IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 2, pp. 69–91. DOI:10.46586/tches.v2022.i2.69-91

Redundancy AES Masking Basis for Attack Mitigation (RAMBAM)

Yaacov Belenky, Vadim Bugaenko, Leonid Azriel, Hennadii Chernyshchyk, Ira Dushar, Oleg Karavaev, Oleh Maksimenko, Yulia Ruda, Valery Teper and Yury Kreimer

FortifyIQ, Inc., 300 Washington Street, Suite 850, Newton, MA 02458 USA
firstname.lastname@fortifyiq.com
https://www.fortifyiq.com/

Abstract. In this work, we present RAMBAM, a novel concept of designing countermeasures against side-channel attacks and the Statistical Ineffective Fault Attack (specifically SIFA-1) on AES that employs redundant representations of finite field elements. From this concept, we derive a family of protected hardware implementations of AES. A fundamental property of RAMBAM is a security parameter d that along with other attributes of the scheme allows for making trade-offs between gate count, maximal frequency, performance, level of robustness to the first and higher-order side-channel attacks, and protection against SIFA-1. We present an analytical model that explains how the scheme reduces the leakage and how the design choices affect it. Furthermore, we demonstrate experimentally how different design choices achieve the required goals. In particular, the compact version exhibits a gate count as low as 12.075 kGE, while maintaining adequate protection. The performance-oriented version provides latency as low as one round per cycle, thus combining protection against SCA and SIFA-1 with high performance which is one of the original design goals of AES. Finally, we assess the leakage of the scheme for the first and the second (bivariate) orders using TVLA methodology on an FPGA implementation and observe resilience to at least 348M traces with 16 Sboxes.

Keywords: Side-channel · DPA · SCA · FIA · AES · Algebraic · Masking · Sbox · Fault injection · SIFA

1 Introduction

As Kocher et al. [KJJ99] have shown in their seminal work, cryptographic secrets, e.g. keys, can be discovered using side-channel attacks that exploit the correlation between intermediate values of the internal state of the cryptographic algorithm and a physical signal, such as power consumption or electromagnetic emanation. Various masking schemes that provide physical security against side-channel attacks by making the signal of intermediate masked values independent of the original values have been suggested [AG01, Tri03, CB08, NSGD12]. Many of these schemes were subsequently shown to be ineffective [MPG05, MPO05, MGH14, CZC+17]. In particular, it was shown that if glitches (multiple transitions of a single gate during a clock cycle) are not taken into account, then masking alone does not suffice, and the secret key can still be discovered.

The suggested approaches to overcome the side-channel leakage caused by glitches can be divided into two categories: *glitch elimination* and *glitch masking*. The former suggests eliminating the very possibility of glitches, typically by employing the dual-rail pre-charge logic technique [TV04, PM05, LMW14, SBHM20]. This technique fits any circuit and combines two basic ideas:

Received: 2021-10-15 Accepted: 2022-12-15 Published: 2022-02-15

1. the dual-rail technique which implements every signal with two complementary wires in order to keep the total power consumption constant;

2. the pre-charge logic technique which adds a pre-charge clock cycle before every functional clock cycle. During every pre-charge clock cycle all wires are brought to a neutral state, so that during both the pre-charge clock cycle and the following functional clock cycle not more than one transition may happen at every wire, meaning no glitches.

The power of the glitch elimination technique lays in its generality. However, it comes at a cost, in particular in the chip area. Great care should be taken during synthesis and place-and-route to achieve sufficient symmetry. While in older glitch elimination schemes [TV04, PM05, LMW14] the latency was at least two clocks per Sbox calculation, in the recent article [SBHM20] a scheme with one clock cycle latency is presented. Still, the gate count remains relatively high even at 100 MHz (see in Table 2 below a comparison against RAMBAM).

The second approach, glitch masking, rather than eliminating the glitches, ensures that they cannot leak any information due to specific requirements on the algorithm used. Glitch masking works at the algorithmic level, and most articles take the AES block cipher as the use case for their research. The data is split into several shares, and the functions used in the algorithm are replaced by functions working with shares that comply with certain limitations, which ensure that no side-channel leakage is possible even in the presence of glitches. We call this approach glitch masking, because the glitches, although existing, do not provide any information useful for side-channel attacks.

One example of glitch masking is Threshold Implementations (TI) [NRR06], in which every elementary function complies with three properties, namely correctness, non-completeness and uniformity. De Cnudde et al. [DCRB+16] presented an efficient protected AES scheme based on the TI paradigm. As shown in [SJR+20], any masking scheme naturally resists SIFA-1 attacks (which assume that faults are injected only in the internal state register) due to the transform property. RAMBAM, being a masking scheme, benefits from this same resistance. However, as we show in Section 3.4.4, RAMBAM protects against simultaneous faults in more bits than in other practical schemes, and this number of bits grows as the redundancy grows. The AES scheme with TI protection presented in [RAD20] is an exception in that it claims protection against SIFA-2 (with no limitation of fault targets), and not only against SIFA-1.

In another glitch masking method, Domain Oriented Masking (DOM) [GMK16], the data is also split into shares, and every share pertains to a *domain*. Logical operations between bits from different domains must be immediately followed by adding fresh randomness. The advantage of DOM is a lower required number of shares than in TI. In the same article an efficient DOM-based AES protected scheme is suggested. De Meyer et al. [DMRB18] took a different approach to glitch masking, suggesting a combination of additive and multiplicative masking, with special attention to the "zero problem" (zero remains zero under any multiplicative masking). The paper also proves the robustness of the suggested scheme against glitches.

Preventing glitch propagation using these algorithmic glitch masking techniques imposes severe limitations on the complexity (e.g. the degree of the polynomial calculated) of the calculation at every clock cycle. As a consequence, the minimal latency is rather high (about 250 clock cycles for AES). In addition, the number of shares (and therefore the gate count, performance, and amount of required random bits) in any glitch masking scheme depends on the maximal order of the attack, against which the scheme is intended to be robust.

Two more papers [Sug19, WM18] enhance the TI approach by totally removing the need for using fresh randomness during the calculations. However they defend only against first order attacks.

It should also be mentioned that, although in the glitch masking approach the absence of leakage can be proven, the proof relies on the assumption that the leakage is additive, namely the leakage of the entire circuit can be calculated as a sum of the leakages of its parts. De Cnudde et al. [CEM18] showed that this assumption does not necessarily hold in reality, and, in fact, for a linear operation performed in two completely independent shares, significant leakage can be measured after tens of millions of traces.

In a completely different research domain, unrelated to security against side-channel attacks, Wu et al. [WHB99] suggested an efficient implementation of the arithmetic of finite fields, in particular of finite fields with characteristic 2. According to this article, the elements of a finite field are represented in a "redundant basis", where the squaring of field elements is a bit permutation, which costs nothing in hardware implementation, and field multiplication is also relatively cheap. The article suggests using this representation for elliptic curve systems for better efficiency (AES had not yet been adopted as a standard in 1999, when the paper was published). If one applies this redundant representation to the field $GF(2^8)$, in terms of which AES is defined, every byte is redundantly represented by 17 bits.

Later, Golić et al. [GT03] independently of [WHB99] suggested using a redundant representation to build a protected multiplicative masking scheme for AES. In the suggested scheme, the calculations are performed in a ring R that includes $GF(2^8)$ as a subring. A homomorphism $H : R \to GF(2^8)$ is defined, and every element of $x \in GF(2^8)$ can be redundantly represented by any element $y \in R$ such that $H(y) = x$. Although the mathematics is described in terms different from [WHB99], in fact the scheme from [GT03] is similar to a particular case of [WHB99]. Namely, the above mentioned redundant representation of the elements of the field $GF(2^8)$ by 17 bits according to [WHB99] can be equivalently expressed as the redundant representation of an element X of $GF(2^8) = GF(2)[x]/(P)$ as $X + CP$, where C is an arbitrary polynomial over $GF(2)$ of a degree less than 9, and multiplication is performed modulo $x^{17} - 1 = (1 + x)PQ$, where $P = 1 + x^3 + x^4 + x^5 + x^8$ and $Q = 1 + x + x^2 + x^4 + x^6 + x^7 + x^8$. In [GT03] the multiplication is performed modulo PQ with the same P and Q; as a result this representation is a 16-bit representation which is more suitable for 16-bit microprocessors than the 17-bit representation according to [WHB99]. The scheme is suggested for practical use in software AES implementations on 16-bit microprocessors. The paper presented a theoretical foundation without a practical implementation. Moreover, based on the analytical results, the authors discover a significant leakage in the suggested scheme. This is likely a result of a lack of re-randomization (see Section 3.2) during the field inversion.

We present a general scheme of AES protection which we called RAMBAM (Redundancy AES Masking Basis for Attack Mitigation) protected against both side-channel attacks and SIFA-1, based on the mathematical foundations similar to [GT03] and [WHB99]. In RAMBAM, every byte of the internal state is replaced before the first round with its randomly chosen preimage in R under a ring homomorphism $H : R \to GF(2^8)$, where R is a ring of characteristic 2. All the AES rounds are performed in R. Finally, the mapping H is applied to the output of the last round. Unlike [GT03] we apply re-randomization (i.e. addition of a random preimage of zero) at every stage of the inversion in the field implemented as raising to the power of 254. This re-randomization is presumably the main reason why our experimentally measured leakage is several orders of magnitude lower than the theoretical leakage reported in [GT03]. RAMBAM was implemented and leakage was measured on an FPGA board.

We attempt neither to eliminate glitches as in the glitch elimination approach, nor to ensure that the information that can be extracted from glitches is totally uncorrelated with the data as in the glitch masking approach. Rather, we introduce a scheme, in which the leakage drops rapidly with the growth of the security parameter d (the number of redundant bits in the byte representation), and already for $d = 8$ we observe no leakage

with up to $3.48 \cdot 10^8$ traces in the 16 Sbox configuration in the emulation on an FPGA board.

To summarize, we make the following contributions in the paper:

- We suggest a protection scheme against power analysis attacks with the following key properties:
  - A security parameter that enables trade-offs between security and gate count
  - Protection against side-channel attacks of any order, with side channel leakage that rapidly decreases as a function of the security parameter
  - Inherent protection against SIFA-1 up to four simultaneous faults in the internal state register
  - A trade-off between gate count and latency, with minimal latency of one round per cycle, a latency not feasible in the glitch masking approach
  - High throughput per gate (for the version optimized for throughput, about 60% better than the previous art)
- We add re-randomization during field inversion that greatly reduces the leakage.
- We study the impact of the choices of P, Q and d on the area, the maximal frequency, and the protection against side-channel and SIFA-1 attacks.
- We study the impact of the choice of the implementation of the Galois field inversion on the area and the maximal frequency.
- We present theoretically and empirically an upper bound of the leakage in RAMBAM, which decreases roughly exponentially, as a function of redundancy. In contrast, in the glitch masking approach leakage exists in reality, although it does not exist in theory.

The remainder of the article is organized as follows. In Section 2, we describe the algorithms. In Section 3, we provide the security considerations behind the choice of the different parameters of RAMBAM. In Section 4, we describe our evaluation of RAMBAM and demonstrate its results. In Section 5, we draw our conclusions.

2 Protected AES Scheme

2.1 Redundant Representation

All bytes in the AES algorithm are interpreted as elements of the Galois field $GF(2^8)$, represented as $F_{P_0} = GF(2)[x]/(P_0)$, namely polynomials over $GF(2)$ modulo P_0. P_0 is a fixed polynomial irreducible over $GF(2)$, defined in the AES standard as:

$$P_0 = x^8 + x^4 + x^3 + x + 1 \tag{1}$$

The bits of a byte $b_7 \dots b_0$ are interpreted as coordinates of a vector in the basis $\langle 1, t_0, \dots, t_0^7 \rangle$, where t_0 is a root of P_0 in $GF(2^8)$. This byte represents the polynomial $\sum_{i=0}^{7} b_i t_0^i$.

Similarly to P_0, any polynomial P out of the 30 irreducible polynomials of degree 8 over $GF(2)$ can be used to construct the field $GF(2^8)$, represented as $F_P = GF(2)[x]/(P)$. Using a linear transformation L, it is possible to switch from the standard P_0-based representation to an alternative (P-based) representation in the basis $\langle 1, t, \dots, t^7 \rangle$, where t is a root of P.

In RAMBAM, in order to randomize the representation, we use a larger algebraic structure — the ring $R_{P\cdot Q} = GF(2)[x]/(P \cdot Q)$, where Q is a polynomial (not necessarily irreducible) of degree $d > 0$ over $GF(2)$. We will denote $P \cdot Q$ as Z. Since $P \cdot Q$ is clearly reducible, $R_{P\cdot Q}$ is a commutative ring, but not a field. The elements of $R_{P\cdot Q}$ are identified with polynomials over $GF(2)$ modulo $P \cdot Q$ of the degree at most $8 + d$, and are represented by $(8 + d)$-bit words. A ring homomorphism $H : R_{P\cdot Q} \to F_P$ is defined as $H(X) = X \bmod P$. Note that the set of all polynomials in $R_{P\cdot Q}$ of a degree less than 8 is a subring of $R_{P\cdot Q}$ isomorphic to F_P, and all these polynomials are fixed points of H. We will identify this subring with F_P.

The polynomials P and Q are parameters of each RAMBAM variant. We call the degree d of the polynomial Q its *redundancy*, because it reflects the number of additional (redundant) bits in the representation of every byte.

Formalizing the above, we give the following definition.

Definition 1. Let d be a positive integer, P, Q — polynomials over $GF(2)$ of orders 8 and d, respectively, where P is irreducible. Let B be a byte interpreted per the AES standard as $\sum b_i x^i$ in the field $GF(2^8)$ represented as $GF(2)[x]/(P_0)$, where P_0 is defined by (1). Let t be a byte in the same representation such that $P(t) = 0$, and let L be the linear transformation that maps the standard AES representation of any byte to the representation of the same byte in the basis $\langle 1, t, \ldots, t^7\rangle$. Let B^* be a $(8 + d)$-bit value interpreted as the element $\sum b_i^* x^i$ of the ring $GF(2)[x]/(P \cdot Q)$. B^* is said to be a *redundant representation* of B if $B^* \bmod P = L(B)$.

Note that any value B has 2^d different interchangeable redundant representations. Therefore it is possible to replace any one of them with another at any moment. This property lays the foundation for the re-randomization (see Section 3.2).

The two independent parameters L (or, equivalently, t) and Q uniquely define a redundant representation. The value d is the degree of Q, and P is the minimal polynomial of t.

2.2 Top-level Algorithm

In this article, we demonstrate an implementation of RAMBAM for AES-128 encryption. AES encryption with other key sizes and AES decryption with any key size can be implemented in a similar way.

Algorithm 1 is the top-level RAMBAM algorithm for protected AES 128 encryption with fixed P, d, Q, L (where addition and multiplication are assumed to be the ring operations). The algorithm has the same general structure as the standard AES algorithm, with the following differences:

1. Before the first round, the bytes of both the key and the input data are transformed from the representation in F_{P_0} to the representation in $F_P \subset R_{P\cdot Q}$. Additionally, every byte of the input data is randomized by adding P multiplied by a random polynomial of a degree less than d. After that every byte of the internal state belongs to $R_{P\cdot Q}$ and is represented by $8 + d$ bits. Alternatively, instead of randomizing the input data, it is possible to randomize the key. On the other hand, the advantage of randomizing only the input data rather than only the key is that in this case the area necessary for the key management may be saved, e.g., by using the compact Sbox implementation by Canright [CB08]

2. $AddRoundKey_d$ and $ShiftRows_d$ are similar to the standard transformations with

the same names, except that every byte is represented by $8 + d$ bits.

Algorithm 1: Protected AES128 Encryption

Function $ProtectedAesEnc_{P,Q,d,L}(key\_in[16], x\_in[16], r[23])$[a] [b]
  Input : $key\_in[16]$ — a 16-byte key; $x\_in[16]$ — a 16-byte input data; $r[23]$ — 23 random d-bit values
  Output : $x\_out[16]$ — a 16-byte output
  /* Switch to the redundant representation -- change the basis and randomize the input data */
  for $i = 0$ **to** 15 **do**
    $key_i = L(key\_in_i)$
    $x_i = L(x\_in_i) + r_i P$
  end for
  /* Perform 10 AES rounds in the redundant representation */
  for $roundIndex = 0$ **to** 9 **do**
    $x = AddRoundKey_d(x, key)$
    $key = ProtectedNextRoundKey(key, roundIndex)$
    $x = ShiftRows_d(x)$
    $x = ProtectedSubBytes_{P,Q,d}(x, r)$ /* see Algorithm 2 below */
    if $r \neq 9$ **then**
      $x = ProtectedMixColumns_{P,Q,d}(x)$
    end if
  end for
  $x = AddRoundKey_d(x, key)$
  /* Derandomize and switch the basis */
  for $i = 0$ **to** 15 **do**
    $x\_out_i = L^{-1}(x_i \bmod P)$
  end for
  return x_out
end

[a] The parameters P, d, Q, L are fixed for a RAMBAM variant.
[b] Here and in all the sub-algorithms, addition and multiplication are ring operations.

3. In $ProtectedMixColumns_{P,Q,d}$ multiplication by 2 and by 3 is replaced with multiplication by $L(2)$ and by $L(3)$, respectively. Since for any given instantiation of the protected AES scheme these values are constant, operations of multiplication by them are implemented as fixed linear functions on bits.

4. $ProtectedSubBytes_{P,Q,d}$ is the only non-linear operation. Its algorithm is described in Section 2.3.

5. $ProtectedNextRoundKey_{P,Q,d}$ is the same as $NextRoundKey$, except that $L(rcon_r)$ is used instead of $rcon_r$, and $ProtectedSubBytes$ is used instead of $SubBytes$.

2.3 Implementation of $ProtectedSubBytes_{P,Q,d}$

In the AES standard, every byte undergoes the transformation $Aff \circ Inv$, where Inv is the inversion in F_{P_0}, defined at 0 as $Inv(0) = 0$, and Aff is a fixed affine transformation. Since the multiplicative group of F_{P_0} has order 255, we can equivalently define $Inv(x) = x^{254}$.

In a ring of characteristic 2, raising to a power of 2^n is a linear transformation. Since linear transformations are much cheaper in HW than ring multiplication, for implementing raising to the power of 254 as a sequence of multiplications and raisings to powers of 2^n,

we search for a sequence optimal according to the following criteria, in the descending order of importance:

1. Minimal number of multiplications (in order to minimize the area);
2. Minimal number of raisings to powers of 2^i (in order to minimize the area);
3. Minimal depth of the longest path (in order to increase the maximal frequency).

In Appendix A we prove that the number of multiplications in such a sequence cannot be less than 4 (and more generally, for inversion by raising to the power of $2^{2^n}-2$ in $GF(2^{2^n})$, where $n>2$, at least $n+1$ multiplications are required).

A brute-force search for all possible sequences with exactly 4 multiplications shows that the requirements 2 and 3 are mutually exclusive, so there are two different versions of $ProtectedSubBytes$, one optimized for the area, the other optimized for maximal frequency. Both of them are not unique.

In both versions, in order to minimize the side-channel leakage, we re-randomize the result of every multiplication and every raising to a power by adding a random polynomial divisible by P. See below in Section 3.2 the rationale behind the re-randomization.

Algorithm 2 implements $ProtectedSubBytes_{P.Q,d}$ optimized for the minimal area. Algorithm 5 in Appendix B is an alternative version of the same function, optimized for the maximal frequency; in this version, the size of the array r must be 24 rather than 23. In both versions, the operations that can be performed in parallel are shown as an indented block in curled brackets.

Algorithm 2: ProtectedSubBytes (optimized for the area)

Function $ProtectedSubBytes_{P.Q,d}(x\_in[16], r[23])$
  Input : $x\_in[16]$ — 16 $(8+d)$-bit values, representing the AES state bytes
    $r[23]$ — 23 random d-bit values (only the last 7 are used here)
  Output : $x\_out[16]$ — 16 $(8+d)$-bit values after the Sbox transformation
  for $i=0$ **to** 15 **do**
    $t = x\_in_i$
    $t2 = Pow2_{P.Q,d}(t) + r_{16}P$
    $t3 = Mul_{P.Q,d}(t, t2) + r_{17}P$
    $t12 = Pow4_{P.Q,d}(t3) + r_{18}P$
    { `/* parallel section */`
      $t14 = Mul_{P.Q,d}(t2, t12) + r_{19}P$
      $t15 = Mul_{P.Q,d}(t3, t12) + r_{20}P$
    }
    $t240 = Pow16_{P.Q,d}(t15) + r_{21}P$
    $t254 = Mul_{P.Q,d}(t14, t240) + r_{22}P$
    $x\_out_i = RAff_{P.Q,d}(t254)$
    Rotate $r[16\ldots22]$ by one position
  end for
  return x_out
end

In these algorithms, $Pow2, Pow4, Pow16, Pow64$ - hardwired linear transformations for raising to the powers of 2,4,16,64 in $R_{P.Q}$, respectively, Mul - multiplication in $R_{P.Q}$, and $RAff$ - one of many linear transformations in $R_{P.Q}$ such that $H(RAff(x) = Aff(H(x))$, where Aff - the affine transformation defined in the AES standard.

The multiplications (Mul) are performed using the schoolbook multiplication algorithm with reduction modulo $Z = P \cdot Q$, as described in Algorithm 3. The actual modular reduction is encapsulated in the function $ModularShiftLeft_{Z,d}$ (Algorithm 4).

Algorithm 3: Multiplication

Function $Mul_{Z,d}(a\_in, b\_in)$
  Input : a_in, b_in — two $(8+d)$-bit values, each one representing a byte
  Output : c_out — a $(8+d)$-bit value representing $a\_in \cdot b\_in \bmod P$
  $c\_out = 0$
  $deg = b\_in$
  for $i = 0$ **to** $7 + d$ **do**
    if $((ai >> i)\&1) = 1$ **then** $c\_out = c\_out + deg$
    $deg = ModularShiftLeft_{Z,d}(deg)$
  end for
  return c_out
end

Algorithm 4: ModularShiftLeft

Function $ModularShiftLeft_{Z,d}(deg)$
  Input : x_in — a $(8+d)$-bit value
  Output : x_out — a $(8+d)$-bit value
  $x\_out = x\_in << 1$
  if $((x\_out >> (8+d))\&1) = 1$ **then**
    $x\_out = x\_out + Z$
  end if
  return x_out
end

3 Security Considerations and Optimizations

In this section, we sketch an analytical basis that provides an intuition behind the algorithms outlined in Section 2. Furthermore, we discuss the degrees of freedom within these algorithms, the existing options and their impact on the security and on the hardware implementation.

3.1 Assessment of the Leakage as a Function of Redundancy

First, we assess the leakage of a weakened version of Algorithm 1, in which $r[i] = 0$, $16 \leq i < 23$ — namely, the input data is randomized, but no re-randomization is applied in the function *ProtectedSubBytes* (Algorithm 2). To assess the leakage, we develop a generic model that derives the leakage from the statistical properties of the intermediate results in the algorithm. This model mimics the hardware behavior by using the algorithm's intermediate results to represent the values of the combinational logic signals in hardware. However, the model is not tied to any specific realization.

The function *ProtectedSubBytes* is essentially a loop over the redundant representations x_in_i of the internal state bytes, where the intermediate results of each loop iteration i depend solely on the value of x_in_i. These intermediate results include the intermediate and the final results of the four invocations of the function Mul (Algorithm 3). We assess the leakage of the clear value of the input bytes through the average Hamming weight of the intermediate results in the following way.

1. Build a "trace" of the loop iteration of *ProtectedSubBytes* for a specific value of the input x_in_i to this loop, with a specific redundancy d and specific polynomials P and Q. The trace consists of the values of the variable c_out of the function

Mul at all $8+d$ iterations of the loop of Mul in all 4 invocations of Mul from $ProtectedSubBytes$ (total of $4(8+d)$ values).

2. Calculate the Hamming weights of these $4(8+d)$ values. The result is a vector of the Hamming weights of the intermediate results.
3. Repeat step 2 for all 2^d redundant representations of a specific clear byte value. The result is a matrix of the Hamming weights of the intermediate results with $4(8+d)$ columns and 2^d rows.
4. Calculate the column averages. The result is a vector of the $4(8+d)$ averaged Hamming weights of the intermediate results.
5. Repeat steps 3-4 for all 256 clear byte values. The result is a matrix of the averaged Hamming weights of the intermediate results with $4(8+d)$ columns and 256 rows.
6. For every column, calculate the maximal deviation (an absolute value) from the column average. The result is a vector of the $4(8+d)$ maximal deviations.
7. Take the largest of these maximal deviations and divide it by $8+d$ ($8+d$ is the number of bits in the redundant representation of each byte). The result is a single number L. This number characterizes the leakage per bit for specific polynomials P, Q.
8. Calculate the characteristic of robustness $R = 1/L^2$, which is expected to be proportional to the average number of traces at which the leakage becomes detectable (i.e., the absolute value of the t-test is greater than 4.5). (We added this step to ease the comparison of this theoretical prediction with experimental data for the number of traces at which the t-test reaches 4.5. Random noise decreases proportionally to the square root of the number of traces, so for a constant signal the number of traces at which the signal-to-noise ratio reaches a predefined level is inversely proportional to the square of the noise level.)
9. Repeat steps 5-8 for all 30 irreducible polynomials P of degree 8, and all 2^{d-1} polynomials of degree d with a non-zero free term, for a specific value of the redundancy d. The result is a matrix of the robustness characteristics for all polynomial pairs P, Q for redundancy d.
10. Repeat step 9 for all the redundancies from 3 to 8.

From this calculation, we learn that the expected leakage significantly depends not only on the redundancy, but also on the selection of the polynomials P, Q. For every value of the redundancy we sort the polynomial pairs according to their robustness characteristics. Comparing the best and the worst pairs for different values of the redundancy we learn that, while the robustness characteristic of the worst polynomial pairs is almost independent of the redundancy, the robustness characteristic of the best polynomial pairs increases roughly exponentially as a function of the redundancy, as shown in Figure 1.

3.2 Re-randomization

The leakage assessment above is based on the fact that in the course of the $ProtectedSbox$ calculation, applied to one of 2^d redundant representations of any given byte value, for every application of the function Mul only 2^d pairs of multiplicand values are possible, because this pair of values is determined by the input to $ProtectedSbox$. In order to make all 2^{2d} (instead of 2^d) pairs possible, and thus overcome this source of leakage we re-randomize each intermediate result by adding a random polynomial divisible by P.

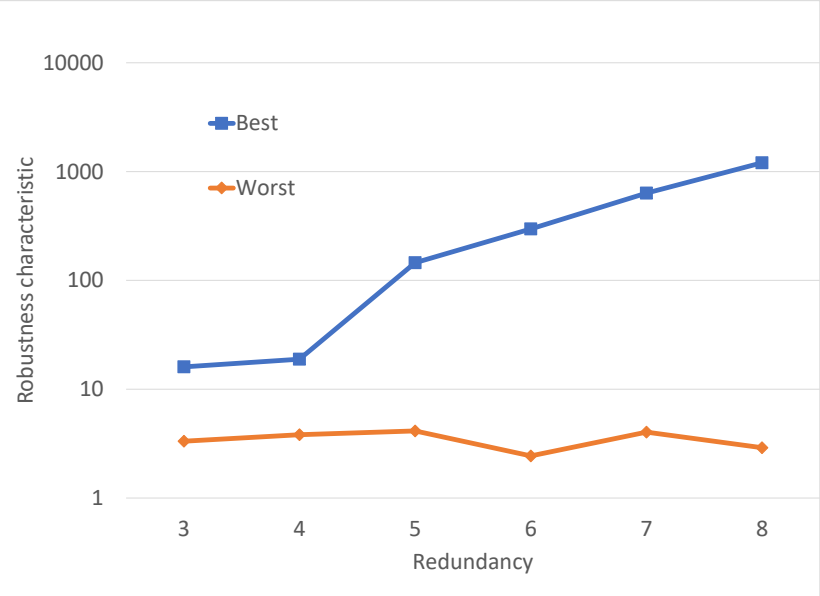


Figure 1: Best and worst robustness characteristics without re-randomization as a function of redundancy

Indeed, as demonstrated in Section 4.3, the leakage drops drastically. The paper by Golić et al. [GT03] lacks this step, and we believe it explains a significant difference between our experimental results and the theoretical leakage calculated in [GT03].

3.3 Reuse of the Random Bits

The number of bits necessary for the re-randomization of one Sbox calculation is kd, where k is the number of transformations (exponentiations and multiplications) in the function $ProtectedSubBytes$ (7 for Algorithm 2, 8 for Algorithm 5), and d is the redundancy. We experimentally found by performing TVLA on an FPGA board (see Section 4.2) that reusing the same random bits over all the bytes of the internal state and over all the AES rounds does not affect the leakage, if the same re-randomization addend is not used by the same logic at consecutive clock cycles. To overcome this problem, we rotate the set of values used for the re-randomization by one position at each cycle, so that the re-randomization applied at the same gates at consecutive clock cycles uses different addends. Therefore, we suggest such a reuse, with the total number of random bits per AES encryption being $(16 + k)d$, $16d$ for the initial key randomization and kd for re-randomization in all the $ProtectedSubBytes$ calculations. However when comparing the amount of randomness in RAMBAM against other schemes in Table 2 and Table 3, we do not take this reuse into account.

3.4 Choice of the Polynomials P and Q

Several criteria guide the selection of P and Q:

1. Q should not be divisible by P.
2. Q should be irreducible.
3. The Hamming Weight of the product $P \cdot Q$ should be as low as possible.

4. The product $P \cdot Q$ should possess specific properties to maximize protection against SIFA-1.

These criteria are discussed in detail in the following subsections.

3.4.1 Q not Divisible by P

If Q is divisible by P, say $Q = P \cdot R$, where $deg(R) = d - 8$, then raising to the power of 2^n modulo $Z = P \cdot Q = P^2 \cdot R$ severely adversely affects the uniformity. If we raise all 2^d representations $X + C \cdot P$ of a byte X to the power of 2^n modulo $Z = P^2 \cdot R$, where $deg(C) < d$, then by linearity we have $(X + C \cdot P)^{2^n} = X^{2^n} + C^{2^n} \cdot P^{2^n} = X^{2^n} + P^2 \cdot C^{2^n} \cdot P^{2^n-1} = X^{2^n} \bmod P^2$. There are only 2^{d-8} different values modulo $Z = P^2 \cdot R$ that give the same remainder modulo R, so the number of possible values decreases by a factor of at least 256. Hence, it is a bad idea to use Q divisible by P.

3.4.2 Irreducible Q

Unlike P which must be irreducible (otherwise $GF(2)[x]/(P)$ is not a field and cannot be isomorphic to $GF(2^8)$), the scheme works correctly with any Q, either reducible or irreducible. However we prefer using an irreducible Q for the following reasons. We assume that Q is not divisible by P for the reason explained above — and since P is irreducible, it means that P and Q are relatively prime. Therefore the Chinese Remainder Theorem is applicable to $X = P \cdot Q$, and we can analyze multiplication modulo Z separately modulo P and modulo Q.

Let X be an arbitrary element of $GF(2^8)$. It has 2^d redundant representations $X + C \cdot P$ modulo Z, all of them equal to x modulo P. On the other hand, since P and Q are relatively prime, P is invertible modulo Q, and therefore they all differ modulo Q and cover all 2^d existing values. For this reason, taking two elements X and Y of $GF(2^8)$, the frequencies of the values modulo Q of the product of their redundant representations are distributed the same way as the values are distributed in the multiplication table modulo Q. Thus multiplication table has 2^{2d} entries. If Q is irreducible, then 0 will appear in $2^{d+1} - 1$ entries (only where at least one of the multiplicands is 0), and any other value will appear in $2^d - 1$ entries. If Q is reducible, then it has non-trivial divisors, so the table will include more zero entries, and will be farther from uniformity than in the case of an irreducible Q. For this reason we prefer irreducible values of Q.

3.4.3 Small Hamming Weight of $P \cdot Q$

The function $ModularShiftLeft_{Z,d}$ ("doubling"), which is called $8 + d$ times by the function $Mul_{Z,d}$, involves a conditional addition (XOR) with $Z = P \cdot Q$. The less '1' bits there are in Z, i.e. the lower the Hamming weight of Z is, the cheaper the doubling is. For this reason we prefer such pairs $\langle P, Q \rangle$ that give $P \cdot Q$ with a low Hamming weight.

Suppose P and Q are irreducible and $d > 1$ ($d = 1$ does not give reasonable protection anyway). Then:

- $HW(Z) = 1$ is impossible, otherwise Z is divisible by the polynomial x of degree 1;
- $HW(Z)$ cannot be even, in particular $HW(Z) = 2$ is impossible, otherwise Z is divisible by the polynomial $x + 1$ of degree 1;

Therefore the minimal possible value of $HW(Z)$ is 3.

A brute force search shows that up to $d = 11$ the only values of d for which there exists an irreducible polynomial P of degree 8 and a polynomial Q of degree d such that $HW(P \cdot Q) = 3$ are 3, 5 and 8. Moreover, for all these values of d an irreducible Q can be chosen. For all other values of d, P and Q can be chosen so that $HW(Z) = 5$, which gives a slightly higher gate count relative to d.

3.4.4 SIFA-1 Related Properties of $P \cdot Q$

The SIFA attack [DEK+18] assumes a fault in several bits of an intermediate state such that:

- Ineffective faults happen with a non-negligible probability (unlike bit-flip, for which any fault affects the state);
- The distribution of the states for which the fault is ineffective is non-uniform.

These assumptions are rather weak, and that is what makes this attack so powerful.

Let's study the applicability of SIFA-1 (a variant of SIFA, which assumes that faults are limited to the internal state register) to RAMBAM. We assume the RAMBAM variant implemented in the target device is known to the attacker, and he knows how to derive the clear value $H(X)$ from a redundant value X. Let's suppose that using a SIFA-1 attack the attacker manages to find non-uniformity in the distribution of a byte of an intermediate internal state. Such an attack must target at least some subset of the $8 + d$ bits of the register containing a redundant representation of this byte. Suppose this subset consists of the bits with indices $k_0, \ldots, k_{n-1}$, where $n \leq 8 + d$. Recall that 2^d redundant representations of a byte value X are given by the formula $X + C \cdot P$, where

$$C = \sum_{j=0}^{d-1} c_j x^j$$

is one of 2^d polynomials of a degree less than d. For the bit with index k_i of $X' = X + C \cdot P$, i.e. the coefficient of t^{k_i} in X', we can write

$$X'_{k_j} = X_{k_j} + \sum_{i=0}^{d-1} c_i p_{k_j - i}$$

Here we assume $p_m = 0$ if $m < 0$ or $m > 8$.

If we interpret $X'_{k_0}, \ldots, X'_{k_{n-1}}$ as a vector Y in F^n, and $c_0, \ldots, c_{d-1}$ as a vector C in F^d (identifying the polynomial C with the vector of its coefficients), then we can write

$$Y = Y^0 + M \cdot C; \text{ where } Y^0 = \langle X_{k_0}, \ldots, X_{k_{n-1}} \rangle$$

and the elements of the $d \times n$ matrix M are defined as

$$M_{ij} = p_{k_j - i} \tag{2}$$

It is easy to see that all 2^n values of $M \cdot C$ are distributed uniformly, assuming uniform distribution of C, if and only if $rank(M) = n$. In this case the values of the vector $Y = Y^0 + M \cdot C$ are distributed uniformly regardless of the value of Y^0, or equivalently regardless of the value of X. It means that if $rank(M) = n$ then a SIFA-1 attack targeting n bits of the redundant state $X'_{k_0}, \ldots, X'_{k_{n-1}}$ cannot be successful, since ineffectiveness of a fault depends of the values of bits $X'_{k_0}, \ldots, X'_{k_{n-1}}$, and those are independent of the value of X.

For every irreducible polynomial P of degree 8 and for every redundancy d there exists n_{max} such that for any $n \leq n_{max}$ group of bits $(k_0, \ldots, k_{n-1})$ the matrix M defined by (2) has rank n — meaning that no successful SIFA-1 attack on a target containing up to n_{max} bits is possible.

Table 4 in Appendix C shows n_{max} as a function of P and d, for d between 1 and 20, and the maximal value of n_{max} over all the polynomials P for every d. In particular, for $d = 8$ the best value is 4, meaning that for specific polynomials P ($0x139$ and $0x1d7$) no

SIFA-1 attack on 4 or less bits can be successful. For $d = 9$ with the same polynomials no SIFA-1 attack on 5 or less bits can be successful.

In comparison, in the glitch elimination scheme [LMW14] two bits (m_x and one of the two bits representing x_m) give full information about the clear bit x, therefore a SIFA-1 attack targeting these two bits may succeed. In glitch masking schemes (TI and DOM) protecting against side-channel attacks of an order r, every bit is represented by $r + 1$ bits. Therefore, for an implementation protecting against second order attacks, a SIFA-1 attack against it that targets 3 bits may succeed.

4 Experimental Results

4.1 Setup

To evaluate the scheme, we produced several RTL implementations, each one for a particular RAMBAM variant with its own values of the redundancy d and the polynomials P and Q. For area and performance comparison, we also added the parameter of the number of Sboxes, creating two variants: a compact variant with a single Sbox and a faster variant with 16 redundant Sboxes for $ProtectedSubBytes$ and 4 compact Canright Sboxes [Can05] for round key calculations. We synthesized the RTLs for the following two target platforms:

1. For area and performance evaluation: an ASIC netlist obtained using the Yosys synthesizer [Wol16] and the NanGate FreePDK45 Open Cell Library [Nan08].

2. For security evaluation: The `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 cycle. The power signal was obtained by measuring the current via a shunt resistor connected serially to the FPGA supply line.

For every variant we acquired two sets of traces with constant key. The input data was constant for one set, and pseudorandom for the second set. The number of acquired traces varied depending on the redundancy. In order to eliminate false positives caused by environmental factors, the two settings were run in an interspersed manner, switching between constant input and random input with every encryption operation.

4.2 Security Evaluation

To evaluate the robustness of RAMBAM against SCA, we used the TVLA methodology [GJJR11], in accordance with which we carried out two experiments with the same AES key — one with a constant input and the other with a pseudo-random input. We performed TVLA for several choices of the parameters, with the redundancy values ranging from 3 to 8. The algorithm used was Algorithm 1, which assumes the reuse of the random input.

For this evaluation, we used the low-area implementation with one instance of the Sbox, which performs one round in 20 clock cycles. The entire AES calculation takes 233 clock cycles (16 clock cycles for internal state loading, $20 \cdot 10 = 200$ clock cycles for 10 AES rounds, 1 clock cycle for the last $AddRoundKey$ and de-randomization, and 16 clock cycles for internal state output).

After 217 clock cycles, it is already possible to feed the next input, in parallel with receiving the previous output. The presence of the leakage, revealed by TVLA, is a necessary, but not a sufficient condition for an attack. In particular, any appearance of clear text in the system, even at the input of the cipher, will exhibit leakage, albeit not prone to attacks. To avert this phenomenon, we used a modified version of the design, in

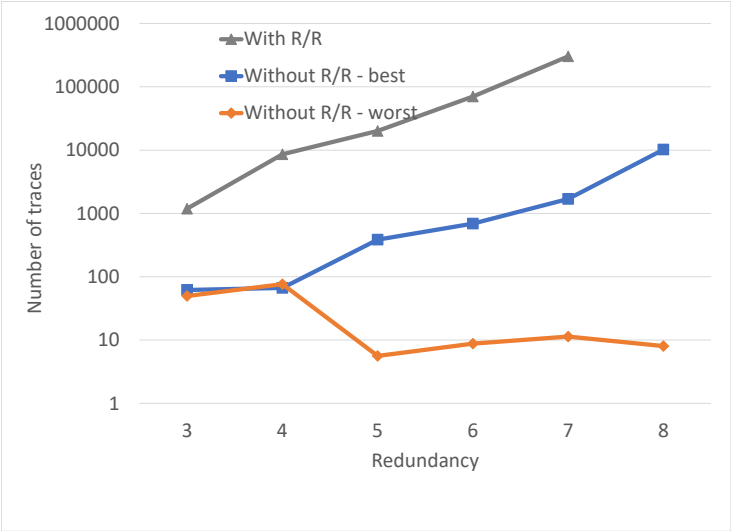


Figure 2: Number of traces at which the t-test reaches 4.5 by absolute value (without re-randomization — worst and best, and with re-randomization)

which the initial randomization and the final de-randomization of the state are omitted. Thus, all the data involved in the calculations were kept in the redundant representation. This approach is in line with the related work [DCRB+16, GMK16, DMRB18]. A full AES calculation, which takes 203 clock cycles (in this setting, we spend only 2 instead of 32 clock cycles for input/output handling, as it is not essential for the TVLA assessment), was measured. We performed first order and bivariate second order TVLA analysis.

Table 1: Polynomial pairs used for the first order TVLA

Redundancy	Without re-randomization				With re-randomization	
	Worst		Best			
	P	Q	P	Q	P	Q
3	0x1dd	0xd	0x169	0x9	0x1dd	0xd
4	0x163	0x1f	0x163	0x17	0x163	0x1f
5	0x1dd	0x33	0x1a9	0x3b	0x1a9	0x3b
6	0x1f9	0x45	0x11b	0x47	0x13f	0x43
7	0x1f5	0xff	0x187	0xfb	0x11b	0x89
8	0x1a3	0x101	0x169	0x17b	0x169	0x17b

The results of the first order TVLA as a function of the redundancy are shown in Figure 2. There are three graphs:

- without re-randomization with the worst pairs of polynomials;
- without re-randomization with the best pairs of polynomials;
- with re-randomization.

The pairs of polynomials corresponding to each point on the graphs in both Figure 1 and Figure 2 are listed in Table 1. For every graph and every redundancy, the estimation of the average number of traces at which leakage starts, is shown. The point for redundancy 8 with re-randomization is intentionally missing, because we have not reached the point at which leakage starts. This graph supports the theory in Section 3.1 in the following ways:

1. A weak dependency of the leakage on the redundancy for the worst polynomials without re-randomization;
2. An approximately exponential dependence of the leakage on the redundancy for the best polynomials without re-randomization;
3. A significant difference in the leakage between the versions with and without re-randomization.

In addition, Figure 3 shows the maximal absolute value of the t-test as a function of the number of traces up to 1M, for different redundancies (1 Sbox). For all the redundancies the t-test reaches 4.5 well before 1M traces — except for redundancy 8, for which we acquired 2.7M traces and found no leakage (not shown in Figure 3).

Figure 4 shows the maximal absolute value of the t-test as a function of the number of traces up to 348M, for redundancy 8 (16 Sboxes). The difference in the number of traces between the two versions is due to the different trace sizes.

The results of the second order bivariate TVLA for redundancy 8 with polynomials $P = \texttt{0x169}, Q = \texttt{0x17b}$ with 1.188M traces (1 Sbox) and 348M traces (16 Sboxes) are presented in Figure 5. In both cases the t-test values never exceed 5.

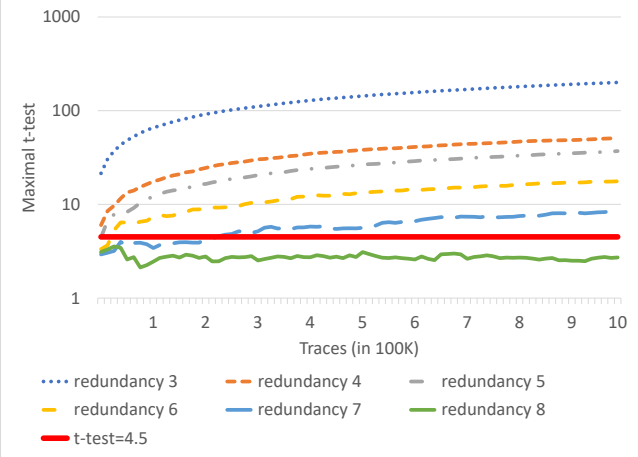


Figure 3: Maximal t-test as a function of the number of traces for different redundancy values (1 Sbox with re-randomization)

4.3 Area and Performance Evaluation

In Table 2, we compare the implementation cost and the effectiveness of RAMBAM (redundancy 8 with 16 protected Sboxes and 4 Canright [Can05] Sboxes for the key expansion) with the implementation of Low-Latency Hardware Masking [SBHM20]. Since in [SBHM20] only latency per round and gate count per Sbox block are provided, the data for latency and throughput for both schemes is per round.

In Table 3, we compare the implementation cost and the effectiveness of RAMBAM (redundancy 8 with one Sbox) with the first order secure and second order secure versions of other schemes with one Sbox — Multiplicative Masking [DMRB18], Masking with $d+1$ shares [DCRB+16], Domain-oriented masking [GMK16], and two schemes which use no fresh randomness [WM18, Sug19].

Note that the advantage in throughput per gate of RAMBAM over [DMRB18] would significantly grow if two Sboxes are used in both schemes. The reason is that, while the gate count in both schemes would grow in roughly the same proportion, the number of

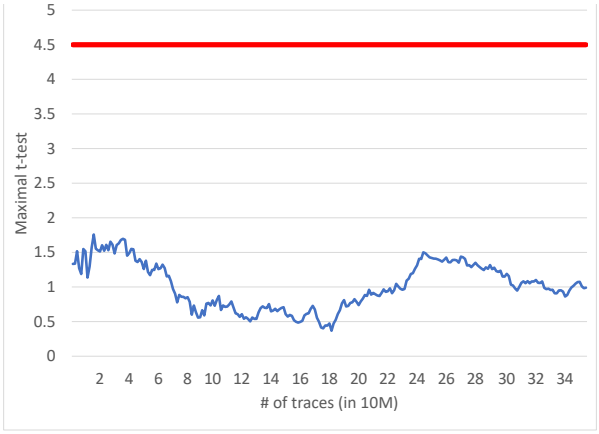


Figure 4: Maximal t-test vs. number of traces for redundancy 8 (16 Sboxes with re-randomization)

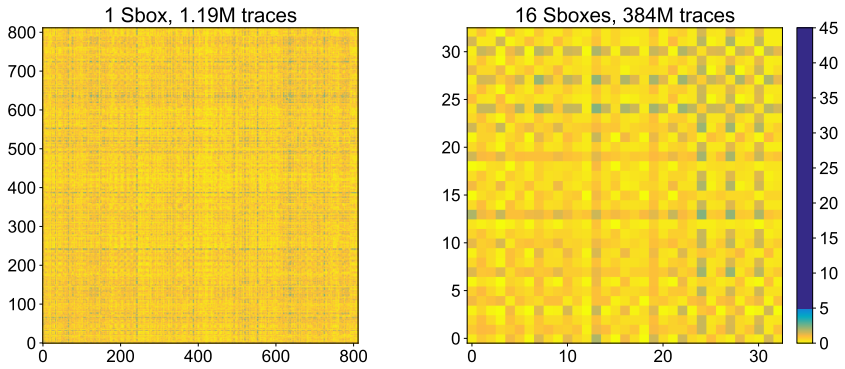
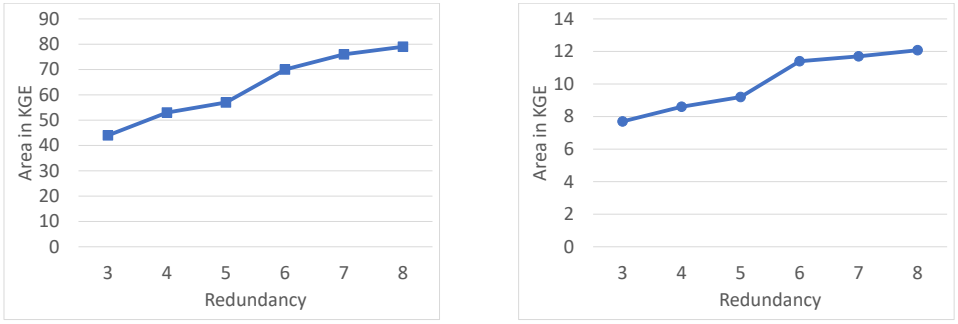


Figure 5: Bivariate second order TVLA results (redundancy 8 with re-randomization)

clock cycles per round would change from 16 to 8 for RAMBAM and from 21=5+16 to 13=5+8 for [DMRB18]. For implementations with more Sboxes, the advantage of RAMBAM over [DMRB18] would grow even more.

Note also that the comparison of the gate count favors the alternative schemes against RAMBAM, because the gate count for RAMBAM includes randomization and de-randomization, while the gate count for the other schemes does not include the logic used to implement the splitting into shares and the recombination of the shares.

Although we reuse the random bits, we do not take this reuse into account in this comparison. In the single Sbox version, the amount of randomness per Sbox is similar to the best previous solutions — except for [WM18, Sug19] where no fresh randomness at all is used, but the protection is against first order attacks only. In the 16 Sboxes version the amount of randomness in [SBHM20] is lower than in RAMBAM, but the area is significantly higher even at 100 M.



(a) Fast Version (16 Protected Sboxes) **(b)** Small Version (one Sbox)

Figure 6: Area as a Function of the Redundancy

Table 2: AES-128 realization area and performance comparison for schemes with 16 Sboxes

Scheme	Area [kGE]	Randomness [bits/Sbox]	Round latency [cycles]	Throughput per gate per round [bit/kGE]
RAMBAM ($d = 8$)	78.9	56[1]	1	1.623
[SBHM20]	123.1	36	1	1.040

Table 3: AES-128 realization area and performance comparison for schemes with 1 Sbox

Order	Scheme	Area [kGE]	Randomness [bits/Sbox]	AES latency [cycles]	Throughput per gate (full AES) [bit/kGE]
Second	RAMBAM ($d = 8$)	12.1	56[1]	233	0.0488[3]
	[DMRB18][2]	10.9[4]	53[4]	256[4]	0.0457
	[DCRB+16][2]	12.6[4]	162[4]	276[4]	0.0367
	[GMK16][2]	12.0[4]	54[4]	246[4]	0.0433
First	[DMRB18][2]	6.6[4]	19[4]	256[4]	0.0762
	[DCRB+16][2]	7.7[4]	54[4]	276[4]	0.0603
	[GMK16][2]	7.3[4]	18[4]	246[4]	0.0709
	[WM18][2]	7.6[5]	0	2804[5]	0.0060
	[Sug19][2]	17.1[5]	0	266[5]	0.0281

[1] Although all the experiments shown above were performed while re-using the same 56 random bits during the entire AES calculation (rather than 56 bits $\times$ 160 Sboxes $\approx$ 9K bits), in this comparison we do not take this re-use into account

[2] The area of the alternative schemes does not include the logic used to implement the splitting into shares and the recombination of the shares

[3] Although the latency is 233 clock cycles, after 217 cycles, it is already possible to start feeding the next input, in parallel with receiving the previous output. For this reason, the throughput calculation is based on 217 clock cycles

[4] Based on [DMRB18, Table 6]

[5] Based on [Sug19, Table 1]

5 Conclusions

In this paper, we proposed RAMBAM, a conceptually new algebraic masking, designed to protect against side-channel attacks and against SIFA-1. The masking mechanism employs redundant representations in the Galois field and adds re-randomization as a key element to keep uniform distribution of the masked state.

A key property of the proposed masking is its flexibility, which leads to a wide variety of configurations. One of the key parameters that allows for this flexibility is the security parameter d that denotes the redundancy. The analytical model, presented in the paper, explains the effect of this parameter and the choice of the polynomials on the leakage.

Our experimental results based on the TVLA methodology and obtained from the FPGA demonstrated that the leakage quickly (at least exponentially) decreases as a function of the redundancy, so that with $d = 8$ no leakage could be observed with at least up to 348M traces in the 16 Sboxes version and at least up to 2.7M traces in the 1 Sbox version, where the acquisition was limited by the experiment's setup.

While the article presented the results for AES-128 encryption in comparison to previous work, other key sizes and decryption can be easily included. When optimized for performance, RAMBAM provides much lower latency than all previous solutions except for [SBHM20] — which on the other hand requires a significantly higher area. Optimized for protection against SIFA-1 attacks, it guarantees robustness to SIFA-1 which targets up to 4 bits in the internal state register.

Future research may focus on a deeper analysis of the correlation between the redundancy and the leakage. Furthermore, it will be interesting to study applications of RAMBAM-like approaches to other cryptographic algorithms defined in terms of Galois field arithmetic, such as ARIA and SM-4.

References

[AG01] Mehdi Laurent Akkar and Christophe Giraud. An implementation of DES and AES, secure against some attacks. Technical report, 2001.

[Can05] D. Canright. A very compact S-box for AES. *Lecture Notes in Computer Science*, 3659:441–455, 2005.

[CB08] D. Canright and Lejla Batina. A very compact "perfectly masked" S-box for AES. Technical report, 2008.

[CEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware Masking, Revisited. *{IACR} Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):123–148, 2018.

[CZC+17] Wei Cheng, Chao Zheng, Yuchen Cao, Yongbin Zhou, Hailong Zhang, Sylvain Guilley, and Laurent Sauvage. How Far Can We Reach? Breaking RSM-Masked {AES-128} Implementation Using Only One Trace. *{IACR} Cryptol. ePrint Arch.*, 2017:1144, 2017.

[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} Trans. Cryptogr. Hardw. Embed. Syst.*, 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.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance validation. *NIST Workshop 2011*, pages 1–12, 2011.

[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.

[GT03] Jovan D. Golić and Christophe Tymen. Multiplicative Masking and Power Analysis of AES. Technical report, 2003.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1666, pages 388–397. Springer, Berlin, Heidelberg, 1999.

[LMW14] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. Gate-level masking under a path-based leakage metric. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8731:580–597, 2014.

[MGH14] Amir Moradi, Sylvain Guilley, and Annelie Heuser. Detecting hidden leakages. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8479 LNCS:324–342, 2014.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. *Lecture Notes in Computer Science*, 3376:351–365, 2005.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *Lecture Notes in Computer Science*, volume 3659, pages 157–171, 2005.

[Nan08] Inc. NanGate. NanGate FreePDK45 Open Cell Library, 2008.

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

[NSGD12] Maxime Nassar, Youssef Souissi, Sylvain Guilley, and Jean Luc Danger. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. *Proceedings -Design, Automation and Test in Europe, DATE*, pages 1173–1178, 2012.

[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.

[PM05] Thomas Popp and Stefan Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. *Lecture Notes in Computer Science*, 3659:172–186, 2005.

[RAD20] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. RS-Mask: Random Space Masking as an Integrated Countermeasure against Power and Fault Analysis. *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2020*, pages 176–187, 11 2020.

[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.

[SJR+20] Sayandeep Saha, Dirmanto Jap, Debapriya Basu Roy, Avik Chakraborty, Shivam Bhasin, and Debdeep Mukhopadhyay. A Framework to Counter Statistical Ineffective Fault Analysis of Block Ciphers Using Domain Transformation and Error Correction. *IEEE Transactions on Information Forensics and Security*, 15:1905–1919, 2020.

[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.

[Tri03] Elena Trichina. Combinational logic design for aes subbyte transformation on masked data. Technical report, 2003.

[TV04] Kris Tiri and Ingrid Verbauwhede. A Dynamic and Differential {CMOS} Logic Style to Resist Power and Timing Attacks on Security {IC's}. *Cryptology ePrint Archive, Report 2004/066*, pages 1–23, 2004.

[WHB99] Huapeng Wu, M. Anwarul Hasan, and Ian F. Blake. Highly regular architectures for finite field computation using redundant basis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1717:269–279, 1999.

[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.

[Wol16] Clifford Wolf. Yosys open synthesis suite, 2016.

A Proof of the Minimal Required Number of Multiplications

In this appendix we research optimal algorithms of inversion by exponentiation in finite fields $GF(2^{2^n})$ for an arbitrary natural n. The value $n = 3$ is relevant for AES. We will denote $N = 2^{2^n}$. In such a field, if $x \neq 0$ then $x^{N-1} = 1$, so $x^{N-2} = x^{-1}$, and the function $F(x) = x^{N-2}$ is the same as the inversion, except that it is defined in 0: $F(0) = 0$, which is exactly what is needed for the AES Sbox function. In any field of characteristic 2, raising to a power of 2^k for any natural k is a linear transformation, and is therefore much cheaper in a HW implementation than a multiplication. We search for optimal sequences of transformations, each transformation being either raising to a power of 2^k for a natural

k or a multiplication, that produces x^{N-2} from x. By "optimal" we mean a sequence with the lowest possible number of multiplications.

Note that any such sequence of operations applied to any $x \in GF(2^{2^n})$, produces a series of field elements $x = x^1 = x^{e_0}, x^{e_1}, \ldots, x^{e_{m-1}} = x^{N-2}$, where the sequence of exponents $e_0 = 1, e_1, \ldots, e_{m-1} = N-2$ is *well formed of order* n in the sense of the following definition:

Definition 2. For any natural n, m, a sequence $e_0, e_1, \ldots, e_{m-1}$ is called *well formed of order* n if $e_0 = 1, e_{m-1} = N-2$, and every element of the sequence except for e_0 is equal either to one of the previous elements multiplied by 2^k modulo $N-1$, or to the sum of any two previous elements modulo $N-1$.

In this definition, raising to a power and addition are performed modulo $N-1$, because this is the order of the multiplicative group of $GF(2^{2^n})$.

We will prove a lower bound for the number of multiplications in a well formed sequence of order a as a function of n.

Lemma 1. *For any natural n and any natural $k < 2^n$, $gcd(2^k + 1, 2^{2^n} - 1) > 1$.*

Proof. Let t be the maximal natural number such that k is divisible by 2^t. Then $k = 2^t r$, where r is odd. Then it is easy to see that both $2^k + 1 = 2^{2^t r} + 1$ and $2^{2^n} - 1$ are divisible by $2^{2^t} + 1$. □

Theorem 1. *For $n > 2$, the number of additions in any well formed sequence of order n is at least $n + 1$.*

Proof. We will call two remainders modulo $N-1$ *equivalent* if one can be produced from the other by a cyclic permutation of the digits in the binary representation. It is easy to see that multiplication by 2^k modulo $N-1$ is equivalent to a cyclic permutation of the digits in the binary representation. Therefore, in the sequence of the equivalence classes of elements of a well formed sequence, multiplications by 2^k just repeat the same equivalence class. Dropping the repetitions, we receive a sequence of equivalence classes $c_0, c_1, \ldots, c_t$ in which every class, except for the first one, is the class of a sum of two elements of preceding (not necessarily different) classes. The number t of elements in this sequence of classes, not counting the first one, will be the number of additions in the original sequence.

It is easy to see that

$$HW(x + y) \leq HW(x) + HW(y) \tag{3}$$

where $HW(x)$ (Hamming weight of x) stands for the number of "1" bits in the binary representation of x. Furthermore, it is easy to see that

$$HW\big(x \bmod (2^k - 1)\big) \leq HW(x) \tag{4}$$

because $HW\big(x \bmod (2^k - 1)\big)$ can be calculated by cutting x into k-bit blocks, adding them up, and applying the same operation to the result if it is still too large. Combining (3) and (4) we receive

$$HW\big((x + y) \bmod (2^k - 1)\big) \leq HW(x) + HW(y) \tag{5}$$

Defining the Hamming weight of a class as the Hamming weight of any of its elements (that differ only by a cyclic permutation of their binary digits and therefore have the same Hamming weight), and using (5) it is easy to prove by induction that $HW(c_i) \leq 2^i$. Since $HW(c_t) = HW(N-2) = 2^n - 1$, we conclude that $t \geq n$.

Now we will prove by contradiction that $t \neq n$; from that, $t \geq n + 1$ will follow.

Suppose $t = n$, and suppose $z = x + y$, where z is a representative of c_t, and x, y are representatives of some preceding classes. We have $2^n - 1 = HW(z) \leq HW(x) + HW(y)$,

so either $HW(x) \geq 2^{n-1}$ or $HW(y) \geq 2^{n-1}$. Without loss of generality suppose $HW(x) \geq 2^{n-1}$. Since $x \in c_i$ for $i < n$, the only possibility is that $i = n-1$, and for every $j < n-1$ the class c_{j+1} is generated by a sum of two different elements of the same class c_j. In particular, the class c_1 is generated by two different elements of c_0. Since the elements of c_0 are two powers of two, a sum of two different elements of c_0 can be written as $2^{a+b} + 2^a = 2^a(2^b + 1)$, where $a \geq 0, b > 0$, so an element of c_1, and therefore any element of c_1, are divisible by $2^b + 1$. By induction, all elements of the following classes $c_2, \ldots, c_{t-1}$, and in particular x, are divisible by $2^b + 1$.

Regarding the second addend y of the sum $z = x + y$, there are two cases.

Case 1. $y \in c_0$. Then $HW(y) = 1$, and $HW(N-2) \leq HW(x) + HW(y) = 2^{n-1} + 1$. Since by the assumption $n > 2$, $2^{n-1} > 2$, and we have $HW(N-2) \leq 2^{n-1} + 1 < (2^{n-1} + 1) + (2^{n-1} - 2) = 2^n - 1$; a contradiction.

Case 2. $y \in c_i$ for $i > 0$. Then according to the above both x and y are divisible by $2^b + 1$, and by Lemma 1 are not coprime with $N-1$, but $N - 2 = x + y$ clearly is coprime with $N - 1$; a contradiction. □

The practical case for AES is $n = 3$, and the number of multiplications according to Theorem 1 cannot be less than $n+1 = 4$. Both versions of the function $ProtectedSubBytes_{P,Q,d}$ (Algorithm 2 and Algorithm 5) reach this theoretical minimum.

B Alternative $ProtectedSubBytes$ Optimized for the Maximal Frequency

Algorithm 5: ProtectedSubBytes (optimized for maximal frequency)

Function $ProtectedSubBytes_{P.Q,d}(x\_in[16], r[24])$
  Input : $x\_in[16]$ - 16 $(8+d)$-bit values, representing the AES state bytes
    $r[24]$ - 24 random d-bit values (only the last 8 are used here)
  Output : $x\_out[16]$ - 16 $(8+d)$-bit values after the Sbox transformation
  for $i = 0$ **to** 15 **do**
    $t = x\_in_i$
    $t2 = Pow2_{P.Q,d}(t) + r_{16}P$
    $t3 = Mul_{P.Q,d}(t, t2) + r_{17}P$
    { `/* parallel section */`
      $t12 = Pow4_{P.Q,d}(t3) + r_{18}P$
      $t48 = Pow16_{P.Q,d}(t3) + r_{19}P$
      $t192 = Pow64_{P.Q,d}(t3) + r_{20}P$
    }
    { `/* parallel section */`
      $t14 = Mul_{P.Q,d}(t2, t12) + r_{21}P$
      $t240 = Pow16_{P.Q,d}(t15) + r_{22}P$
    }
    $t254 = Mul_{P.Q,d}(t14, t240) + r_{23}P$
    $x\_out_i = RAff_{P.Q,d}(t254)$
    Rotate $r[16 \ldots 23]$ by one position
  end for
  return x_out
end

C Robustness Against SIFA-1

Table 4 lists the maximal number of bits in a SIFA-1 target for which SIFA-1 is guaranteed to fail, as a function of the polynomial P and the redundancy d. The last row shows the maximal value for a fixed redundancy d over all the polynomials P.

Table 4: Robustness Against SIFA

	Redundancy d																			
P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0x11b	0	0	0	1	2	2	3	3	3	4	4	5	5	6	6	6	7	8	8	9
0x11d	0	0	0	1	1	2	3	3	3	3	4	4	5	5	6	6	7	7	7	8
0x12b	0	0	1	1	2	2	2	2	2	3	4	5	5	5	6	6	6	7	7	8
0x12d	0	0	1	1	2	2	2	3	3	4	4	4	5	5	6	6	7	7	7	8
0x139	0	0	1	1	1	2	3	4	5	5	5	5	5	5	5	5	6	6	6	7
0x13f	0	0	1	1	1	2	2	2	3	4	4	4	5	6	6	6	6	7	8	8
0x14d	0	0	1	1	1	2	2	3	3	4	4	4	5	5	6	7	7	7	8	8
0x15f	0	1	1	1	2	2	2	2	2	2	3	3	4	4	4	5	5	6	6	7
0x163	0	0	0	1	1	1	2	3	3	4	4	5	5	5	6	6	6	7	7	8
0x165	0	0	1	1	1	2	2	3	3	4	4	4	5	5	6	7	7	7	8	8
0x169	0	0	1	1	2	2	2	3	3	4	4	4	5	5	6	6	7	7	7	8
0x171	0	0	0	1	1	2	3	3	3	3	4	4	5	5	6	6	7	7	7	8
0x177	0	1	1	1	1	1	2	2	3	3	3	4	4	5	5	5	6	6	6	7
0x17b	0	1	1	1	2	2	2	3	3	3	3	4	4	4	5	6	7	7	7	8
0x187	0	0	0	0	1	1	2	2	3	4	4	5	5	6	6	7	7	7	8	8
0x18b	0	0	0	1	1	1	2	3	3	4	4	4	4	5	6	6	7	7	7	8
0x18d	0	0	0	1	1	1	2	3	3	4	4	5	5	5	6	6	6	7	7	8
0x19f	0	0	1	1	2	2	2	2	3	3	3	3	4	4	5	5	5	6	7	8
0x1a3	0	0	0	1	1	1	2	3	3	4	4	4	4	5	6	6	7	7	7	8
0x1a9	0	0	1	1	2	2	2	2	2	3	4	5	5	5	6	6	6	7	7	8
0x1b1	0	0	0	1	2	2	3	3	3	4	4	5	5	6	6	6	7	8	8	9
0x1bd	0	1	1	1	2	2	2	3	3	3	3	4	4	4	5	6	7	7	7	8
0x1c3	0	0	0	0	1	1	2	2	3	4	4	5	5	6	6	7	7	7	8	8
0x1cf	0	0	1	1	1	2	2	2	3	4	4	5	5	6	6	7	7	8	9	9
0x1d7	0	1	1	1	1	2	3	4	5	5	5	5	5	5	5	5	6	6	7	7
0x1dd	0	1	1	1	1	1	2	2	3	3	3	4	4	5	5	5	6	6	6	7
0x1e7	0	0	1	1	1	2	2	2	3	4	4	5	5	6	6	7	7	8	9	9
0x1f3	0	0	1	1	2	2	2	2	3	3	3	3	4	4	5	5	5	6	7	8
0x1f5	0	1	1	1	2	2	2	2	2	2	3	3	4	4	4	5	5	6	6	7
0x1f9	0	0	1	1	1	2	2	2	3	4	4	4	5	6	6	6	6	7	8	8
max	0	1	1	1	2	2	3	4	5	5	5	5	5	6	6	7	7	8	9	9