Let's Share a Secret: Share-Reduced Design of M&M for the AES S-box

Haruka Hirata[1][0009−0004−0997−3340], Daiki Miyahara[1][0000−0002−5818−8937], Yuko Hara[2][0000−0001−9486−5272], Kazuo Sakiyama[1][0000−0002−4414−815X], and Yang Li[1][0000−0003−0219−5289]

[1] The University of Electro-Communications, Tokyo, Japan
{h.haruka,miyahara,liyang,sakiyama}@uec.ac.jp
[2] Institute of Science Tokyo, Tokyo, Japan
hara@cad.ict.e.titech.ac.jp

Abstract. Masks and Macs (M&M), proposed at TCHES 2019, is a powerful countermeasure against physical attacks. This scheme combines masking and circuit duplication with information-theoretic MAC tags. Specifically, masking mitigates side-channel analysis, while circuit redundancy protects against fault analysis. However, it was reported that the AES encryption circuit protected by M&M is vulnerable to zero-value attacks, and an extended countermeasure was proposed to address this issue. Counteracting the attacks is mandatory due to their ease of execution, but this proposed method requires additional circuits to detect faults. While the implementation cost remains reasonable for lower-security-order designs, it becomes considerable for higher-security-order implementations. In this paper, we propose an area-efficient implementation of M&M that reduces the number of shares while preserving its functionality. We focus on the S-box and compare our approach's security and implementation cost to the original M&M. Furthermore, we discuss the limitations of share reduction in maintaining the security of the original M&M scheme.

Keywords: AES · M&M · side-channel analysis · fault analysis · combined countermeasure.

1 Introduction

Side-channel analysis exploits physical information leaked during cryptographic operations, such as power consumption, electromagnetic emissions, or processing time [13, 14]. On the other hand, in fault analysis, an attacker intentionally induces faults in cryptographic devices and then analyzes the resulting ciphertext [4]. Since such attacks that exploit physical information or faults do not rely on the mathematical strength of cryptographic algorithms, countermeasures at the implementation level are crucial. However, countermeasures for these physical attacks are often discussed separately, and research on comprehensive and efficient countermeasures remains limited.

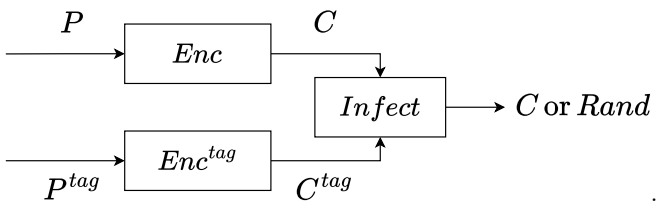

Fig. 1. An overview of M&M. The circuits are duplicated for plaintext and tag, and encryptions are performed in parallel.

Masks and Macs proposed by De Meyer et al. at TCHES 2019 [8] provides resistance to side-channel and fault analysis by combining masking with circuit redundancy using information-theoretic message authentication code (MAC) tags used for verifying the output. This scheme can be implemented with reasonable costs compared to other countermeasures [19, 21], which were state-of-the-art at the time. However, it was reported that the AES encryption circuit protected by M&M (hereafter referred to as M&M-AES) is vulnerable to zero-value attacks at TCHES 2022 [10]. This vulnerability is caused by Canright's design for AES S-box [5], and the tag corresponding to zero is also zero in the M&M scheme. As a result, faults remain undetected on both value and tag circuits, and a correct ciphertext is obtained, i.e., the fault is ineffective. To counteract zero-value attacks, the authors of [10] added detection circuits to the intermediate calculations of the S-box, enabling fine-grained fault detection. While the implementation costs of their schemes for second-order security are reasonable, they would be considerable for higher-order security implementations because the added circuits for detection are nonlinear operations. The costs increase as $(d+1)^2$ with the security order d. Despite the considerable costs, counteracting zero-value attacks is inevitable due to their ease of execution.

In this paper, we take on this challenge and attempt to design a share-reduced version of M&M-AES to suppress the implementation cost overhead. Specifically, we focus on reducing the number of shares in the masking implementation of the S-box, thereby achieving a more efficient M&M-AES. This paper proposes a share-reduced implementation of the S-box and discusses the trade-off between effectiveness and security against probing attacks and differential fault analysis (DFA) attacks. To the best of our knowledge, this is the first work to reduce the number of shares on the redundant circuit while maintaining its security.

The rest of the paper is organized as follows. Section 2 provides previous work, an overview of M&M, zero-value attacks, and their countermeasure. Section 3 describes a methodology to implement the share-reduced S-box. In Section 4, we compare implementation costs and analyze security. Finally, we conclude this paper in Section 5.

Algorithm 1 $Infect$

Input: c, τ^c, α
Output: c or $rand$

$R \xleftarrow{\$} GF(2^k)\backslash\{0\}$
$\tilde{c} \leftarrow c + R \cdot (\alpha \cdot c + \tau^c)$
Output $\tilde{c}$

2 Previous work

This section provides an overview of previous works, the M&M technique, and describes zero-value attacks on M&M-AES and their countermeasures, extending the original M&M scheme.

2.1 Masks and Macs

Redundant circuits using MAC tag In the M&M scheme, the value circuit, which receives the plaintext, and the tag circuit for the MAC tag of the plaintext are implemented in parallel, as shown in Figure 1. The tag τ^x corresponding to the value $x \in GF(2^k)$ is obtained using the tag key $\alpha \in GF(2^k)$ as $\tau^x := \alpha x$, and is used to verify the correctness of the output. Simply redundantly implementing the value circuit allows an attacker to bypass the fault detection by causing the same fault in the replicated circuits. In contrast, M&M makes bypassing fault detection more difficult by redundantly implementing the tag circuit instead of the value circuit, a countermeasure against Differential Fault Analysis (DFA) attacks. Fault detection is performed by comparing the outputs of each circuit using the $Infect$ function. The calculation method for $Infect$ is described in Algorithm 1. If the ciphertext c or the tag τ^c contains a fault, then $\alpha \cdot c + \tau^c \neq 0$, meaning that the faulty ciphertext masked with random values will be output. The $Infect$ calculation is typically performed at the block size level. For example, spreading a one-byte fault across 16 bytes is required as a countermeasure against DFA attacks. However, the multiplication cost for 16 bytes is very high, which is a significant issue. Therefore, in the M&M-AES implementation, following the method by Lomné et al. [15], one-byte computation is performed 16 times instead of a 128-bit multiplication.

Masking against side-channel analysis Masking [11] is a well-studied technique to counteract the side-channel analysis, originated from Ishai, Sahai, and Wagner, where the secret data is encoded into multiple values, called shares, using random numbers. The circuit keeps the value in a shared form until the computation ends. Thus, the encryption is performed in a shared form. When the number of shares is $d + 1$, the scheme is resistant to observation using d probes (known as d-probing attacks), and d is referred to as the security order. In addition to circuit redundancy for the Countermeasure against DFA, the

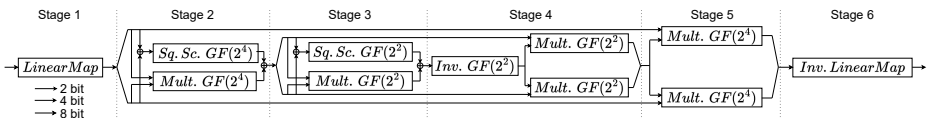

Fig. 2. The pipelined AES S-box with six stages proposed by De Cnudde et al. [7].

M&M scheme implements masking for both the value and tag circuits. Therefore, the implementation cost of the circuit countermeasures by M&M increases by approximately $2(d+1)^2$.

While M&M allows the use of arbitrary masking methods, M&M-AES [8] employs a Consolidating Masking Scheme [18], hereafter referred to as CMS, which extends threshold implementation [17] as a case study implementation.

2.2 Zero-value attacks against M&M-AES

The calculation of the multiplicative inverse has a high implementation cost. Canright proposed a compact inversion algorithm [5], and the inversion of zero is defined as zero. The inverse circuit for M&M-AES is shown in Figure 2. In the following, we explain the vulnerability in Canright's design, exploiting zero-value input.

First, the mapping λ_n from $GF((2^n)^2)$ to $GF(2^n)$ is defined as follows:

$$\lambda_n((a,b)) := ab + (a+b)^2\nu, \tag{1}$$

where $\nu \in GF(2^n)$ is a constant. The elements of $GF((2^n)^2)$ are represented in the form $a\beta^{2^n} + b\beta$, where $a, b \in GF(2^n)$, with β being a root of an irreducible polynomial, and $[\beta^{2^n}, \beta]$ denoting the normal basis. Still, this paper represents it as a pair (a, b) for simplicity.

The multiplicative inverse in $GF((2^n)^2)$ is given by

$$(c,d) = (b\ \lambda_n((a,b))^{-1}, a\ \lambda_n((a,b))^{-1}), \tag{2}$$

and the inverse $y = x^{-1}$ is computed as follows:

$$c = \left[ab + (a+b)^2\nu\right]^{-1} b, \tag{3}$$

$$d = \left[ab + (a+b)^2\nu\right]^{-1} a. \tag{4}$$

The computation within the brackets corresponds to Stage 2 of the circuit shown in Figure 2, which is λ_4. When the input x is zero, the outputs c and d will also be zero. The value in Stage 1 after applying the field isomorphism to $GF((2^4)^2)$ is $(a, b) = (0, 0)$, and thus the final multiplication of inversion multiplies by zero if and only if $x = 0$. Multiplying by zero is the key point of the zero-value attack; even if an attacker injects faults during the computation inside the brackets (i.e., inversion over $GF(2^4)$), these errors are nullified by multiplication with zero, resulting in a correct output value. Such faults that do not affect the output are called ineffective faults.

2.3 Countermeasure against zero-value attacks

As mentioned, M&M is ineffective against ineffective faults caused at the algorithm level, and zero-value attacks can be easily conducted. To counter zero-value attacks, Hirata et al. [10] added detection circuits to the susceptible stages, i.e., Stages 2, 3, and 4, to achieve fine-grained detection. Fault detection is performed by utilizing the intermediate values of the inversion on the value and tag circuits, i.e., the output of each stage.

Hirata et al. [10] showed that the map λ has multiplicative homomorphic properties. By this property, the faults can be detected before they are nullified by multiplying by zero. The outputs of Stage 2–4 can be written as $\lambda_4(x)$, $\lambda_2(x)$, and $\lambda_4(x)^{-1}$, respectively. Since the outputs of each stage in the value and tag circuits are $\lambda_n(x)$ and $\lambda_n(\alpha x)$, the following equation always holds for any x and α due to the homomorphism property of λ:

$$\lambda_n(x)\lambda_n(\alpha) + \lambda_n(\alpha x) = 0. \tag{5}$$

Therefore, faults can be detected by comparing the product of outputs of the value and the α circuits at each stage with the output of the tag circuit as in Eq. (5). Furthermore, a masked OR-accumulator accumulates the calculation results of λ to prevent attackers from determining the timing of fault occurrence.

The fault detection technique described above requires three multiplication circuits, a comparator (for calculating Eq. (5)), and an OR-accumulator register. After the encryption, the circuit computes $Match\,check$ described in Algorithm 2 instead of $Infect$, then applies Kronecker's δ-function to check whether the fault has been injected or not. The additional modules are summarized as follows:

- Multiplication,
- OR-accumulator,
- δ-function.

These functions and accumulators are nonlinear operations and implemented in a shared form; thus, their costs are estimated as $(d+1)^2$ with a security order d. The second-order λ-detection M&M cost overhead factor is 1.33×, comparing with the original M&M, as reported in [10]. While this overhead remains reasonable for second-order implementation, higher security orders would significantly increase implementation costs.

Moreover, the amount of randomness required for refreshing increases as the security order increases. Despite this, the rise in cost is inevitable as a countermeasure against zero-value attacks, which are easy to execute. Therefore, in the following sections, we aim to reduce the implementation cost of M&M-AES while balancing robustness and cost efficiency.

3 Implementation of a share-reduced S-box

In this section, we explore the implementation of M&M-AES with a reduced number of shares, focusing on the S-box.

Algorithm 2 *Match check*

Input: c, τ^c, α
Output: z
$z = 0$
for $i = 1$ to 16 **do**
  $z_i \leftarrow \alpha \cdot c_i \oplus \tau_i^c$ // just compare
  $z \leftarrow$ shared-OR(z, z_i) // then accumulate
end for
Output z

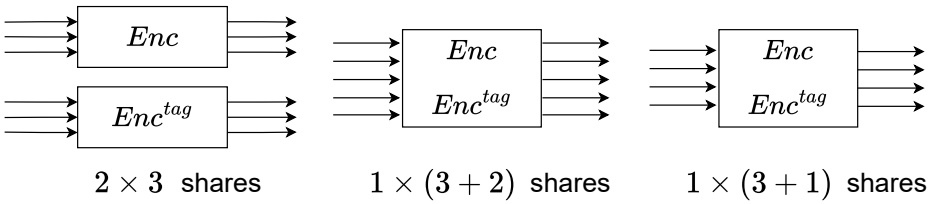

Fig. 3. The countermeasure circuit based on spatial redundancy, such as M&M (left), and the share-reduced M&Ms proposed in this paper (middle and right).

3.1 Overview of the share-reduced implementation

The conceptual diagram of the circuit is shown in Figure 3. In duplication-based countermeasures such as M&M, the total number of shares, $d + 1$, increases with the number of redundant circuit components. For example, as illustrated in Figure 3, adding one redundant circuit doubles the total number of shares to $2(d + 1)$. In contrast, we propose a share-reduced variant of M&M to lower implementation costs. We consolidate the value and the tag circuit, where several shares are literally *shared* among the circuits, and indicate the number of shares that can be reduced.

As depicted in Figure 4, we reuse the random value(s) in the value circuit and the tag circuit, and we name the resued shares *dependent share* S_{dep} and the others shares *independent share* S_{indep}. Hereafter, we refer to S_{dep} and S_{indep} as the number of shares, not the share value.

This paper describes an implementation methodology for a four-share implementation as an example. However, the five-share version can be implemented in the same way. Hence, partial summations of the shares indicate x and τ^x, respectively, and r_0 and r_1 are common:

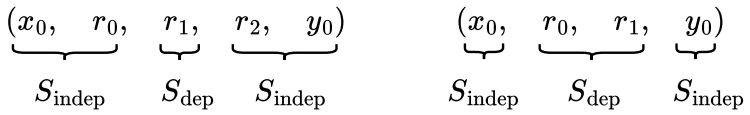

Fig. 4. (In)dependent shares of five-share (left) and four-share (right) implementation.

$$\begin{aligned} x &= x_0 + r_0 + r_1, \\ \tau^x &= \quad r_0 + r_1 + y_0. \end{aligned}$$

3.2 Implementing a share-reduced S-box

As already mentioned in Section 2.2, the inversion of M&M-AES S-box is implemented based on Canright's algorithm and the proposal by De Cnudde et al. (shown in Figure 2). The implementation comprises the following four operations, explaining Canright's algorithm in detail.

- Field isomorphism $\phi\colon GF(2^8) \to GF((2^4)^2)$, and its inverse ϕ^{-1},
- Squaring and scaling over $GF(2^4)$, $GF(2^2)$,
- Inversion over $GF(2^2)$,
- Multiplication over $GF(2^4)$, $GF(2^2)$.

Except for multiplication, the operations above are additively holomorphic. We show that each operation is an additively homomorphic map and describe the method for performing multiplication for this design.

Field isomorphism The map ϕ and its inverse map ϕ^{-1} are defined as field automorphisms. Therefore, for any $a, b \in GF(2^8)$, the operation is trivially homomorphic for both addition and multiplication, i.e., $\phi(a)+\phi(b) = \phi(a+b)$ and $\phi(a)\phi(b) = \phi(ab)$ holds (the same holds for ϕ^{-1}).

Squaring and scaling The sum of two inputs is squared and scaled by a constant value. The function $f_n\colon (a, b) \mapsto (a+b)^2\nu$, where $a, b, \nu \in GF(2^n)$, can be similarly defined for any n.

$$\begin{aligned} f_n((a,b)) + f_n((c,d)) &= (a+b)^2\nu + (c+d)^2\nu && (6)\\ &= (a^2+b^2+c^2+d^2)\nu, && (7) \end{aligned}$$

$$\begin{aligned} f_n((a,b)+(c,d)) &= f_n((a+c, b+d)) && (8)\\ &= ((a+b)+(c+d))^2\nu && (9)\\ &= (a^2+b^2+c^2+d^2)\nu. && (10) \end{aligned}$$

The function has additive homomorphic properties, as shown in Eqs. (7) and (10).

Inversion over $GF(2^2)$ This computation is just a bit-swapping and is denoted as $g\colon (a, b) \mapsto (b, a)$ for $a, b \in GF(2)$.

$$\begin{aligned} g((a,b)) + g((c,d)) &= (b,a) + (d,c) && (11)\\ &= (b+d, a+c), && (12) \end{aligned}$$

$$g((a,b)+(c,d)) = g((a+c, b+d)) \tag{13}$$
$$= (b+d, a+c). \tag{14}$$

As with squaring and scaling, from Eqs. (12) and (14), it can be seen that the multiplication inversion over $GF(2^2)$ exhibits additively homomorphic properties.

Again, we note that all of the above calculations are additive homomorphic. Therefore, calculations can be performed share-wise, and the output share can thus be obtained with the additively homomorphic function F.

$$(z_0,\, z_1,\, z_2,\, z_3) \leftarrow (F(x_0),\, F(r_0),\, F(r_1),\, F(y_0)).$$

Multiplication As mentioned, multiplication is not a homomorphic function, and independent calculations for each share cannot be performed. Therefore, we calculate them with all shares. Here, we take the multiplication on Stage 2 as an example. The upper and lower four bits of x are denoted as x^{top} and x^{bot}, respectively (applied for y as well). We compute $x^{top}x^{bot} = (x_0^{top} + r_0^{top} + r_1^{top})(x_0^{bot} + r_0^{bot} + r_1^{bot})$ and $y^{top}y^{bot} = (y_0^{top} + r_0^{top} + r_1^{top})(y_0^{bot} + r_0^{bot} + r_1^{bot})$. The resulting terms for the value ($x^{top}x^{bot}$) and the tag ($x^{top}x^{bot}$) are as follows:

$$\begin{aligned} &x^{top}x^{bot} \\ &= x_0^{top}x_0^{bot} + x_0^{top}r_0^{bot} + x_0^{top}r_1^{bot} \\ &\quad + r_0^{top}x_0^{bot} + \boldsymbol{r_0^{top}r_0^{bot}} + \boldsymbol{r_0^{top}r_1^{bot}} \\ &\quad + r_1^{top}x_0^{bot} + \boldsymbol{r_1^{top}r_0^{bot}} + \boldsymbol{r_1^{top}r_1^{bot}}, \end{aligned} \tag{15}$$

$$\begin{aligned} &y^{top}y^{bot} \\ &= y_0^{top}y_0^{bot} + y_0^{top}r_0^{bot} + y_0^{top}r_1^{bot} \\ &\quad + r_0^{top}y_0^{bot} + \boldsymbol{r_0^{top}r_0^{bot}} + \boldsymbol{r_0^{top}r_1^{bot}} \\ &\quad + r_1^{top}y_0^{bot} + \boldsymbol{r_1^{top}r_0^{bot}} + \boldsymbol{r_1^{top}r_1^{bot}}. \end{aligned} \tag{16}$$

As shown in Eqs. (15) and (16), the terms in bold $r_0^{top}r_0^{bot}$, $r_0^{top}r_1^{bot}$, $r_1^{top}y_0^{bot}$, $r_1^{top}r_0^{bot}$, and $r_1^{top}r_1^{bot}$ are overlapping between the value and tag calculations; thus we can omit these computations. In general, the number of overlapping multiplications is represented as $(d_{\mathrm{dep}})^2$. Therefore, the 18 ($= 2 \times (3^2)$) multiplications are reduced to 14 for $d_{\mathrm{dep}} = 2$.

Then, we combine the terms relating x_0 to the share of z_0 (the same as for y_0) to satisfy the correctness of the output shares, namely, ensuring $x_0 + r_0 + r_1 = x$ and $y_0 + r_0 + r_1 = \tau^x$. Hence, the output share is computed as follows, where registers are denoted as $[\cdot]_{reg}$, and R_i represents a random value for refreshing:

$$\begin{aligned} t_0 &= x_0^{top}x_0^{bot} + R_0 + R_1 \\ t_1 &= x_0^{top}r_0^{bot} + R_1 + R_2 \\ t_2 &= x_0^{top}r_1^{bot} + R_2 + R_3 \\ t_3 &= r_0^{top}x_0^{bot} + R_3 + R_4 \\ t_4 &= r_1^{top}x_0^{bot} + R_4 + R_5 \end{aligned} \qquad z_0 = [t_0]_{reg} + [t_1]_{reg} + [t_2]_{reg} + [t_3]_{reg} + [t_4]_{reg}$$

$$
\begin{aligned}
t_5 &= r_0^{top} r_0^{bot} + R_5 + R_6 \\
t_6 &= r_0^{top} r_1^{bot} + R_6 + R_7 \qquad & z_1 &= [t_5]_{reg} + [t_6]_{reg} \\
t_7 &= r_1^{top} r_0^{bot} + R_7 + R_8 \\
t_8 &= r_1^{top} r_1^{bot} + R_8 + R_0 \qquad & z_2 &= [t_7]_{reg} + [t_8]_{reg} \\
t_9 &= y_0^{top} y_0^{bot} + R_0 + R_9 \\
t_{10} &= y_0^{top} r_0^{bot} + R_9 + R_{10} \\
t_{11} &= y_0^{top} r_1^{bot} + R_{10} + R_{11} \\
t_{12} &= r_0^{top} y_0^{bot} + R_{11} + R_{12} \\
t_{13} &= r_1^{top} y_0^{bot} + R_{12} + R_5 \qquad & z_3 &= [t_9]_{reg} + [t_{10}]_{reg} + [t_{11}]_{reg} + [t_{12}]_{reg} + [t_{13}]_{reg}
\end{aligned}
$$

Based on the CMS scheme, we refresh shares with the ring-refreshing approach of [7]. We discuss the security for the probing model of the implementation in Section 4.3.

Multplying α^2 to the tag shares The multiplicative inverses obtained from the calculations are x^{-1} and $(\alpha x)^{-1}$, but the value and tag do not match ($\because (\alpha x)^{-1} \neq \tau^{x^{-1}}$). By multiplying the tag share by α^2, the tag corresponding to x^{-1} is obtained ($\alpha^2(\alpha x)^{-1} = \alpha x^{-1}$). However, the share after multiplying by α^2 becomes (x_0, $\alpha^2 r_0$, $\alpha^2 r_1$, $\alpha^2 y_0$), and the shares for x no longer satisfies the correctness, i.e., $x_0 + \alpha^2 r_0 + \alpha^2 r_1 \neq x$. To tweak this, we add r_0 and r_1 to the x_0 share, and $\alpha^2 r_0$ and $\alpha^2 r_1$ to the y_0 share. Thus, the share for the inversion output is as follows.

$$
\begin{aligned}
z_0 &\leftarrow x_0 + \alpha^2 r_0 + \alpha^2 r_1, \\
z_1 &\leftarrow \alpha^2 r_0 + r_0, \\
z_2 &\leftarrow \alpha^2 r_1 + r_1, \\
z_3 &\leftarrow \alpha^2 y_0 + r_0 + r_1.
\end{aligned}
$$

Affine transformations We compute an affine transformation with the inversions to obtain the S-box outputs. The calculation for the value circuit is straightforward: $A(x) = L(x) + c$, where L is a linear transformation and $c = 0\text{x}63$ is a constant value, while the tag shares require another matrix for the tag circuit.

From [8], the matrix $\mathbf{M_{tag}}$ for a transformation of the tag is obtained as follows;

$$
\mathbf{M_{tag}} = [\psi(\alpha L(128\alpha^{-1})) \quad \psi(\alpha L(64\alpha^{-1})) \quad \cdots \quad \psi(\alpha L(\alpha^{-1}))],
$$

where ψ is the isomorphism between $GF(2^8)$ and $GF(2)^8$. The matrix depends on the tag key α and can be precomputed; hence, it is represented as a constant value. The affine transformation for the tag is computed as $A_{tag}(t) = \mathbf{M}_\alpha \psi(t) +$

Table 1. Area comparison. The column of the original M&M includes two S-boxes.

	Original M&M	This work	
	6 shares	5 shares	4 shares
Area [μm^2]	8854.61	8192.80	6717.032
Reduction ratio	-	0.075	0.24

c_{tag}, where $c_{tag} = \alpha c$. Thus, the output shares of the S-box are as follows;

$$\begin{aligned} z_0 &\leftarrow A(x_0) + A_{tag}(r_0) + A_{tag}(r_1), \\ z_1 &\leftarrow A_{tag}(r_0) + A(r_0), \\ z_2 &\leftarrow A_{tag}(r_1) + A(r_1), \\ z_3 &\leftarrow A_{tag}(y_0) + A(r_0) + A(r_1). \end{aligned}$$

Moreover, for the latency optimization, we could compute the matrix $\mathbf{M_{tag}}$ as

$$\mathbf{M_{tag}} = [\psi(\alpha L(128\alpha)) \quad \psi(\alpha L(64\alpha)) \quad \cdots \quad \psi(\alpha L(\alpha))],$$

and this allows us to omit multiplying α^2 to the tag shares after calculating the inversion. Therefore, we can directly apply the affine transformation to the inversion outputs.

The tag key α is a secret; thus, computations regarding the matrix and the tag key are performed in shared form.

4 Implementation cost and security evaluation

We compare the implementation costs of the second-order secure S-box of M&M-AES and the share-reduced S-box and analyze the security of the share-reduced implementation against Differential Fault Analysis (DFA) and probing attacks.

4.1 Implementation Cost

We implemented the S-box in Verilog-HDL and compared its circuit area and randomness implementation costs.

Area We synthesize them with the open-source tool Yosys [22] and the Nangate 45 nm Open Cell Library [16]. Circuit areas for both the M&M-AES S-box and the share-reduced S-box are shown in Table 1. The circuit area[3] was reduced by approximately 24 % for four-share and 7.5 % for five-share compared to two M&M-AES S-boxes.

[3] An area of 0.798 μm^2 corresponds to one 2-input, 1-output NAND gate.

Table 2. Implementation cost for S-box with security order d and reducing level S_{dep} (register bit).

	Original M&M-AES	Share-reduced design
Stage 1	$16(d+1)$	$16(d+1) - 8S_{\mathrm{dep}}$
Stage 2	$8(d+1)^2 + 16(d+1)$	$8(d+1)^2 + 16(d+1) - (8S_{\mathrm{dep}} + 4S_{\mathrm{dep}}^2)$
Stage 3	$4(d+1)^2 + 24(d+1)$	$4(d+1)^2 + 24(d+1) - (12S_{\mathrm{dep}} + 2S_{\mathrm{dep}}^2)$
Stage 4	$8(d+1)^2 + 16(d+1)$	$8(d+1)^2 + 16(d+1) - (8S_{\mathrm{dep}} + 4S_{\mathrm{dep}}^2)$
Stage 5	$16(d+1)^2$	$16(d+1)^2 - 8S_{\mathrm{dep}}^2$
Stage 6	$16(d+1)$	$16(d+1) - 8S_{\mathrm{dep}}$
Total	$36(d+1)^2 + 88(d+1)$	$36(d+1)^2 + 88(d+1) - (44S_{\mathrm{dep}} + 18S_{\mathrm{dep}}^2)$

Table 3. Comparison of randomness for S-box ([bit/cycle]). The columns of the original M&M include two S-boxes.

	Original M&M	This work	
	6 shares	5 shares	4 shares
Stage 1	0	0	0
Stage 2	72	64	52
Stage 3	36	32	26
Stage 4	72	64	52
Stage 5	144	128	104
Stage 6	0	0	0
Total	324	288	234

While the number of shares in the M&M circuit was reduced from 6 to 4, the multiplications in the S-box were only reduced from 18 to 14 (a reduction rate of approximately 22 %).

To compare implementation costs for higher security orders d, Table 2 summarizes the number of register bits required for the S-box implementation at each stage. According to the table, a ratio of required bits (obtained by share-reduced / original M&M-AES) is expected to increase as the share reduction level d_{dep} increases. Thus, the share-reduced implementation contributes significantly to cost efficiency when a higher reduction level is available.

Randomness The number of random bits consumed during the S-box computation is summarized in Table 3. Multiplications requiring refreshing are reduced from 18 to 14. Thus, the overall randomness consumption is also decreased. The implementation is based on the original M&M-AES from [8] and utilizes the ring-refreshing [18]. The use of randomness can be optimized by employing Domain-Oriented Masking [9], which refreshes only the cross products between different domains (e.g., $x_0^{top} \cdot r_1^{bot}$).

4.2 Security against differential fault analysis

In this section, we discuss the resistance of the proposed share-reduced S-box against Differential Fault Analysis (DFA) under two scenarios: when different

faults occur independently in the value and tag circuits, and when an identical fault occurs in both circuits.

We note that the share-reduced implementation is vulnerable to zero-value attacks as provided by Hirata [10] since it is based on the original M&M-AES. We do not consider the zero-value attacks in this paper, and the share-reduced implementation must integrate λ-detection M&M to counteract the attacks.

Different faults in the circuits We assume that faults Δ_c and Δ_τ occur independently in the value and tag circuit. This includes the case where $\Delta_c = \Delta_\tau$. A calculation for the fault detection in $Infect$ is performed as follows:

$$\begin{aligned}(c + \Delta_c)\alpha + (\tau^c + \Delta_\tau) &= c\alpha + \Delta_c\alpha + c\alpha + \Delta_\tau \\ &= \Delta_c\alpha + \Delta_\tau. \end{aligned} \tag{17}$$

When Δ_c and Δ_τ are random, the probability that equation (17) equals zero is $2^{-8} = 0.00392$. Even if the faults are constant, the probability remains 0.00392 because α takes uniformly random values. Thus, the fault detection probability of M&M is very high. Then, if $\alpha = 0$ and a fault occurs only in the value circuit (i.e., $\Delta_\tau = 0$), equation (17) will always equal zero. However, the probability $\Pr[\alpha = 0]$ is 0.00392 since the tag key α is randomly generated for each encryption operation, and this corresponds to injecting proper faults Δ_c and Δ_τ into the circuit such that equation (17) holds. Therefore, this implementation guarantees the same security as the original M&M-AES for different circuit faults.

Identical faults in the circuits In our proposed share-reduced S-box circuit, the shares r_0 and r_1 are literally "shared" between the value and tag circuits. As a result, when faults occur in both or one of the shares, identical faults are easily introduced into both the value circuit and the tag circuit. This means $\Delta_c = \Delta_\tau \neq 0$, which, for simplicity, is denoted as Δ. Under this condition, the computation in $Infect$ is given by

$$(c + \Delta)\alpha + (\tau^c + \Delta) = c\alpha + \Delta\alpha + c\alpha + \Delta \tag{18}$$

$$= \Delta\alpha + \Delta, \tag{19}$$

and equation (19) equals zero only when $\alpha = 1$.

Therefore, faults that occur in both the value circuit and the tag circuit can be detected with a high probability, unlike faults that occur independently in each circuit. However, when $\alpha = 1$, equation (19) is always zero for any fault Δ, resulting in a vulnerability to DFA attacks, and the attacker needs just a single fault on the circuit. Consequently, this tag key $\alpha = 1$ would be a flaw in the share-reduced S-box. It remains as long as the shares are "shared"; thus, the flaw is inevitable in the share-reduced implementation.

Vulnerable tag key $\alpha = 1$ As described in the previous section, the critical tag key $\alpha = 1$ exists for which M&M always fails to detect faults, making DFA

attacks easily executable. This issue can be addressed by ensuring the tag key α does not take the value 1. However, if $\alpha \in GF(2^8)$ is not uniformly distributed, the output of $Infect$ becomes biased. Moreover, it has already been pointed out that this bias can be exploited for attacks [2].

De Meyer et al. [8] noted that when α is not uniform in $GF(2^8)$, output randomization using $Infect$ becomes insecure due to the biased outputs. Therefore, we use $Match\,check$ to detect faults instead of $Infect$, following $Match\,check$ method proposed in [10], This approach does not randomize the output but merely checks for faults. If a fault is detected, all output bits are set to 0. Consequently, the bias observed in the output does not occur. Hence, DFA attacks caused by the critical key can be mitigated.

4.3 Security against probing attacks

The security order of the original M&M-AES is $d = 2$, and this guarantees that both the value circuit and the tag circuit withstand up to two probing[4]. On the other hand, we integrate these into a single circuit in our design. Hence, we consider an attacker obtaining share(s) across the value and tag circuits.

The attacker is allowed to observe up to $d = 2$ shares in the second-order attack scenario, and the joint distribution $\Pr[\text{Probe} = \{x_0, y_0\}]$, where x_0 and y_0 are independent shares, is as follows;

$$\Pr[\text{Probe} = \{x_0, y_0\}] = \frac{254}{65024}. \tag{20}$$

To guarantee second-order security against probing attacks, the conditional probability below must hold;

$$\Pr[\text{Probe} = \{x_0, y_0\} \mid x] = \Pr[\text{Probe} = \{x_0, y_0\}], \tag{21}$$

with the given unmasked, namely, the secret value x. This equation means the probing does not leak information regarding x.

However, from the following lemma, equation (21) does not hold when $x = 0$ for the four-share implementation.

Lemma 1. *$x_0 = y_0$ if and only if $x = 0$, for any tag key α.*

Proof. (1) $x = 0 \Rightarrow x_0 = y_0$.
If $x = 0$, then $x_0 = 0 + r_0 + r_1$ and $y_0 = \alpha \cdot 0 + r_0 + r_1$. Thus, $x_0 = y_0$.
(2) $x = 0 \Leftarrow x_0 = y_0$.
If $x + r_0 + r_1 = \alpha x + r_0 + r_1$, then it follows that $x = \alpha x$. Since $\alpha \notin \{0, 1\}$, this equation holds only when $x = 0$. □

Hence, the conditional probability is obtained as follows;

$$\Pr[\text{Probe} = \{x_0, y_0\} \mid x = 0] = \begin{cases} 1, & x_0 = y_0, \\ 0, & x_0 \neq y_0. \end{cases} \tag{22}$$

[4] The authors proposed the first- and second-order implementation, but we take only the second-order version in this paper.

Furthermore, when the observed values are identical, i.e., $x_0 = y_0$, the attacker would obtain information that the secret value x is zero. This observation indicates a potential for other zero-value attacks; the attacker knows whether the value processed is zero, despite the implementation must withstand up to two probes. Therefore, the security order of the four-share implementation is towards the first-order, implying that at least three shares must be independent to prevent second-order attacks.

Theorem 1. *The maximum allowable share reduction, S_{dep}, must be less than $\frac{d+1}{2}$ to ensure dth-order security.*

Proof. To ensure dth-order probing security, the number of independent shares, S_{indep}, must be at least $d+1$. Therefore,

$$S_{\text{indep}} = 2(d+1) - 2S_{\text{dep}} \geq d+1, \tag{23}$$

$$\Rightarrow \quad d+1-2S_{\text{dep}} \geq 0, \tag{24}$$

$$\Rightarrow \quad S_{\text{dep}} \leq \frac{d+1}{2}. \tag{25}$$

Thus, the maximum allowable share reduction is given by $S_{\text{dep}} \leq \frac{d+1}{2}$. □

From Theorem 1, the five-share implementation (i.e., $S_{\text{dep}} = 1$) ensures second-order probing security. The original M&M-AES, i.e., six-share implementation, follows this theorem, and any two probing reveals neither secret value x nor the tag αx, withstanding the second-order attacks.

Verification of refreshing gadgets by SILVER In our design, the refreshing of shares is performed as follows: the left side corresponds to the five-share implementation, while the right side represents the four-share implementation.

$$\begin{aligned} z'_0 &= z_0 + R_0 + R_1 \\ z'_1 &= z_1 + R_1 + R_2 \\ z'_2 &= z_2 + R_2 + R_0 \\ z'_3 &= z_3 + R_0 + R_3 \\ z'_4 &= z_4 + R_3 + R_2 \end{aligned} \qquad \begin{aligned} z'_0 &= z_0 + R_0 + R_1 \\ z'_1 &= z_1 + R_1 + R_2 \\ z'_2 &= z_2 + R_2 + R_0 \\ z'_3 &= z_3 + R_0 + R_1 \end{aligned}$$

As discussed above, the security of the design has the potential to be downgraded due to the reuse of randomness despite the reduction in implementation cost. To confirm whether the number of dependent shares, $S_{\text{dep}} = 1$, ensures security equivalent to that of the original M&M-AES, we conduct a leakage evaluation using the formal verification tool SILVER [12]. Not only did we identify a potential flaw against another type of zero-value attack, but we also observed a reduction in the security level of SNI (Strong Non-Interference) [1] in the four-share implementation due to the reuse of randomness, as shown in Table 4. On the other

Table 4. Verification results of refreshing gadgets by SILVER.

Model	Result		
	$d+1$ Masking [7]	This work	
	3 shares	5 shares	4 shares
Probing	PASS ($d \leq 2$)	PASS ($d \leq 4$)	PASS ($d \leq 3$)
NI	PASS ($d \leq 2$)	PASS ($d \leq 4$)	PASS ($d \leq 3$)
SNI	**PASS** ($d \leq 2$)	**PASS** ($d \leq 2$)	**PASS** ($d \leq 1$)
PINI	PASS ($d \leq 2$)	PASS ($d \leq 4$)	PASS ($d \leq 3$)
Uniformity	PASS	PASS	PASS

hand, the five-share implementation has the 2-SNI security equivalent to the $d+1$ masking scheme from [7], which is used in the M&M-AES implementation.

These refreshing operations are performed independently on each share, ensuring that all shares remain isolated and do not interfere. Consequently, the results of the probing, NI, and PINI [6] models indicate that the implementation can withstand up to the total number of shares minus one probing.

Here, we note that the security order of the share-reduced implementation must be less than three, i.e., $d \leq 2$, because probing x_0, r_0, and r_1 reveals a secret value x. Furthermore, the security against the probing model decreases when the value is zero, as previously mentioned. However, SILVER does not recognize this condition, leading to "PASS" results with security levels greater than two for both the five-share and four-share implementations, as shown in Table 4. Nevertheless, the verification results confirm that the design achieves at least second-order security in the probing, NI, and PINI models. Therefore, we focus on the SNI model to ensure security equivalent to the original M&M-AES.

5 Conclusion

Based on our findings, we proposed a share-reduced implementation of M&M-AES to improve area efficiency while maintaining security. By reducing the number of shares in the masking implementation of the S-box, our approach mitigates the increase in the implementation cost associated with higher-security-order designs.

We practically confirmed that the proposed design achieves reductions of area and randomness for both five-share and four-share implementations. Our analysis demonstrated that the five-share design is comparable to the original M&M-AES in terms of security against probing and DFA attacks. In contrast, the four-share design reduces the security level. Then, we identified general limitations in share reduction to guarantee the required security order.

In future work, it is essential to evaluate the security of its application to the entire AES encryption process. This study focuses on only the S-box and explores the share-reduced implementation. We will extend our new design to the entire AES encryption circuit and conduct evaluations using power consumption traces for leakage analysis based on TVLA [3, 20] and fault injection experiments for further investigation.

Acknowledgments. This work was supported by JST CREST (grant number JPMJCR23M2), JSPS Bilateral Joint Research Projects (grant number JPJSBP120242301), and JSPS Research Fellows (grant number JP25KJ1291). We would like to express our deepest gratitude to Svetla Nikova for granting permission to use the M&M source code to implement the S-box.

This is a post-peer-review, pre-copyedit version of an article published in Applied Cryptography and Network Security, 23rd International Conference, ACNS 2025, Munich, Germany, June 23–26, 2025, Proceedings, Part I. The final authenticated version is available online at: https://doi.org/10.1007/978-3-031-95761-1.

References

1. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 116–129. CCS '16, Association for Computing Machinery, New York, NY, USA (2016)
2. Battistello, A., Giraud, C.: Fault analysis of infective AES computations. In: 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography. pp. 101–107 (2013)
3. Becker, G.T., Cooper, J., DeMulder, E.K., Goodwill, G., Jaffe, J., Kenworthy, G., Kouzminov, T., Leiserson, A.J., Marson, M.E., Rohatgi, P., Saab, S.: Test vector leakage assessment (TVLA) methodology in practice. In: International Cryptographic Module Conference (2013)
4. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Advances in Cryptology — CRYPTO '97. LNCS, vol. 1294, pp. 513–525. Springer (1997)
5. Canright, D.: A very compact S-Box for AES. In: Cryptographic Hardware and Embedded Systems — CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer (2005)
6. Cassiers, G., Standaert, F.X.: Trivially and efficiently composing masked gadgets with probe isolating non-interference. IEEE Transactions on Information Forensics and Security **15**, 2542–2555 (2020)
7. De Cnudde, T., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., Rijmen, V.: Masking AES with d+1 shares in hardware. In: Bilgin, B., Nikova, S., Rijmen, V. (eds.) Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016. p. 43. ACM (2016)
8. De Meyer, L., Arribas, V., Nikova, S., Nikov, V., Rijmen, V.: M&M: Masks and macs against physical attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(1), 25–50 (2019)
9. Groß, H., Mangard, S., Korak, T.: Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In: Proceedings of the ACM Workshop on Theory of Implementation Security. p. 3. ACM (2016)
10. Hirata, H., Miyahara, D., Arribas, V., Li, Y., Miura, N., Nicova, S., Sakiyama, K.: All you need is fault: Zero-value attacks on AES and a new λ-detection M&M. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(1), 133—156 (2023)
11. Ishai, Y., Sahai, A., Wagner, D.A.: Private circuits: Securing hardware against probing attacks. In: Advances in Cryptology — CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer (2003)

12. Knichel, D., Sasdrich, P., Moradi, A.: SILVER - statistical independence and leakage verification. In: Advances in Cryptology — ASIACRYPT 2020. pp. 787—816 (2020)
13. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Advances in Cryptology — CRYPTO '96. LNCS, vol. 1109, pp. 104–113. Springer (1996)
14. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology — CRYPTO '99. LNCS, vol. 1666, pp. 388–397. Springer (1999)
15. Lomné, V., Roche, T., Thillard, A.: On the need of randomness in fault attack countermeasures - application to AES. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography. pp. 85–94 (2012)
16. NANGATE. The Nangate 45nm Open Cell Library. https://www.nangate.com/
17. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Information and Communications Security. LNCS, vol. 4307, pp. 529–545. Springer (2006)
18. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Advances in Cryptology — CRYPTO 2015. LNCS, vol. 9215, pp. 764–783. Springer (2015)
19. Reparaz, O., Meyer, L.D., Bilgin, B., Arribas, V., Nikova, S., Nikov, V., Smart, N.P.: CAPA: the spirit of beaver against physical attacks **10991**, 121–151 (2018)
20. Schneider, T., Moradi, A.: Leakage assessment methodology - A clear roadmap for side-channel evaluations. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems — CHES 2015. LNCS, vol. 9293, pp. 495–513. Springer (2015)
21. Schneider, T., Moradi, A., Güneysu, T.: Parti: Towards combined hardware countermeasures against side-channel and fault-injection attacks. In: Bilgin, B., Nikova, S., Rijmen, V. (eds.) Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016. p. 39. ACM (2016)
22. Wolf, C.: Yosys Open SYnthesis Suite. https://yosyshq.net/yosys/