W_i^K is, of course, the i^{th} row of the matrix W^K.

Proof. We'll prove that $\sum_{\langle t,n\rangle \in M} W_i^{\{t\}} = 0$; then $\sum_{j<|N(M)|} W_{i_j}^{M^j} = 0$ will follow by Lemma 14.

Let's define V^M as the $|M| \times d$ matrix with $\{W_i^{\{t\}} | \langle t,i\rangle \in M\}$ as its rows.

Clearly, column i of V^M is $\Xi^M(v_i)$. Since the vectors v_i span $Ker(H)$, it is easy to see that $\Xi^M(Ker(H)) = V^M(Ker(H))$, and $dim(\Xi^M(Ker(H))) = rank(V^M)$, so for a degenerate Ξ^M the rank is lower than the number of rows $|M|$, and there must be a linear dependency between the rows of V^M. On the other hand, since for any proper subset $M' \subset M$ the function $\Xi^{M'}$ is not degenerate, any subset of the rows of V^M must be linearly independent. Therefore, the coefficients of all rows in the linear dependency must be non-zero, i.e., 1 — which means that the sum of all rows of V^M must be 0. □

Example 4. Figure 5 illustrates W^K, V^M and the two equal sums from (19).

In the following subsections we will describe how to build a LUT suite which is (1,1,0)-robust in the n-probing model (with $n \in \{1,2\}$).

5.1 1-Probing Model

In this section we show that for (1,1,0)-robustness in the 1-probing model, redundancy 2 is sufficient, and provide an example of such a LUT suite.

5.1.1 Sufficient Condition for (1,1,0)-Robustness of STORM1 in the 1-Probing Model

Lemma 16. *If there are no all-zero rows in $W^{\{0\}}$, there are no all-zero rows in $W^{\{1\}}$ and $W^{\{2\}}$ as well.*

Proof. $W^{\{1\}}$, $W^{\{2\}}$ are bases of $Ker(H)$, like $W^{\{0\}}$, due to the invertibility of Λ_S and Λ_{DS}. If there is an all-zero row in $W^{\{1\}}$ or in $W^{\{2\}}$, it means that one of the coordinates of all vectors from $Ker(H)$ is zero — in contradiction to the assumption that there are no all-zero rows in $W^{\{0\}}$. □

Theorem 3. *If there are no all-zero rows in $W^{\{0\}}$ and for each bit index $0 \le i < 8+d$, $W_i^{\{0\}} \neq W_i^{\{2\}}$ and $W_i^{\{1\}} \neq W_i^{\{2\}}$, then the LUT suite is (1,1,0)-robust in the 1-probing model, regardless of Δ_S and Δ_{DS}.*

Proof. Since there are no all-zero rows in $W^{\{0\}}$, by Lemma 3 H is 1-uniform.

Suppose $|N(M)| = 1$ and $\{0,1\} \nsubseteq M$ (see Requirement 3). If $|M| = 1$, then the matrix V^M has one row, which by assumption cannot be all-zeros, so $dim(\Xi^m(Ker(H))) = 1 = |M|$, Ξ^M is non-degenerate and by Lemma 13 uniform. If $|M| \ge 2$, then $M = \{0,2\}$ or $M = \{1,2\}$, and the two rows of the matrix V^M are either $W_i^{\{0\}}$ and $W_i^{\{2\}}$, or $W_i^{\{1\}}$ and $W_i^{\{2\}}$. In both cases, the rows by assumption do not match, so $dim(\Xi^m(Ker(H))) = 2 = |M|$, Ξ^M is non-degenerate and by Lemma 13 E^M is uniform.

We have shown that in all cases Requirement 3 holds, therefore by Theorem 2 the LUT suite is (1,1,0)-robust in the 1-probing model. □

5.1.2 Construction of a LUT Suite with Redundancy 2 that Guarantees the (1,1,0)-Robustness of STORM1 in the 1-Probing Model

Matrix $W^{\{0\}}$ must have no all-zero rows, and each of the matrices which define $\Lambda_D : Ker(H) \to Ker(H)$ and $\Lambda_{DS} : Ker(H) \to Ker(H)$ in the basis $\langle v_0, v_1 \rangle$ must be either $L_0 = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right)$ or $L_1 = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, with $\Lambda_S = \Lambda_D^{-1} \circ \Lambda_{DS}$ (see (10)). Both $\Delta_S : F \to Ker(H)$ and $\Delta_D : F \to Ker(H)$ can be chosen arbitrarily.

Indeed, in a pair of rows $\langle W_i^{\{0\}}, W_i^{\{2\}} \rangle$ or $\langle W_i^{\{1\}}, W_i^{\{2\}} \rangle$ the second row can be produced from the first row by application of L_0 or L_1. It is easy to check that both L_0 or L_1 have no non-zero fixed points, so the assumptions of Theorem 3 hold, and the LUT-suite is (1,1,0)-robust in the 1-probing model.

5.1.3 Example of a LUT Suite with Redundancy 2 that Guarantees the (1,1,0)-Robustness of STORM1 in the 1-Probing Model

The matrices $W^{\{0\}} = W^{\{1\}}$ and $W^{\{2\}}$ are defined by (20). Functions $\Delta_S : F \to Ker(H)$ and $\Delta_{DS} : F \to Ker(H)$ can be chosen arbitrarily. It is easy to check that there are no all-zero rows and the rows with the same index are different between $W^{\{0\}} = W^{\{1\}}$ and $W^{\{2\}}$.

$$W^{\{0\}} = W^{\{1\}} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad W^{\{2\}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \tag{20}$$

By presenting this LUT suite, we have proven the following

Theorem 4. *There exists a LUT suite with redundancy 2 that guarantees the (1,1,0)-robustness of STORM1 in the 1-probing model.*

5.2 2-Probing Model

In this section, we first describe how to build a LUT suite that guarantees the (1,1,0)-robustness of STORM1 in the 2-probing model with redundancy 7 (Section 5.2.1), but do not present a specific LUT suite. Instead, we proceed to redundancy 6. In Section 5.2.2, we describe more intricate techniques which are necessary to build a protected LUT suite with redundancy 6 and at the same time make it possible to shrink the table record size from 28 to 17 bits. In this paper, this is the only case in which the choice of functions Δ_S and Δ_{DS} is essential.

5.2.1 Sufficient Condition for the (1,1,0)-Robustness of STORM1 in the 2-Probing Model, and Redundancy 7

Theorem 5. *If in the five matrices $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$, $W^{\{0,2\}}$, $W^{\{1,2\}}$ there are no all-zero rows and no matching rows, then the LUT suite defined by matrices $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$ guarantees the (1,1,0)-robustness of STORM1 in the 2-probing model.*

Proof. 2-uniformity of H follows from Lemma 4.

We are going to prove that all functions Ξ^M with either $|N(M)| = 2$ and $\{0,1\} \nsubseteq M^0$ and $\{0,1\} \nsubseteq M^1$, or $|N(M)| = 1$ (see Requirement 4), are non-degenerate.

Suppose that some of these functions are degenerate. Then a minimal degenerate function Ξ^M must exist among them.

If $|M| = 1$, it means that the only row of V^M is all-zeros, which cannot happen by assumption. Therefore, $|M| \geq 2$.

If for this function $|N(M)| = 2$, then by Lemma 15

$$\sum_{j<|N(M)|} W_{i_j}^{M^j} = 0 \tag{21}$$

or equivalently

$$W_{i_0}^{M^0} = W_{i_1}^{M^1} \tag{22}$$

Since M^0 is non-empty and $\{0,1\} \nsubseteq M^0$, it is easy to see that M^0 can be one of the following: $\{0\}$, $\{1\}$, $\{2\}$, $\{0,2\}$, $\{1,2\}$. The same applies to M^1, so there is at least one pair of matching rows in the five matrices $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$, $W^{\{0,2\}}$, $W^{\{1,2\}}$.

If $|N(M)| = 1$, then $|M^0| = |M|$. Since $|M| \geq 2$, we can represent (not necessarily uniquely) $M^0 = M_0^0 \cup M_1^0$, where M_0^0 and M_1^0 are disjoint non-empty sets, neither of which is a superset of $\{0,1\}$, which means that

$$W_{i_0}^{M_0^0} = W_{i_0}^{M_1^0} \tag{23}$$

so in this case we also have a pair of matching rows in five matrices $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$, $W^{\{0,2\}}$, $W^{\{1,2\}}$.

In other words, if all rows in the five matrices are not all-zeros and are different, then functions Ξ^M for all sets M listed in Requirement 4 are non-degenerate and therefore the corresponding functions E^M are uniform; additionally, by Theorem 1 H is 2-uniform, and by Theorem 2 the LUT suite guarantees the (1,1,0)-robustness of STORM1 in the 2-probing model. □

Using Theorem 5, it is easy to find a (1,1,0)-secure LUT suite in the 2-probing model with redundancy 7, since there are $5(8+7) = 75$ rows in the five matrices, and the set of non-zero values of a row is $2^7 - 1 = 127$. We will not present an example of such a LUT suite, since, in fact, redundancy 6 suffices, as we will explain in Section 5.2.2, including a specific example of such a LUT suite.

5.2.2 Construction of LUT Suites for Redundancy 6

With redundancy 6 there are $5(8+6) = 70$ rows in the five matrices, and only $2^6 - 1 = 63$ different possible values, so there are at least 7 matching pairs of rows, and the functions Ξ^M corresponding to these pairs of rows are degenerate. Still, assuming that there are no pairs of matching rows within each one of the tables $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$ (as it would violate the 2-uniformity of H), it may be possible to ensure that they are uniform despite being degenerate, with an appropriate choice of the free term $Sbox(H(x))+\Delta_S(H(x))$ of function S^*, and the free term $2 \cdot Sbox(H(x)) + \Delta_{DS}(H(x))$ of function DS^*, in order to ensure that the free term of E^M assumes the same value, (say, 0) in all cosets of H, and therefore $E^M(C)$ is the same in any coset C of H, i.e, E^M is uniform. Of course, $Sbox(H(x))$ and $2 \cdot Sbox(H(x))$ are fixed for any coset of H, but $\Delta_S(H(x))$ and $\Delta_{DS}(H(x))$ can be chosen arbitrarily. For any coset of H, each pair of matching rows imposes a linear equation on the values of $\Delta_S(H(x))$ and $\Delta_{DS}(H(x))$. The coefficients in these linear equations are the same for all cosets, but the free terms differ. Since the values of both $\Delta_S(H(x))$ and $\Delta_{DS}(H(x))$ are in a 6-dimensional space $Ker(H)$, the matrix M corresponding to this system of equations may (or may not) be invertible up to 12 equations, i.e., up to 12 matching pairs of rows. LUT suites with redundancy 6, at most 12 matching pairs of rows in the five matrices, and an invertible matrix M are extremely rare, but we found some.

There is an additional consideration when choosing a specific LUT suite. Suppose there is a matching pair of rows between table $W^{\{1\}}$ and $W^{\{2\}}$. It means that a bit of $S^*(x)$ always matches a bit of $DS^*(x)$, i.e., there are two identical bits at fixed positions in every record, and it is possible to compress the table by dropping one of these identical bits. Moreover, if there is a matching pair of rows between table $W^{\{0\}}$ and $W^{\{1\}}$ (or $W^{\{0\}}$ and $W^{2\}}$, it means that a bit of x always matches a bit of $S^*(x)$ (or $DS^*(x)$). In this case, it is possible to copy some bits of the address to a register, and at the next clock cycle read them from this register rather than from the table, thus also shrinking the record size. On the other hand, if at least one of the rows in a matching pair is from table $W^{\{0,2\}}$ or $W^{\{1,2\}}$, although functionally it would work out correctly to drop one of the bits involved in the linear dependency and calculate it as the sum of all other involved bits, it would invalidate our security proof. Therefore, if there are $n \leq 12$ pairs of matching rows and each pair comes from the three distinct tables $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$ (where each of the two rows of the pair comes from a separate table), then the table record is compressible from 28 bits to $28 - n$ bits.

We searched for LUT suites with at most 12 matching pairs of rows in the five matrices aiming for the highest possible level of compression. We found more than 44K such LUT suites compressible to 17-bit records, but not a single LUT suite compressible to 16-bit records. Our hypothesis is that they do not exist, but presently we cannot prove it.

5.2.3 Example of a LUT Suite with Redundancy 6 which Guarantees the (1,1,0)-Robustness of STORM1 in the 2-Probing Model

A LUT suite with redundancy 6 and 12 matching pairs, 11 of them among matrices $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$, is represented below by $W^{\{0\}}$, $W^{\{1\}}$, $W^{\{2\}}$ (24). The rows that appear twice are highlighted. Matrices $W^{\{0,2\}}$, $W^{\{1,2\}}$ (25) are also shown, with the only matching rows in this pair of matrices highlighted in both. With $\Delta_S(x)$ and $\Delta_{DS}(x)$ appropriately chosen for each $x \in F$, the functions E^M corresponding to these rows are uniform although Ξ^M are degenerate.

$$W^{\{0\}} = \begin{pmatrix} 1&1&1&0&1&1 \\ 0&1&0&0&1&1 \\ 0&0&0&1&1&1 \\ 0&0&1&1&1&0 \\ 0&1&0&1&0&1 \\ 1&1&0&1&1&1 \\ 0&1&0&1&1&0 \\ 1&0&1&1&0&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{pmatrix} \quad W^{\{1\}} = \begin{pmatrix} 0&0&1&1&1&0 \\ 1&0&1&1&1&1 \\ 1&0&0&0&0&0 \\ 1&0&0&1&1&0 \\ 1&1&0&1&1&1 \\ 0&1&1&0&1&0 \\ 1&1&1&1&0&1 \\ 0&0&1&0&1&1 \\ 0&1&1&1&1&1 \\ 1&0&0&1&0&1 \\ 1&1&1&1&1&0 \\ 1&0&1&0&1&0 \\ 1&1&0&0&1&0 \\ 1&1&1&0&0&0 \end{pmatrix} \quad W^{\{2\}} = \begin{pmatrix} 0&1&1&0&0&1 \\ 0&0&0&0&0&1 \\ 1&0&0&0&1&1 \\ 1&0&1&0&0&1 \\ 0&1&1&1&0&0 \\ 0&1&1&1&1&1 \\ 0&0&1&1&0&1 \\ 1&0&1&0&1&0 \\ 1&0&1&1&0&0 \\ 0&1&0&0&0&0 \\ 1&1&0&1&0&0 \\ 1&1&0&0&1&0 \\ 1&0&1&1&1&1 \\ 1&1&1&1&1&0 \end{pmatrix} \tag{24}$$

$$W^{\{0,2\}} = \begin{pmatrix} 1&0&0&0&1&0 \\ 0&1&0&0&1&0 \\ 1&0&0&1&0&0 \\ 1&0&0&1&1&1 \\ 0&0&1&0&0&1 \\ 1&0&1&0&0&0 \\ 0&1&1&0&1&1 \\ 0&0&0&1&1&0 \\ 0&0&1&1&0&0 \\ 0&0&0&0&0&0 \\ 1&1&1&1&0&0 \\ 1&1&0&1&1&0 \\ 1&0&1&1&0&1 \\ 1&1&1&1&1&1 \end{pmatrix} \quad W^{\{1,2\}} = \begin{pmatrix} 0&1&0&1&1&1 \\ 1&0&1&1&1&0 \\ 0&0&0&0&1&1 \\ 0&0&1&1&1&1 \\ 1&0&1&0&1&1 \\ 0&0&0&1&0&1 \\ 1&1&0&0&0&0 \\ 1&0&0&0&0&1 \\ 1&1&0&0&1&1 \\ 1&1&0&1&0&1 \\ 0&0&1&0&1&0 \\ 0&1&1&0&0&0 \\ 0&1&1&1&0&1 \\ 0&0&0&1&1&0 \end{pmatrix} \tag{25}$$

The system of linear equations for any $x \in \hat{F}$ which has to be solved to find Δ_S and Δ_{DS} is

$$\begin{pmatrix} 1&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&0&0&0 \\ 1&0&1&1&1&1&1&0&1&1&1&1&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&1&1&1&0&0 \\ 0&1&1&1&1&1&0&1&1&1&1&1&0&0 \\ 1&1&1&1&1&0&1&1&1&1&1&0&0&0 \\ 1&0&1&0&1&0&1&0&1&0&1&0&0&0 \\ 1&1&0&0&1&0&1&1&0&0&1&0&0&0 \\ 0&1&0&1&0&0&1&1&1&0&0&0&0&0 \end{pmatrix} \begin{pmatrix} d^0 \\ d^1 \\ d^2 \\ d^3 \\ d^4 \\ d^5 \\ s^0 \\ s^1 \\ s^2 \\ s^3 \\ s^4 \\ s^5 \end{pmatrix} = \begin{pmatrix} x^7 \\ 0 \\ (2 \cdot Sbox(x))^1 \\ (Sbox(x))^0 + x^3 \\ (Sbox(x))^1 \\ (Sbox(x))^2 \\ (Sbox(x))^4 + x^5 \\ (2 \cdot Sbox(x))^5 \\ 0 \\ (2 \cdot Sbox(x))^7 \\ 0 \\ (2 \cdot Sbox(x))^7 + x^7 \end{pmatrix} \tag{26}$$

After these equations are solved, it is possible to calculate the free terms $\Delta_S(x)$, $\Delta_{DS}(x)$:

$$\Delta_S(x) = W^{\{1\}} \begin{pmatrix} s^0 \\ s^1 \\ s^2 \\ s^3 \\ s^4 \\ s^5 \end{pmatrix} \qquad \Delta_{DS}(x) = W^{\{2\}} \begin{pmatrix} d^0 \\ d^1 \\ d^2 \\ d^3 \\ d^4 \\ d^5 \end{pmatrix} \tag{27}$$

The solution of (26) is

$$\begin{pmatrix} d^0 \\ d^1 \\ d^2 \\ d^3 \\ d^4 \\ d^5 \\ s^0 \\ s^1 \\ s^2 \\ s^3 \\ s^4 \\ s^5 \end{pmatrix} = \begin{pmatrix} 1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0\,1 \\ 1\,1\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,0\,1\,1\,1\,1\,1 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\ 1\,1\,0\,0\,1\,0\,0\,1\,0\,1\,1\,1 \\ 0\,1\,1\,1\,1\,1\,1\,0\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,1 \\ 0\,0\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1 \\ 1\,0\,1\,0\,0\,0\,0\,1\,1\,1\,1\,1 \end{pmatrix} \begin{pmatrix} x^7 \\ 0 \\ (2 \cdot Sbox(x))^1 \\ (Sbox(x))^0 + x^3 \\ (Sbox(x))^1 \\ (Sbox(x))^2 \\ (Sbox(x))^4 + x^5 \\ (2 \cdot Sbox(x))^5 \\ 0 \\ (2 \cdot Sbox(x))^7 \\ 0 \\ (2 \cdot Sbox(x))^7 + x^7 \end{pmatrix} \tag{28}$$

or equivalently

$$\begin{pmatrix} d^0 \\ d^1 \\ d^2 \\ d^3 \\ d^4 \\ d^5 \\ s^0 \\ s^1 \\ s^2 \\ s^3 \\ s^4 \\ s^5 \end{pmatrix} = \begin{pmatrix} 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,1\,1\,1\,1\,1 \\ 1\,1\,1\,1\,0\,0\,1\,1\,1\,1 \\ 1\,0\,1\,1\,1\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\ 1\,1\,0\,1\,1\,1\,1\,1\,0\,0 \\ 1\,1\,1\,1\,1\,0\,1\,1\,0\,0 \\ 1\,0\,1\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,1\,0 \end{pmatrix} \begin{pmatrix} x^3 \\ x^5 \\ x^7 \\ Sbox(x))^0 \\ Sbox(x))^1 \\ Sbox(x))^2 \\ Sbox(x))^4 \\ (2 \cdot Sbox(x))^1 \\ (2 \cdot Sbox(x))^5 \\ (2 \cdot Sbox(x))^7 \end{pmatrix} \tag{29}$$

Combining (29) with (27) and (24), the direct calculation of $\Delta_S(x)$ and $\Delta_{DS}(x)$ is

$$\Delta_S(x) = \begin{pmatrix} 1\,0\,0\,1\,0\,0\,0\,0\,0 \\ 1\,0\,0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,1\,0\,1\,1\,0 \\ 0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 0\,1\,1\,0\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,1\,1\,0\,1\,0\,1\,0\,1 \\ 1\,0\,0\,1\,1\,0\,0\,1\,0 \\ 1\,1\,1\,1\,1\,1\,1\,0\,1 \\ 1\,0\,0\,1\,1\,1\,0\,0\,1 \\ 0\,1\,1\,0\,1\,1\,1\,1\,0 \\ 1\,0\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,0\,1\,0\,0\,1\,1\,1 \end{pmatrix} \begin{pmatrix} x^3 \\ x^5 \\ x^7 \\ Sbox(x))^0 \\ Sbox(x))^1 \\ Sbox(x))^2 \\ Sbox(x))^4 \\ (2 \cdot Sbox(x))^1 \\ (2 \cdot Sbox(x))^5 \end{pmatrix} \tag{30}$$

$$\Delta_{DS}(x) = \begin{pmatrix} 1&1&0&1&0&1&1&0&1&1 \\ 0&0&0&0&0&0&0&1&0&0 \\ 1&0&1&1&1&0&0&1&1&0 \\ 1&1&0&1&0&0&1&0&1&1 \\ 0&0&1&0&0&1&0&0&0&0 \\ 1&0&0&1&1&0&0&1&1&0 \\ 0&0&1&0&0&1&0&1&0&0 \\ 0&1&1&0&1&1&1&1&0&1 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&1&0&1&1&1&1&1 \\ 1&0&1&1&1&0&0&0&1&0 \\ 1&0&0&1&1&1&0&1&1&0 \\ 1&0&0&1&1&1&0&0&1&0 \end{pmatrix} \begin{pmatrix} x^3 \\ x^5 \\ x^7 \\ Sbox(x))^0 \\ Sbox(x))^1 \\ Sbox(x))^2 \\ Sbox(x))^4 \\ (2 \cdot Sbox(x))^1 \\ (2 \cdot Sbox(x))^5 \\ (2 \cdot Sbox(x))^7 \end{pmatrix} \tag{31}$$

It is possible to directly verify that Requirement 4 holds for this LUT suite. Such a verification completes the proof of the following

Theorem 6. *There exists a LUT suite with redundancy 6 that guarantees the (1,1,0)-robustness of STORM1 in the 2-probing model.*

6 Additional Protections and Security Proof for STORM2

In the previous sections, we assumed the independence of the power consumption related to memory accesses from the address and data (STORM1). In this section, we make no assumptions regarding memory-read related leakage (STORM2). In Section 6.1, we describe two additional protections to be used in STORM2, on top of the proper choice of the LUT suite as described above. In Section 6.2, we introduce a probing model applicable to memory reads. In Section 6.3, we prove the security of STORM2 in this probing model with redundancy 6 by presenting a specific LUT suite.

6.1 STORM2 Specific Protections

6.1.1 Address and Data Randomization (LUT Randomization)

The LUT T^* described above consists of records $T^*[x]$ such that $T^*[x]$, or $T^*[x]\|x$ if the compression described in Section 5.2.2 is used, contains all bits of $S^*(x)$ and of $DS^*(x)$. Let's apply to T^* a classical data and address randomization, and transform T^* into $T^{*'}$ as follows:

$$T^{*'}[x] = T^*[x + \delta a] + \delta d \tag{32}$$

where δa (the *address offset*) and δd (the *data offset*) are chosen at random. (Here, as elsewhere in this document, addition denotes the XOR operation.) For now, let's assume that there are two slots for the LUT, and after each AES encryption the entire LUT is copied from one slot to the other with fresh δa and δd. Then at the address bus we will send $x + \delta a$ instead of x, and on the data bus we will read $T^*[x] + \delta d$ instead of $T^*[x]$, and subsequently apply the appropriate corrections. Clearly, if both δa and δd are distributed uniformly, then both $x + \delta a$ and $T^*[x] + \delta d$ are distributed uniformly and bear no information regarding the clear components. However, this protection may not be enough, as it does not take transitions into account. Indeed, a XOR between two subsequent values on the address bus or the data bus is not masked with the address/data offset anymore, and does bear information regarding the clear components.

In practice, if we rewrite the entire LUT after each AES encryption, the performance drops drastically, which is unacceptable in many cases. It is possible to rewrite the LUT

gradually, several records or even one record at a time, and switch to the other slot once the rewriting is completed. The impact of rewriting one record at a time on the performance is negligible, and it does not affect the protection as long as the leakage from 2^{8+d} AES encryptions ($2^{14} = 16,384$ for $d = 6$) performed with the same offsets is too weak to be exploitable. In our experiments, this is the case.

6.1.2 Random (Dummy) Reads

Additionally, we add reads from random addresses between any two consecutive real reads. There are two ways to add these reads:

- The fast option. Use three rather than two LUT slots. One slot is the target of the LUT rewriting in progress. Each one of the remaining two slots is used for real reads and dummy reads alternately, so that at any two consecutive clock cycles one of the reads is dummy.
- The compact option. Instead of adding a third slot, execute dummy reads during every alternate clock cycle from the only slot in use.

6.2 Probing Model for Memory Reads

In STORM2, to analyze robustness against leakage from memory reads, we suggest the following

Definition 10. A scheme is called *memory-read robust* if:
1. The knowledge of all bits at the address bus for two consecutive clock cycles provides no information regarding sensitive values.
2. The knowledge of all bits at the data bus for two consecutive clock cycles provides no information regarding sensitive values.

Two consecutive clock cycles are mentioned in this definition in order to take transitions into account.

6.3 Existence of LUT Suites which are Memory-Read Robust and (1,1,0)-Robust in the 2-Probing Model

Theorem 7. *There exist LUT suites which are memory-read robust and (1,1,0)-robust in the 2-probing model, provided that the address and data are randomized as described in Section 6.1.1, and dummy reads are performed as described in Section 6.1.2.*

Proof. We will prove that the LUT suite presented in Section 5.2.3 with LUT randomization and dummy reads is memory-read robust and (1,1,0)-robust in the 2-probing model. Its (1,1,0)-robustness in the 2-probing model was already proven in Theorem 6, and clearly the LUT randomization and dummy reads do not adversely affect the proof, so what is left to prove is the memory-read robustness.

For the address bus, the theorem holds for any LUT suite. Indeed, the address is distributed uniformly for both real and dummy reads; besides, the addresses for any two consecutive clock cycles are clearly independent, since one of the reads is random, so there is no information regarding the clear components at the address bus, even when transitions are taken into account.

The situation with the data bus is different, since it is wider than the address bus, so that not all of the possible values appear in the records, and the set of values present in the table depends on the data offset. Therefore, although a value at the data bus can correspond to any clear component, the probability distribution of the read dummy values may differ depending on the clear components. However, for the LUT suite presented

in Section 5.2.3, it is possible to directly verify that the following relationship between $S^*(x)$ and $DS^*(x)$ holds for any x:

$$DS^*(x) = \begin{pmatrix} 0&0&0&0&1&0&1&1&0&0&0&1&1&0 \\ 1&0&0&0&1&0&0&1&1&0&1&1&0&1 \\ 0&0&0&0&0&0&0&0&0&1&1&0&0&1 \\ 0&1&1&0&0&0&1&1&1&1&0&1&0&0 \\ 0&1&0&1&1&0&0&1&0&1&1&0&1&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&1 \\ 0&1&0&0&1&0&0&0&1&1&0&0&1&0 \\ 0&0&0&0&0&0&1&0&0&1&1&1&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0 \end{pmatrix} \cdot S^*(x) \tag{33}$$

Therefore, the set of the record values in the table before randomization is a linear space L. With randomization, the value read from the table is distributed uniformly regardless of the clear component. If the read value is x, the set of all values in the table is the affine space $L + x$, and the value read at the dummy read clock cycle is uniformly distributed in this affine space, regardless of the clear component. □

7 Experimental Section

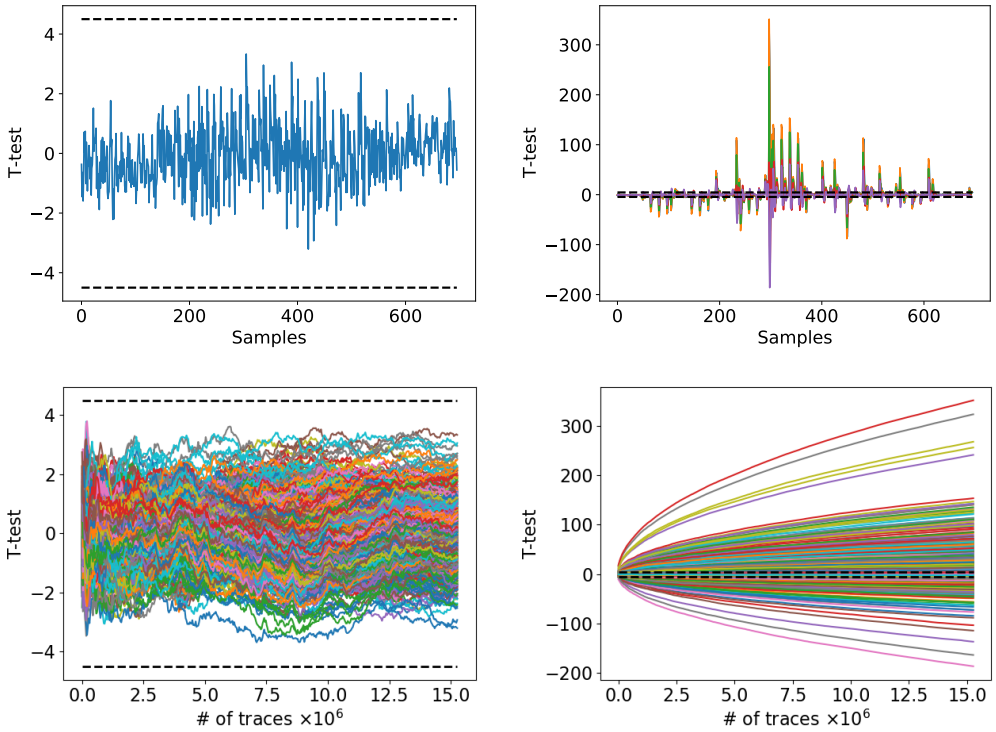
7.1 Setup

Our experiments were performed at `CW305 Artix` FPGA target board by NewAE Technology [OC14]. The traces were collected using the NewAE Technology `ChipWhisperer-Lite` kit at the rate of four samples per clock cycle. The power signal was obtained by measuring the current via a shunt resistor connected serially to the FPGA supply line.

For the experimental evaluation, we used two RTL implementations of AES128, with redundancies 2 (LUT suite as described in Section 5.1.3) and 6 (LUT suite as described in Section 5.2.3), with the same hardwired pre-expanded key $K =$ `2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C`. The LUTs were implemented in the FPGA's BRAM. As this BRAM is leaky, the RTL supports both STORM2 additional protections described in Section 6. The dummy reads are supported in the fast option (three slots, one for real reads, one for dummy reads, and one in which the next randomized version of the LUTs is being built). To enable a sanity check, the RTL supports disabling the randomization — when this option is turned on, all random bits are zeros.

To diminish the internal noise, in our RTL one real read (and one dummy read) is performed at each clock cycle, and each round takes 16 clock cycles. Before each AES encryption, a single record is read from the active table, modified for a different address offset and data offset, and written to the slot where the next LUT is being built (total of 2 clock cycles per AES encryption), so the randomized version of LUT is changed every $2^{8+2} = 1,024$ AES encryptions for redundancy 2 and every $2^{8+6} = 16,384$ AES encryptions for redundancy 6.

The RTL supports execution of a long sequence of AES encryptions, in which approximately half of the encryptions have a fixed plaintext $FP =$ `73 7A A9 06 C2 B3 F8 AF 45 16 FC 97 7C 4C D1 93` and the other half have a pseudorandom plaintext. The order in which the two kinds of the plaintext are interspersed is defined at random, using a separate



(a) First Order (b) Second Order

Figure 6: Warmup Experiment: Redundancy 2 with dummy reads and LUT randomization (a) 1^{st}-Order T-test (b) 2^{nd}-Order T-test (TOP - *vs.* Time Samples, BOTTOM - *vs.* #Traces)

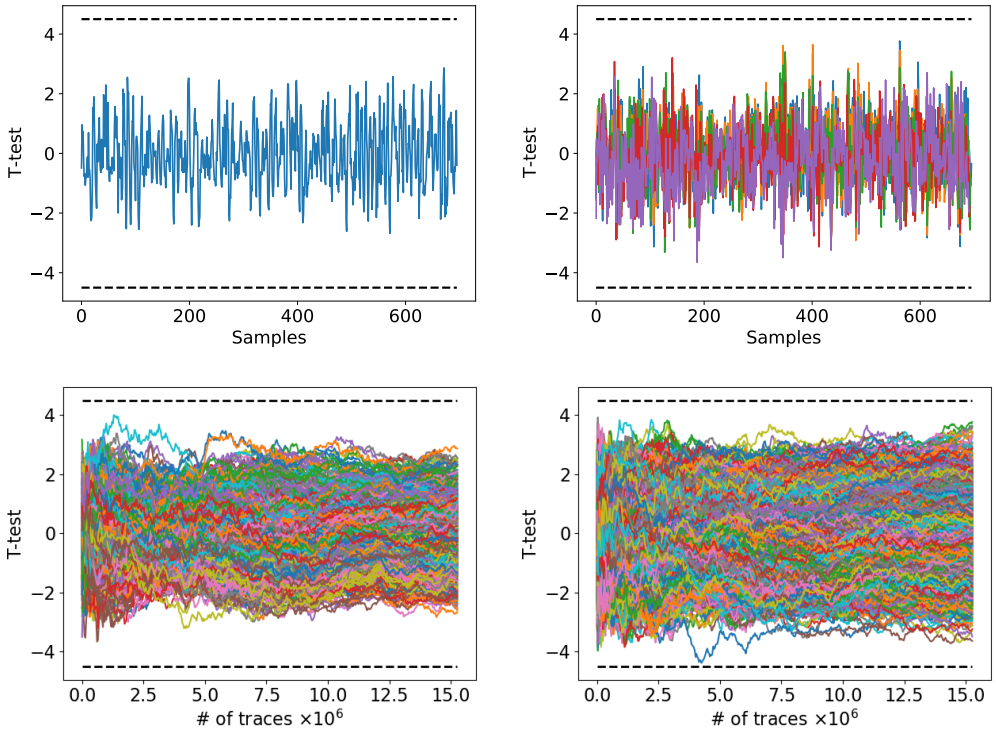
source of randomness which cannot be disabled. The fixed plaintext is chosen so that the input to all S-boxes at the fifth round is all-zeros. To eliminate possible false-positives from the plaintext in the clear, the RTL works according to the following pseudocode:

Algorithm 5: STORM measurement Loop Iteration

Function $EncLoop(S, \kappa, r)$
  Input : $S[2]$ — two randomized plaintexts
    κ — randomized expanded key K
    r — a single random bit
  $res = AesEncLUT^*(10, \kappa, S[0]) + \Delta$ `/* 10 is the number of rounds */`
  if r=*0* **then**
    $S[0], S[1] = res, S[1]$
  else
    $S[0], S[1] = S[1], res$
  end if
end

where:

- S — two slots for randomized plaintext. Initially one of them contains the fixed plaintext FP, and the second one an externally supplied plaintext
- $\Delta = AES_K(FP) + FP$ — the XOR of the fixed plaintext and the corresponding ciphertext



(a) First Order

(b) Second Order

Figure 7: 2^{nd}-Order Security: Redundancy 6 with dummy reads and LUT randomization (a) 1^{st}-Order T-test (b) 2^{nd}-Order T-test (TOP - *vs.* Time Samples, BOTTOM - *vs.* #Traces)

7.2 Security Evaluation

In this section we present the results of experiments in three configurations:

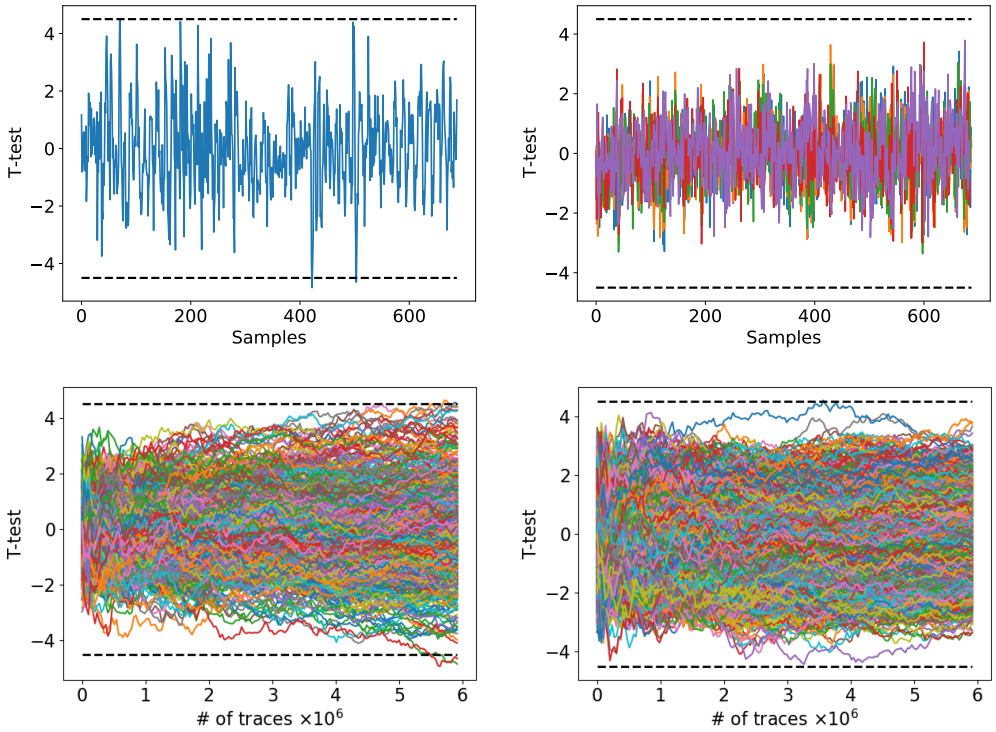
1. Warmup Experiment (1^{st}-Order Security): Redundancy 2 with dummy reads and LUT randomization (15.3M traces)
2. 2^{nd}-Order Security: Redundancy 6 with dummy reads and LUT randomization (15.3M traces)
3. LUT Randomization Disabled: Redundancy 6 with dummy reads, but without LUT randomization (5.95M traces). We stopped this experiment prematurely as the leakage in the first order became apparent.

We will denote the i^{th} sample of the j^{th} trace as x_i^j. Let S_0 and S_1 be the sets of trace indices with the fixed and random clear plaintexts, respectively.

In each configuration, we stored the acquired traces in files, 34K traces in each file. Iteratively, merging these files one-by-one into the current subset of traces, we calculated the following values, denoting the set indices of the traces in the current subset as M:

- First-order t-tests — for each sample index i, the t-test between $\{x_i^j \mid j \in M \cap S_0\}$ and $\{x_i^j \mid j \in M \cap S_1\}$
- Second-order t-tests (**Univariate** and **Multivariate**)— for each pair of sample indices i_0 and i_1 such that $i_0 \leq i_1 \leq i_0 + 4$, the t-test between

$$\{(x_{i_0}^j - \overline{x_{i_0}^j})(x_{i_1}^j - \overline{x_{i_1}^j}) \mid j \in M \cap S_0\})$$



(a) First Order (b) Second Order

Figure 8: LUT Randomization Disabled: Redundancy 6 with dummy reads, but without LUT randomization (a) 1^{st}-Order T-test (b) 2^{nd}-Order T-test (TOP - *vs.* Time Samples, BOTTOM - *vs.* #Traces)

and

$$\{(x^j_{i_0} - \overline{x^j_{i_0}})(x^j_{i_1} - \overline{x^j_{i_1}}) \mid j \in M \cap S_1\})$$

Figures 6, 7 and 8 depict the results of these experiments for configurations item 1, item 2 and item 3, respectively. At each figure there are four plots. The upper row of plots shows the t-test as a function of the sample index, for the first and second orders. For the second order, five graphs are shown at the same plot, corresponding to $i_1 - i_0$ (between 0 and 4). I.e., a value of 0 implies a **univariate** 2^{nd}-order test, and values of 1 to 4 imply a **multivariate** 2^{nd}-order test. In the lower row, each one of the two plots (for the first and second order t-tests) consists of multiple graphs, where each graph represents the t-test of a sample or a pair of samples as a function of the number of tests.

Our conclusions from these graphs:

- For redundancy 2 with dummy reads and LUT randomization:
  - No first-order leakage is observed at 15.3M, confirming Theorem 3.
  - There is a very significant leakage in the second order (starting from $\approx 2.5K$ traces) which illustrates Theorem 1.
  - The leakage in the second order at the same time serves as a sanity check for our measurements with redundancy 2.
  - The highest absolute values of the second order t-test are observed at the fifth round at which all inputs to the S-box are zeros, which serves as an additional sanity check.

Table 2: Tradeoff between security level, RAM size, and latency

Configuration	Clock cycles per LUT read	# of LUT slots	First-order leakage	Second-order leakage
With LUT randomization (**fast**)	1	3	No	No
With LUT randomization (**compact**)	2	2	No	No
Without LUT randomization (**fast**)	1	2	At $\approx 5M$ traces	No
Without LUT randomization (**compact**)	2	1	At $\approx 5M$ traces	No

- For redundancy 6 with dummy reads and LUT randomization:
  - Neither first nor second order leakage is observed at 15.3M, confirming Theorem 5.
- For redundancy 6 with dummy reads, but without LUT randomization:
  - There is a slight leakage in the first order (starting at $\approx 5M$ traces) which proves that the BRAM is leaky and the STORM2 protections are relevant.
  - The same leakage in the first order at the same time serves as a sanity check for our measurements with redundancy 6.
  - No second-order leakage is observed at 5.95M. Together with only the very slight first-order leakage mentioned above, this configuration provides a tradeoff shown in Table 2.

7.3 Area and Performance Evaluation

In this section we investigate the implementation costs of RAMBAM and STORM1/2, and compare them to state-of-the-art schemes [DMRB18, DCRB+16, GMK16, WM18, Sug19]. All area measurements were obtained with the Cadence Genus 21.14-s082_1, using the Open Cell Nangate 45nm library [NAN] and are expressed in 2-input NAND gate equivalents (GEs). For the evaluation of the maximum clock frequencies of STORM1/2 we assume the following timing constraints on the memory block: an input setup of ≈ 0.5 ns and an output delay of ≈ 0.1 ns.

In [BBA+22] (Tables 2 and 3), the area and performance of RAMBAM are compared against the area and performance of other protected AES schemes suggested in literature, with 16 S-boxes (Table 2) and 1 S-box (Table 3). In this section, we expand these tables to include STORM1 (which assumes memory without leakage) and STORM2 (with leaky memory). For STORM2, we consider the compact configuration STORM2C (two LUT slots per S-box computation; each dummy read takes an extra clock cycle) and the fast configuration STORM2F (three LUT slots per clock cycle; dummy reads are performed in parallel with real reads).

In Table 3, besides STORM1 and STORM2, we added Code Based Masking (CBM) [WMCS20]. Although CBM assumes SW implementation, so there is no gate count for comparison, we include it to illustrate the huge difference in the amount of randomness. Regarding the latency per round, although we could not estimate how many clock cycles one round would take in a HW implementation of CBM; it is clear that it would take significantly more than one clock cycle, due to the SNI limitations.

Like other schemes, where the initial randomization is not amortized in the amount of randomness per round or per S-box, we did not count it for RAMBAM and STORM in Tables 3 and 4.

For the estimated maximal clock frequency, we provide data only for STORM and RAMBAM, as we have no data for other schemes.

In Table 5, we compare the number of bits per byte for the initial randomization for STORM against all share-based schemes.

Table 3: AES-128 realization area and performance comparison for schemes with 16 S-boxes

Scheme	Area [KB]	Memory [kGE][1]	Randomness [Bytes/round]	Round latency [cycles]	Max.freq. [MHz]
STORM1 ($d = 6$)	9.0	557	0	1	833
STORM2C ($d = 6$)	12.2	1,049	28	2	700
STORM2F ($d = 6$)	13.7	1,671	28	1	700
RAMBAM ($d = 8$)	78.9[2]	N/A	112[2]	1[2]	212
CBM ([WMCS20])	N/A	N/A	$\geq 4,320$	> 1	N/A
[SBHM20]	123.1	N/A	72	1	

Table 4: AES-128 realization area and performance comparison for schemes with one S-box

Order	Scheme	Area [kGE][1]	Memory [KB]	Randomness [bits/S-box]	AES latency [cycles]	Max.freq. [MHz]
Second	STORM1 ($d = 6$)	8.8	34	0	168	840
	STORM2C ($d = 6$)	9.1	64	14	333	860
	STORM2F ($d = 6$)	9.2	102	14	168	860
	RAMBAM ($d = 8$)	12.1[3]	N/A	56[3]	233[3]	217
	[DMRB18]	10.9[3]	N/A	53[3]	256[3]	
	[DCRB+16]	12.6[3]	N/A	162[3]	276[3]	
	[GMK16]	12.0[3]	N/A	54[3]	492[3]	
First	STORM1 ($d = 2$)	5.3	2	0	168	800
	STORM2C ($d = 2$)	5.7	4	10	333	800
	STORM2F ($d = 2$)	5.7	6	10	168	800
	[DMRB18]	6.6[3]	N/A	19[3]	256[3]	
	[DCRB+16]	7.7[3]	N/A	54[3]	276[3]	
	[GMK16]	7.3[3]	N/A	18[3]	246[3]	
	[WM18]	7.6[3]	N/A	0[3]	2804[3]	
	[Sug19]	17.1[3]	N/A	0[3]	266[3]	

[1] 1 GE = 0.798 μm^2
[2] Based on [BBA+22, Table 2]
[3] Based on [BBA+22, Table 3]

Table 5: # of random bits per byte for the initial randomization

	First order	Second order
STORM	2	6
Share-based schemes	8	16

8 Discussion and Conclusion

In this study we proposed STORM, a non-share-based scheme that synergies the unique algebraic structure proposed in RAMBAM and the security-parameter it provides (the

Table 6: High abstraction level comparison of the main characteristics of CBM, RAMBAM and STORM variants.

	CBM for AES	RAMBAM	STORM1 ($d = 6$)	STORM2 ($d = 6$)
Key idea	Packing multiple secret bytes in a single codeword over $GF(2^8)$	Embedding of $GF(2^8)$ into a polynomial ring over $GF(2)$ (dimension $8 + d$)	Embedding of $GF(2^8)$ into a vector space over $GF(2)$ (dimension $8 + d$) + LUTs	Embedding of $GF(2^8)$ into a vector space over $GF(2)$ (dimension $8 + d$) + LUTs
Algebraic structure	Vector space over $GF(2^8)$	Binary polynomial ring	Vector space over $GF(2)$	Vector space over $GF(2)$
Relies on the algebraic structure of the S-box transformation	Yes	Yes	No	No
Latency HW	> 1 clock cycle per round	1 clock cycle per round	1 clock cycle per round	1 clock cycle per round
Multiplication complexity in the code space (n — code size)	$\geq 4n^2$ multipliers over $GF(2^8)$ ($\geq 256n^2$ AND gates) per round	$16n^2$ multipliers over $GF(2)$ (AND gates) per round	N/A	N/A
Security	provable	asymptotic	provable	provable
Provable Security	Based on symbol-level SNI	N/A	Based on jointly independent distributions	Based on jointly independent distributions and LUT randomization
Assumptions of the proof	SNI	N/A	Leakage-free memory, requirements to LUTs	LUT randomization, requirements to LUTs
Number of random bits per one round	$1920 \cdot d \cdot (d + k)/k$ bytes (at least $960 \cdot d \cdot (d + 16)$ bits for the maximal possible value $k = 16$)	$112 \cdot d$	0	$16(d + 8)$

redundancy, d), with the utilization of look-up-tables (LUTs) in memory. STORM is proposed with two variants, STORM1 which achieves provable security while assuming a *leakage-free* memory, and STORM2 which does not hold such assumptions and randomizes the memory LUTs to achieve provable security.

Like code-based-masking (CBM) [WMCS20], RAMBAM and STORM are code-based schemes in the sense that their set of representations is a code in the vector space $GF(2)^{8+d}$

as listed in Table 6. RAMBAM however requires a richer structure, specifically a ring on $GF(2)^{8+d}$ and a ring-homomorphism. However, as discussed above, RAMBAM and STORM are significantly different from code-based-masking (CBM), where the main idea is that multiple secret bytes are packed in a single codeword over $GF(2^8)$ and non-interference based notions (t-S/NI) are essential to achieving provable security. RAMBAM and STORM take a different approach and use codes over $GF(2)$ without t-S/NI-gadgets, leading to their efficiency and compactness. Namely, in RAMBAM the key idea is the embedding of $GF(2^8)$ into a polynomial ring over $GF(2)$ (dimension $8+d$), whereas in STORM the key idea is the embedding of $GF(2^8)$ into a *vector space* over $GF(2)$ (dimension $8+d$) plus the important utilization of LUTs which enables efficient randomization. Whereas both CBM and RAMBAM rely on the algebraic structure of the AES, STORM, because of its utilization of LUTs, has no such limitation (making it more general in some sense).

The main added value of RAMBAM over CBM is the reduced multiplication complexity in the code space, where CBM requires expensive field multipliers, and RAMBAM only requires multipliers over $GF(2)$ which are far less expensive, albeit with no security proof but with a security parameter, i.e., security is *asymptotic* in RAMBAM. On the other hand, STORM is quite different. While it does utilize the redundancy d just like RAMBAM, all the non-linear operations are performed via memory access. This in turn relaxes the mathematical requirement from a ring to a vector space, enables easy and cheap randomization, and most importantly, allows provability. As discussed, whereas the provable security of CBM is based on symbol-level strong non-interference properties, STORM's provable security is based on jointly independent distributions. For STORM1, a leakage-free memory is assumed, and the basic security proofs from Sections 5.1.3 and 5.2.3 are sufficient. In STORM2 no assumptions are made regarding leakage from memory, and the LUT randomization mechanism on-top of the vector space allows provability.

In addition to other features, perhaps one of the main savings in the STORM variants is its randomness requirements (or the number of random bits per round) as listed in the table: STORM2 requires ≈ 4 times less randomness as compared to RAMBAM, whereas RAMBAM requires more than an order-of-magnitude less randomness as compared to CBM. This reflects direct savings both in randomness-generation electronic cost (or expansion), and in the electronic cost associated with scheme-internal computations with this randomness (e.g., refreshes).

Table 6 summarizes the differences between CBM (for AES), RAMBAM, STORM1, and STORM2.

Thus overall, STORM effectively solves the long-standing challenge of combining high security against SCA with low gate count and high performance for AES implementations by offering a different tradeoff (memory utilization) that may be preferable to RAMBAM in many practical cases. Unlike RAMBAM for which it is experimentally shown that the leakage rapidly decreases as redundancy grows, but lacks a security proof (though the intuition behind this is explained), STORM has proven security. For applications with limited resources (e.g., IoT devices) STORM2 can be configured with a relatively small amount of SRAM, starting from 4 KB (redundancy 2 which provides proven protection against first-order attacks). For applications that require high performance (e.g., servers with intensive encrypted communications) the SRAM size is typically not a limiting factor, and the various advantages of STORM2 compared to other solutions are quite significant.

A Proof of the Invertibility of Matrix Z in Lemma 6

Lemma 17. *Let Λ be an arbitrary (not necessarily invertible) $n \times n$ matrix over an arbitrary field F of characteristic 2. Then the $4n \times 4n$ matrix*

$$Z = \begin{pmatrix} \Lambda & \Lambda + I & I & I \\ I & \Lambda & \Lambda + I & I \\ I & I & \Lambda & \Lambda + I \\ \Lambda + I & I & I & \Lambda \end{pmatrix} \tag{34}$$

is invertible, where I is the $n \times n$ identity matrix.

Proof. [3] First, we represent

$$Z = B + AD \tag{35}$$

where

$$B = \begin{pmatrix} 0 & I & I & I \\ I & 0 & I & I \\ I & I & 0 & I \\ I & I & I & 0 \end{pmatrix} \tag{36}$$

$$A = \begin{pmatrix} I & I & 0 & 0 \\ 0 & I & I & 0 \\ 0 & 0 & I & I \\ I & 0 & 0 & I \end{pmatrix} \tag{37}$$

$$D = \begin{pmatrix} \Lambda & 0 & 0 & 0 \\ 0 & \Lambda & 0 & 0 \\ 0 & 0 & \Lambda & 0 \\ 0 & 0 & 0 & \Lambda \end{pmatrix} \tag{38}$$

It is easy to directly verify that

$$A^4 = 0 \tag{39}$$

$$AD = DA = \begin{pmatrix} \Lambda & \Lambda & 0 & 0 \\ 0 & \Lambda & \Lambda & 0 \\ 0 & 0 & \Lambda & \Lambda \\ \Lambda & 0 & 0 & \Lambda \end{pmatrix} \tag{40}$$

$$(B + I)A = 0 \tag{41}$$

(here, of course, I is the $4n \times 4n$ identity matrix), and therefore

$$BA = A \tag{42}$$

and

$$Z = B + AD = B + BAD = B(I + AD) \tag{43}$$

After these observations, we can explicitly find Z^{-1}:

$$Z^{-1} = (I + AD + (AD)^2 + (AD)^3)B^{-1} \tag{44}$$

Indeed,

$$\begin{aligned} Z^{-1} \cdot Z = (I + AD + (AD)^2 + (AD)^3)B^{-1}B(I + AD) = \\ (I + AD + (AD)^2 + (AD)^3)(I + AD) = I + (AD)^4 = I + A^4D^4 = I \end{aligned} \tag{45}$$

□

[3] We are grateful to Victor Halperin who suggested this proof

References

[AMZ+24] Maitreyi Ashok, Saurav Maji, Xin Zhang, John Cohn, and Anantha P Chandrakasan. A secure digital in-memory compute (imc) macro with protections for side-channel and bus probing attacks. In *2024 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–2. IEEE, 2024.

[BBA+22] Yaacov Belenky, Vadim Bugaenko, Leonid Azriel, Hennadii Chernyshchyk, Ira Dushar, Oleg Karavaev, Oleh Maksimenko, Yulia Ruda, Valery Teper, and Yury Kreimer. Redundancy AES Masking Basis for Attack Mitigation (RAMBAM). *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2), 2022.

[BBC+19] Gilles Barthe, Sonia Belaïd, Gaëtan Cassiers, Pierre-Alain Fouque, Benjamin Grégoire, and François-Xavier Standaert. Maskverif: Automated verification of higher-order masking in presence of physical defaults. In *European Symposium on Research in Computer Security*, pages 300–318. Springer, 2019.

[BCC+14] Julien Bringer, Claude Carlet, Hervé Chabanne, Sylvain Guilley, and Houssem Maghrebi. Orthogonal direct sum masking: A smartcard friendly computation paradigm in a code, with builtin protection against side-channel and fault attacks. In *IFIP International Workshop on Information Security Theory and Practice*, pages 40–56. Springer, 2014.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *International Conference on Smart Card Research and Advanced Applications*, pages 64–81. Springer, 2014.

[CGLS20] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Transactions on Computers*, 2020.

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of security proofs from one leakage model to another: A new issue. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 69–81. Springer, 2012.

[CO19] Kangqi Chen and Erdal Oruklu. Side-channel attack resilient design of a 10t sram cell in 7nm finfet technology. In *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 860–863. IEEE, 2019.

[CS21] Gaëtan Cassiers and François-Xavier Standaert. Provably secure hardware masking in the transition-and glitch-robust probing model: Better safe than sorry. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 136–158, 2021.

[DCEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware Masking, Revisited. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(2):123–148, 2018.

[DCRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d + 1 shares in hardware. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9813 LNCS:194–212, 2016.

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):547–572, 2018.

[DMRB18] Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. Multiplicative Masking for AES in Hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 431–468, 2018.

[FGDP+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François Xavier Standaert. Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):89–120, 2018.

[GGB+22] Ofek Gur, Tomer Gross, Davide Bellizia, François-Xavier Standaert, and Itamar Levi. An in-depth evaluation of externally amplified coupling (eac) attacks—a concrete threat for masked cryptographic implementations. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 70(2):783–796, 2022.

[GKF18] Robert Giterman, Osnat Keren, and Alexander Fish. A 7t security oriented sram bitcell. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(8):1396–1400, 2018.

[GLZ12] Jin Gong, Wenyi Liu, and Huixin Zhang. Multiple Lookup Table-Based AES Encryption Algorithm Implementation. *Physics Procedia*, 25:842–847, 2012.

[GMK16] Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In *TIS@ CCS*, page 3, 2016.

[GSNP24] Aastha Gupta, Ravi Sindal, Vaibhav Neema, and Ashish Panchal. Secure embedded sram from side channel and data imprinting attacks. In *International Conference on Security, Surveillance and Artificial Intelligence (ICSSAI-2023)*, pages 80–87. CRC Press, 2024.

[GVL+18] Robert Giterman, Maoz Vicentowski, Itamar Levi, Yoav Weizman, Osnat Keren, and Alexander Fish. Leakage power attack-resilient symmetrical 8t sram cell. *IEEE Transactions on very large scale integration (VLSI) systems*, 26(10):2180–2184, 2018.

[HCKG19] Weng-Geng Ho, Kwen-Siong Chong, Tony Tae-Hyoung Kim, and Bah-Hwee Gwee. A secure data-toggling sram for confidential data protection. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(11):4186–4199, 2019.

[HJP+21] Shanshi Huang, Hongwu Jiang, Xiaochen Peng, Wantong Li, and Shimeng Yu. Secure xor-cim engine: Compute-in-memory sram architecture with embedded xor encryption. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(12):2027–2039, 2021.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *Boneh, D. (eds) Advances in Cryptology - CRYPTO 2003. Lecture Notes in Computer Science*, volume 2729, pages 463–481, 2003.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Annual international cryptology conference*, pages 388–397. Springer, Berlin, Heidelberg, 1999.

[KP21] Osnat Keren and Ilia Polian. Ipm-red: combining higher-order masking with robust error detection. *Journal of Cryptographic Engineering*, 11(2):147–160, 2021.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. Silver–statistical independence and leakage verification. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 787–816. Springer, 2020.

[LBS19] Itamar Levi, Davide Bellizia, and François-Xavier Standaert. Reducing a masked implementation's effective security order with setup manipulations and an explanation based on externally-amplified couplings. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):293–317, 2019.

[LGD24] Jianqing Liu, Na Gong, and Hritom Das. Two birds with one stone: Differential privacy by low-power sram memory. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[LK24] Itamar Levi and Osnat Keren. Consolidated Linear Masking (CLM): Generalized Randomized Isomorphic Representations, Powerful Degrees of Freedom and Low(er)-cost. *IACR Cryptol. ePrint Arch.*, Paper 2024/967, 2024.

[LMMS23] Daniel Lammers, Amir Moradi, Nicolai Müller, and Aein Rezaei Shahmirzadi. A Thorough Evaluation of RAMBAM. *CCS 2023 - Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1705–1717, 2023.

[MM22] Nicolai Müller and Amir Moradi. PROLEAD A Probing-Based Hardware Leakage Detection Tool. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):311–348, 2022.

[MMSS19] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-resistant masking revisited. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 256–292, 2019.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M Gammel. Side-channel leakage of masked CMOS gates. In *Cryptographers' Track at the RSA Conference*, pages 351–365. Springer, 2005.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *International workshop on cryptographic hardware and embedded systems*, pages 157–171. Springer, 2005.

[NAN] NANGATE. The NanGate 45nm open cell library. Available online at: http://www.nangate.com.

[Nat01] National Institute of Standards and Technology. FIPS 197-1: Advanced Encryption Standard (AES). 2001.

[NMS23] Syed Farah Naz, Debabrata Mondal, and Ambika Prasad Shah. Side-channel attack resilient rhbd 12t sram cell for secure nuclear environment. *IEEE Transactions on Device and Materials Reliability*, 2023.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4307 LNCS, pages 529–545, 2006.

[OC14] Colin O'Flynn and Zhizhang Chen. ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research. In Emmanuel Prouff, editor, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8622 LNCS, pages 243–260, Cham, 2014. Springer International Publishing.

[RDV12] Vladimir Rožić, Wim Dehaene, and Ingrid Verbauwhede. Design solutions for securing sram cell against power analysis. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 122–127. IEEE, 2012.

[SBHM20] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson. Low-Latency Hardware Masking with Application to AES. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):300–326, 2020.

[SKM+23] Brojogopal Sapui, Jonas Krautter, Mahta Mayahinia, Atousa Jafari, Dennis Gnad, Sergej Meschkov, and Mehdi B Tahoori. Power side-channel attacks and countermeasures on computation-in-memory architectures and technologies. In *2023 IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2023.

[SNS24] Ayan Sharma, Syed Farah Naz, and Ambika Prasad Shah. Secure and reliable single-ended 10t sram cell. In *2024 8th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, pages 1–3. IEEE, 2024.

[Sug19] Takeshi Sugawara. 3-share threshold implementation of AES S-box without fresh randomness. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):123–145, 2019.

[WGC+21] Yoav Weizman, Robert Giterman, Oron Chertkow, Maoz Wicentowski, Itamar Levi, Ilan Sever, Ishai Kehati, Osnat Keren, and Alexander Fish. Low-cost side-channel secure standard 6t-sram-based memory with a 1% area and less than 5% latency and power overheads. *Ieee Access*, 9:91764–91776, 2021.

[WM18] Felix Wegener and Amir Moradi. A first-order SCA resistant AES without fresh randomness. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10815 LNCS:245–262, 2018.

[WMCS20] Weijia Wang, Pierrick Méaux, Gaëtan Cassiers, and François-xavier Standaert. Efficient and Private Computations with Code-Based Masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):128–171, 2020.

[ZWW16] Keji Zhou, Pengjun Wang, and Liang Wen. Design of power balance sram for dpa-resistance. *Journal of Semiconductors*, 37(4):045002, 2016.