# On Efficient and Secure Code-based Masking: A Pragmatic Evaluation

Qianmei Wu, Wei Cheng, Sylvain Guilley, Fan Zhang, Wei Fu

IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 3, pp. 192–222. DOI:10.46586/tches.v2022.i3.192-222

# On Efficient and Secure Code-based Masking: A Pragmatic Evaluation

Qianmei Wu[1,4], Wei Cheng[2,3*], Sylvain Guilley[3,2],

Fan Zhang[1,5,6*] and Wei Fu[7]

[1] School of Cyber Science and Technology, College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027, China,
{qianmei,fanzhang}@zju.edu.cn
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, 91120, Palaiseau, France
wei.cheng@telecom-paris.fr
[3] Secure-IC S.A.S., 75015, Paris, France,
sylvain.guilley@secure-ic.com
[4] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing, 100878, China
[5] Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province
[6] Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies, Hangzhou, China, 310027
[7] Ant Group, Hangzhou, 310013, China

**Abstract.** Code-based masking is a highly generalized type of masking schemes, which can be instantiated into specific cases by assigning different encoders. It captivates by its side-channel resistance against higher-order attacks and the potential to withstand fault injection attacks. However, similar to other algebraically-involved masking schemes, code-based masking is also burdened with expensive computational overhead. To mitigate such cost and make it efficient, we contribute to several improvements to the original scheme proposed by Wang et al. in TCHES 2020. Specifically, we devise a computationally friendly encoder and accordingly accelerate masked gadgets to leverage efficient implementations. In addition, we highlight that the amortization technique introduced by Wang et al. does not always lead to efficient implementations as expected, but actually decreases the efficiency in some cases.

From the perspective of practical security, we carry out an extensive evaluation of the concrete security of code-based masking in the real world. On one hand, we select three representative variations of code-based masking as targets for an extensive evaluation. On the other hand, we aim at security assessment of both encoding and computations to investigate whether the state-of-the-art computational framework for code-based masking reaches the security of the corresponding encoding. By leveraging both leakage assessment tool and side-channel attacks, we verify the existence of "security order amplification" in practice and validate the reliability of the leakage quantification method proposed by Cheng et al. in TCHES 2021. In addition, we also study the security decrease caused by the "cost amortization" technique and redundancy of code-based masking. We identify a security bottleneck in the gadgets computations which limits the whole masked implementation. To the best of our knowledge, this is the first time that allows us to narrow down the gap between the theoretical security order under the probing model (sometimes with simulation experiments) and the concrete side-channel security level of protected implementations by code-based masking in practice.

**Keywords:** Side-Channel Analysis · Code-based Masking · Efficient Implementation · Security Evaluation · Inner Product Masking · Redundant Sharing

*The corresponding author.


Received: 2022-01-15 Accepted: 2022-03-15 Published: 2022-06-08

# 1 Introduction

Side-channel attacks are nowadays commonly considered a significant threat to the cryptographic devices since various unintentional side-channel leakage can be captured and exploited by a motivated adversary. To thwart such threat, numerous protection mechanisms have been proposed, among which *masking* scheme prevails due to a formal and sound theoretical foundation [CJRR99, ISW03, PR13]. Masking splits the sensitive variable into $n$ shares, enabling any $d$ of them to be independent of the protected sensitive variable, for any $d < n$. The mapping method from sensitive variable(s) to shares is called *encoding function*. Furthermore, masking schemes also address the masked computations on those split shares to guarantee an ultimate correct calculation result. These masked computations are called *private computations*. Thus a complete masking scheme encompasses a secure encoding function and a strategy to perform private computations.

Boolean masking (BM) is one of the simplest masking schemes, whose encoding function utilizes the simple XOR operation. One line of research devotes to replacing the simple XOR operation with higher algebraic ones, since the latter is regarded to be capable of reducing information leakage and thus enhancing the side-channel resistance. In this direction, a variety of masking schemes emerge such as multiplicative masking [GT02], affine masking [FMPR10], polynomial masking [PR11, GM11], inner product masking (IPM) [BFG15, BFG+17] and direct sum masking (DSM) [BCC+14, PGS+17]. Interestingly, coding theory has also been introduced in this field and radiates new vitality [BCC+14, PGS+17, WMCS20, CGC+21]. More recently, a more general case called code-based masking appears, which covers BM, IPM, DSM, polynomial masking and so on. It utilizes a unified coding form $Z = X\mathbf{G} + Y\mathbf{H}$ for encoding, where $Z \in \mathbb{F}_q^n$, $X \in \mathbb{F}_q^k$ and $Y \in \mathbb{F}_q^m$ denote masked variables, sensitive variables and random masks, respectively; $\mathbf{G}$ (resp., $\mathbf{H}$) is a generator matrix of the code $\mathcal{C}$ (resp., $\mathcal{D}$). Code-based masking is deemed to decrease the information leakage by increasing the "statistical security order" (or bit-probing security order) of masked implementations [WSY+16, PGS+17, CGC+20]. Moreover, the underlying error-correction capability enables code-based masking to have potential against fault injection analysis for the encoded variables.

Code-based masking can be more secure but suffers from higher computational overhead, actually originating from the positive and negative aspects of the high-algebraic structure. In case of IPM, prior works [BFG15] and [BFG+17] make great efforts at cost reduction for IPM. The results are considerable but still not as efficient as BM. Recent research [WMCS20] has proposed a generic computational solution for code-based masking and devised the "cost amortization" technique to speed up the masked computations. Equipped with the cost amortization technique, the overhead of code-based masking considering both bilinear multiplications (multiplication of two non-constant values) and randomness can be reduced considerably given large enough number of amortized sensitive variables (say $k$) and security order (say $d$). However, the performance of code-based masking is still much lower than that of Boolean masking for commonly used choices of $k$ and $d$ [WMCS20, WGS+20]. Therefore, the secure and efficient implementation of code-based masking schemes is still an open challenge.

Plenty of researches concentrate on evaluating the side-channel resistance of code-based masking schemes under theoretical models or by simulated experiments [BFG15, BCC+14, WMCS20, CGC+21, CS21]. To the best of our knowledge, there have been few investigations regarding the practical security of high-algebraic masked implementations when confronting the actual side-channel attacks, leaving a huge gap in this field. One reason might be that security level in practice is a complicated outcome interacted by various factors involving the theoretical scheme, implementation strategy, the specific platform, the execution environment and so on. Moreover, the associated security evaluation is also a tough task since it significantly relies on the expertise of attackers or saying evaluators. At the very least, capturing the observations of protected implementations and

the related data processing themselves are already time-consuming and labor-intensive works. Regarding practical side-channel evaluation, Balasch et al. [BFG+17] have collected the actual acquisitions and utilized Test Vector Leakage Assessment (TVLA) [GGJR+11] to analyze the leakage behavior of BM and IPM implementations. Besides, a sound and extensive evaluation of high-algebraic masking schemes (including code-based masking) still lacks with respect to practical security assessed by side-channel attacks.

**Our Contributions.** In this work, we aim at a pragmatic evaluation of code-based masking in real-world implementations. Our contributions are summarized as follows:

- We devise the first complete and generic implementation (in the form of improved mathematical equations and algorithms, and also of assembly code) of code-based masking in protecting AES-128.
- We provide a practical validation of "security order amplification" in code-based masking, which fills the gap between theory and practice.
- We confirm the reliability of a coding-theoretic framework which allows us to enhance the concrete side-channel resistance level of generic encoders in practice.
- Although the "cost amortization" technique seems promising, we illustrate a security loss because of amortization from a security perspective.
- We demonstrate that the redundancy in sharing usually leads to a security decrease in the sense of practical side-channel attacks.
- We identify a structural security bottleneck that ruins the security order amplification in code-based masking.

In particular, we highlight that the last three points give rise to challenges in the practical use of the probing model: the same security orders in theory may result in distinct concrete security levels in practice. That is, new practice-relevant models should be developed for more accurate side-channel evaluations. Moreover, in accordance with TCHES submission policy, our implementations will be available afterward on Github and for further evaluation as TCHES Artifacts.

## 2 Preliminaries

We denote the finite field of order $q$ as $\mathbb{F}_q$ and the field addition operation as $\oplus$. And we denote with $[n]$ the set of integers from 1 to $n$ (both included). Let calligraphies (e.g., $\mathcal{C}$) denote the linear code. We utilize bold lower cases (e.g., $\mathbf{x}$) to represent the vectors over $\mathbb{F}_q^{|\mathbf{x}|}$, where $|\mathbf{x}|$ defines the length of vector. The notation $\odot$ denotes the element-wise multiplication between two vectors $\mathbf{x}$ and $\mathbf{y}$ over $\mathbb{F}_q^n$, that is $\mathbf{x} \odot \mathbf{y} = (\mathbf{x}_1\mathbf{y}_1, \cdots, \mathbf{x}_n\mathbf{y}_n)$. We use $\mathbf{e}_i$ to denote a canonical vector: its $i^{th}$ element is 1 and all of its other elements are 0. Let bold capital letters (e.g., $\mathbf{A}$) represent matrices over $\mathbb{F}_q^{r\times c}$, constructed with $r$ rows and $c$ columns. $\mathbf{A}[i, *]$ (resp., $\mathbf{A}[*, i]$) denotes the $i^{th}$ row (resp., column) of $\mathbf{A}$, and $\mathbf{A}[i : j, *]$ (resp., $\mathbf{A}[*, i : j]$) denotes the matrix made up of the $i^{th}$ to $j^{th}$ rows (resp., columns) of $\mathbf{A}$. The symbols $\mathbf{A}^{-1}$ and $\mathbf{A}^T$ denote the (generalized) inverse and transpose of $\mathbf{A}$, respectively. For two matrices $\mathbf{A}$ and $\mathbf{B}$, we denote their product as $\mathbf{A} \times \mathbf{B}$ (or in short $\mathbf{AB}$), and $[\mathbf{A}, \mathbf{B}]$ (resp., $[\mathbf{A}; \mathbf{B}]$) is the concatenation of columns (resp., rows) of $\mathbf{A}$ and $\mathbf{B}$. The notation $\otimes$ represents tensor product between two matrices.

In the remainder of this paper, we use $\mathbf{A}$ to represent the practical encoder (introduced in Section 3.1). Let $\mathbf{O}^{x\times y}$ and $\mathbf{I}_x$ be a zero matrix and an identity matrix over $\mathbb{F}_q^{x\times y}$ and $\mathbb{F}_q^{x\times x}$, respectively. We denote the $\hat{\mathbf{x}}$ as the codeword of $\mathbf{x}$ (namely $\hat{\mathbf{x}} = \mathbf{xA}$). Note that all indices in this paper start from 1 instead of 0.

In this paper, two kinds of security orders are involved, namely security orders at word-level and bit-level under the probing model [ISW03, PGS+17], that are defined in $\mathbb{F}_{2^8}$ and $\mathbb{F}_2$, respectively. Note that the latter is also the security order in the bounded moment model [BDF+17]. In addition, we use the following coding-theoretic properties.

**Definition 1** (Dual Distance [MS77] and Kissing Number [1] [SLC+21])**.** Considering a linear code $\mathcal{C}$, its dual distance $d_{\mathcal{C}}^{\perp}$ is the minimum Hamming weight $w_H(u)$ of nonzero $u \in \mathbb{F}^n$, such that $\sum_{c\in\mathcal{C}}(-1)^{c\cdot u} \neq 0$. The dual distance $d_{\mathcal{C}}^{\perp}$ of a linear code $\mathcal{C}$ coincides with the minimum distance $d_{\mathcal{C}^{\perp}}$ of the dual code $\mathcal{C}^{\perp}$. Accordingly, the kissing number $B_d$ is the number of codewords in $\mathcal{C}$ at minimum distance $d$ to the all-0 codeword: $B_d = |\{x \in \mathcal{C} \,|\, w_H(x) = d\}|$.

**Definition 2** (Adjusted Kissing Number [CGC+21])**.** Let $\mathcal{C}$, $\mathcal{D}$ denote two linear codes; their adjusted kissing number $B'_d$ is:

$$B'_d = |\{(x, y) \in (\mathcal{D}\backslash\mathcal{C})^2 \,|\, x + y \in \mathcal{C}, w_H(x) = w_H(y) = d\}|. \tag{1}$$

We use the above coding-theoretic properties mainly over $\mathbb{F}_2$ unless otherwise stated, and a linear code can be easily expanded from $\mathbb{F}_{2^8}$ into $\mathbb{F}_2$ by using the sub-field representation [MS77, CGC+21]. It is worth mentioning that the kissing number $B_{d_{\mathcal{D}}^{\perp}}$ plays an important role in indicating side-channel resistance of code-based masking, which is defined on the dual code $\mathcal{D}^{\perp}$. Similarly, the adjusted kissing number $B'_{d_{\mathcal{D}}^{\perp}}$ is defined on $\mathcal{C}^{\perp}$ and $\mathcal{D}^{\perp}$ in code-based masking [CGC+21]. The latter is degraded to the former in non-redundant cases where we shall have $\mathcal{C}^{\perp} \cap \mathcal{D}^{\perp} = \{0\}$, e.g., in IPM and DSM.

## 3 Efficient Construction of Code-based Masking

In this part, we first give a brief introduction of the generic encoder and private computations for code-based masking devised in [WMCS20]. Then a detailed illustration of our improved construction will be exhibited and it is followed by comparisons regarding computational complexity and discussions about the "cost amortization" method. Finally, we have investigated efficient implementations tailored to a specific platform.

So far, there is only one valid construction of computational framework proposed by Wang et al. [WMCS20] for code-based masking in a general sense. In particular, Wang et al. [WMCS20] propose a generic encoder for code-based masking and three masked gadgets for its private computations. The generic encoder is actually a generator matrix $\mathbf{A}_{\text{gene}}$ over $\mathbb{F}_q^{(k+m)\times n}$ which consists of two parts: the upper part $\mathbf{G}$ over $\mathbb{F}_q^{k\times n}$ relating to the sensitive variables and the lower part $\mathbf{H}$ over $\mathbb{F}_q^{m\times n}$ corresponding to the random masks. It requires that $\mathbf{G}$ and $\mathbf{H}$ are both full-rank with $\mathcal{C}_{\mathbf{G}} \cap \mathcal{C}_{\mathbf{H}} = \{\mathbf{0}\}$ [WMCS20]. The masked gadgets address the element-wise multiplication operations and linear transformations in the masked domain. The three gadgets are different but basically follow a similar procedure. Such procedure should be fed with codeword(s) over $\mathbb{F}_q^n$ and transforms the input vectors into a matrix over $\mathbb{F}_q^{n\times n}$ by an outer product at the very beginning. Hence we can regard the masked computations throughout the gadgets as matrix operations. Eventually, an output vector over $\mathbb{F}^n$ can be obtained, which is actually the codeword of the unmasked output. A special step during such procedure lies in a back-and-forth switch between the code-based masking and the additive sharing [2], which is the core for both multiplication and linear functions in the current computational framework, though usually at cost of practical security losses (which will be assessed in Section 4).

[1] The kissing number is a geometric notion originally defined for lattices and sphere packings. Recently, it is adapted for the linear codes over finite fields [SLC+21].

[2] In this paper, we use equivalently "Boolean masking" and "additive sharing", e.g., to keep consistency with descriptions in [WMCS20].

## 3.1 Practical Construction of Generic Encoder

In order to improve the performance of code-based masking [WMCS20], we devise a relatively fixed construction for the generic encoder. The basic idea is to set as many entries as possible to 0, rendering the generator matrix to be more sparse. Sparsity implies the corresponding 0-involved multiplications can be omitted, so as to mitigate computational overhead. Our construction remains the concatenation of two matrices $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ and $\mathbf{H} \in \mathbb{F}_q^{m\times n}$, that is $\mathbf{A} = [\mathbf{G}; \mathbf{H}]$. It is illustrated in Equation 2 below.

$$\mathbf{A} = \begin{pmatrix} 1 & & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & \ddots & & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ a_{11} & \cdots & a_{1k} & a_{1(k+1)} & \cdots & a_{1(n-m)} & 1 & & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \ddots & \\ a_{m1} & \cdots & a_{mk} & a_{m(k+1)} & \cdots & a_{m(n-m)} & 0 & & 1 \end{pmatrix}. \tag{2}$$

We consider the matrices $\mathbf{G} = [\mathbf{I}_k, \mathbf{O}^{k\times(n-k)}]$ (also suggested in [WMCS20]), and $\mathbf{H} = [\mathbf{R}, \mathbf{I}_m]$, where $\mathbf{R}$ is an $m \times (n-m)$ matrix whose entries $a_{ij} \in \mathbb{F}_q$ (for $1 \le i \le m$,$1 \le j \le (n-m)$) remain free for practical designers, and that we explore in this article. The two identity matrices $\mathbf{I}_k$ and $\mathbf{I}_m$ ensure that $\mathbf{G}$ and $\mathbf{H}$ are both full-rank, satisfying $\mathcal{C}_\mathbf{G} \cap \mathcal{C}_\mathbf{H} = \{\mathbf{0}\}$ as well. Therefore, $\mathbf{A} = [\mathbf{G}; \mathbf{H}]$ corresponds to a valid generic encoder, called the *practical encoder* $\mathbf{A}$ in this paper. We highlight that the generic encoder (proposed in [WMCS20]) could be set in the form of $\mathbf{A}$ for better performance. Since on the one hand, any generator matrices of the linear code can be transformed into systematic forms, therefore any variations of the generic encoder shall be equivalent to the instances of $\mathbf{A}$. On the other hand, as many as possible entries of $\mathbf{A}$ are set to 0 or 1, hence in contrast with the generic encoder, it is more computationally friendly as massive related (both 0-involved and 1-involved) multiplications can be omitted. Moreover, the practical encoder retains the ability to apply the "cost amortization" technique [WMCS20].

## 3.2 Improving Masked Gadgets

Our main strategy to accelerate the masked gadgets is to eliminate as many as possible field multiplications during gadget computations via the sparsity of the practical encoder [3]. In addition, we also tune up and reduce the redundant parts in the masked gadgets. We should claim that our improvement will not cause a loss of security in theory, but only mitigate the computational overhead. As is introduced above, the three masked gadgets (multiplication Gadget, L Gadget and Ls Gadget) proposed in [WMCS20] actually share a similar framework (straightforwardly applied in multiplication Gadget, while L Gadget and Ls Gadget have slight variations). Therefore, here we only consider the common framework in multiplication Gadget. Our improvement encompasses three parts as follows.

**Part 1: Simplifying Refresh Variables $\hat{\mathbf{R}}_1$ and $\hat{\mathbf{R}}_2$.** In the original framework, there are two refreshing operations by two well-constructed matrices $\hat{\mathbf{R}}_1$ and $\hat{\mathbf{R}}_2$. Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be two matrices uniformly distributed over $\mathbb{F}_q^{n\times m}$, $\hat{\mathbf{R}}_1$ can be computed as $\hat{\mathbf{R}}_1 = ([\mathbf{O}^{n\times k}, \mathbf{R}_1]\mathbf{A})^T$ and $\hat{\mathbf{R}}_2 = [\mathbf{O}^{n\times k}, \mathbf{R}_2]\mathbf{A}$. Explicitly, the existence of matrix $\mathbf{O}^{n\times k}$ prevents the upper $k \times n$ part $\mathbf{G}$ of matrix $\mathbf{A}$ from participating in the computations. Hence the construction of $\hat{\mathbf{R}}_1$ and $\hat{\mathbf{R}}_2$ can be simplified by removing the needless product of $\mathbf{O}^{n\times k}$ and $\mathbf{G}^{k\times n}$. Therefore, $\hat{\mathbf{R}}_1$ and $\hat{\mathbf{R}}_2$ can be improved as follows in Equation 3.

$$\hat{\mathbf{R}}_1 = (\mathbf{R}_1 \times \mathbf{H})^T, \qquad \hat{\mathbf{R}}_2 \quad = \mathbf{R}_2 \times \mathbf{H}. \tag{3}$$

[3] It has been noted in related works that *genetic algorithms* can help achieve sparsity beyond plain systematic forms [CDD+15]. Such technique is hard to formalize and hence is left open as future work.

**Part 2: Reducing Internal Computation.** As mentioned above, there exists a special transformation from code-based masking to additive sharing in the original framework. Here we focus on this transformation. The input is a matrix $\mathbf{S} \in \mathbb{F}_q^{n\times n}$ and the transformation is conducted by multiplying each row $\mathbf{S}[i,*]$ of $\mathbf{S}$ with a corresponding pre-computed matrix $\mathbf{M}_i$ over $\mathbb{F}_q^{n\times(k+m)}$, resulting in an $n \times (k+m)$ matrix $\mathbf{T}$, for $i \in [n]$. To enable matrix $\mathbf{T}$ to be the additive sharings, $\mathbf{M}_i$ is defined as $\mathbf{M}_i \stackrel{def}{=} (\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}})\mathbf{E}$, where $\tilde{\mathbf{A}} \stackrel{def}{=} [\mathbf{A}^{-1}[*,1:k], \mathbf{O}^{n\times m}]$ and $\mathbf{E}$ over $\mathbb{F}^{(k+m)^2\times(k+m)}$ is the concatenation of the $(k+m)^2$-length $e_i^T$ for $i \in \{j(k+m)+j+1 | 0 \le j < k+m\}$ [WMCS20]. However, we find that the last $n \times m$ sub-matrix of $\mathbf{M}_i$ is always a zero matrix so that can be eliminated, and thus we set $\mathbf{M}_i = \mathbf{M}_i^*[*,1:k]$ and $\mathbf{M}_i^* \stackrel{def}{=} (\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}})\mathbf{E}$. As a consequence, the resulting matrix $\mathbf{T}$ turns into an $n \times k$ matrix. The improved process for this transformation is illustrated in Equation 4 below.

$$\begin{pmatrix} \mathbf{S}[1,1] & \cdots & \mathbf{S}[1,n] \\ \vdots & \ddots & \vdots \\ \mathbf{S}[n,1] & \cdots & \mathbf{S}[n,n] \end{pmatrix} \rightarrow \begin{matrix} \times\mathbf{M}_1 \\ \vdots \\ \times\mathbf{M}_n \end{matrix} \rightarrow \begin{pmatrix} \mathbf{T}[1,1] & \cdots & \mathbf{T}[1,k] \\ \vdots & \ddots & \vdots \\ \mathbf{T}[n,1] & \cdots & \mathbf{T}[n,k] \end{pmatrix}. \tag{4}$$

Regarding the correctness of the reduced computation, we have the following lemma.

**Lemma 1.** *The reduced computation of* $\mathbf{T}$ *in Equation 4 is functionally equivalent to the original one in [WMCS20], hence the correctness is kept unchanged.*

*Proof.* Following the above computational procedure, we have $\tilde{\mathbf{A}} \stackrel{def}{=} [\mathbf{A}^{-1}[*,1:k], \mathbf{O}^{n\times m}]$ and $\mathbf{E} \in \mathbb{F}^{(k+m)^2\times(k+m)}$ as the concatenation of the $(k+m)^2$-length $e_i^T$ for $i \in \{j(k+m)+j+1 | 0 \le j < k+m\}$. Therefore, multiplying with $\mathbf{O}^{n\times m}$ gives the right part sub-matrix of $\mathbf{M}_i$ being all zeros. Similarly, after multiplying with $\mathbf{M}_i$ returns the right part sub-matrix of $\mathbf{T}$ being $\mathbf{O}^{n\times m}$. Removing above all-zero sub-matrices yields Equation 4. □

**Part 3: Removing Re-encoding.** After the transformation from code-based masking to additive sharing, the matrix $\mathbf{T}$ (the original $n \times (k+m)$ one in [WMCS20]) is multiplied by generator matrix $\mathbf{A}_{\text{gene}}$ for re-encoding (the conversion back from additive sharing to code-based masking). However, after being improved by the second part, the matrix $\mathbf{T}$ has only $k$ columns so that only the upper part $\mathbf{G}$ (over $\mathbb{F}_q^{k\times n}$) of $\mathbf{A}$ involves in the product. Recall that $\mathbf{G} = [\mathbf{I}_k, \mathbf{O}^{k\times(n-k)}]$, hence the re-encoding process is equivalent to $\mathbf{T}^{n\times k}$ concatenated with an $n \times (n-k)$ zero matrix (namely $\mathbf{T}^{n\times k}\mathbf{G}^{k\times n} = [\mathbf{T}^{n\times k}, \mathbf{O}^{n\times(n-k)}]$). As a result, a product originally between two matrices $\mathbb{F}_q^{n\times(k+m)} \times \mathbb{F}_q^{(k+m)\times n}$ is simplified to be an easy and efficient concatenation operation.

**Lemma 2.** *Given that* $\mathbf{A}_{\text{gene}}$ *in code-based masking takes the form of Equation 2, then removing re-encoding by* $\mathbf{A}$ *is functionally equivalent to the original algorithm in [WMCS20].*

*Proof.* Since the upper sub-matrix of $\mathbf{A}$ is $\mathbf{G} = [\mathbf{I}_k, \mathbf{O}^{k\times(n-k)}]$ and the output of previous step is $\mathbf{T}' = [\mathbf{T}, \mathbf{O}^{n\times m}]$, then $\mathbf{T}'\mathbf{A} = \mathbf{T}\mathbf{G} = [\mathbf{T}, \mathbf{O}^{n\times m}] = \mathbf{T}'$. This concludes the proof. □

Here we present the detailed construction in Algorithm 1 for the improved framework (or saying multiplication Gadget). From a security perspective, our improved algorithm has the same security order as in [WMCS20] under the word-level probing model. That is, if $\mathbf{A}$ is a $d$th-order secure encoder, then the improved multiplication Gadget is $d$th-order secure as well. The proof of [WMCS20] still applies. By applying the above improvements, L Gadget and Ls Gadget (detailed in Appendix A) which follow the similar procedure have also been sped up with no loss of security.

Since the field multiplication is usually the most costly part (with no native instruction) in the masked implementations, the above improvements shall boost the performance of

code-based masking from the algorithmic level, especially when an appropriate encoder $\mathbf{A}$ is chosen.

**Algorithm 1:** Improved Multiplication Gadget

**Input:** Codewords $\hat{\mathbf{x}} = \mathbf{xA}$ and $\hat{\mathbf{y}} = \mathbf{yA}$ of $\hat{\mathbf{x}}, \hat{\mathbf{y}} \in \mathbb{F}_q^n$ and $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^k$
**Output:** $\hat{\mathbf{z}} \in \mathbb{F}_q^n$ such that $\mathbf{z} = \mathbf{x} \odot \mathbf{y}$

Initialize $\mathbf{R}_1$ and $\mathbf{R}_2$ uniformly over $\mathbb{F}_q^{n \times m}$
$\hat{\mathbf{R}}_1 = (\mathbf{R}_1\mathbf{H})^T$, $\hat{\mathbf{R}}_2 = \mathbf{R}_2\mathbf{H}$
**for** $i = 1$ **to** $n$ **do**
  **for** $j = 1$ **to** $n$ **do**
    $\mathbf{S}[i, j] = \hat{\mathbf{x}}[i]\hat{\mathbf{y}}[j] \oplus \hat{\mathbf{R}}_1[i, j]$
  **end**
**end**
$\mathbf{M}_i = \mathbf{M}_i^*[*, 1 : k]$, where $\mathbf{M}_i^* = (\tilde{\mathbf{A}}[i, *] \otimes \tilde{\mathbf{A}})\mathbf{E}$ and $\tilde{\mathbf{A}} \stackrel{def}{=} [\mathbf{A}^{-1}[*, 1 : k], \mathbf{O}^{n \times m}]$
**for** $i = 1$ **to** $n$ **do**
  $\mathbf{T}[i, *] = \mathbf{S}[i, *] \times \mathbf{M}_i$
**end**
$\mathbf{K} = [\mathbf{T}, \mathbf{O}^{n \times (n-k)}] \oplus \hat{\mathbf{R}}_2$
**for** $j = 1$ **to** $n$ **do**
  $\hat{\mathbf{z}}[j] = \sum_{i=1}^{n} \mathbf{K}[i, j]$
**end**
**return** $\hat{\mathbf{z}}$

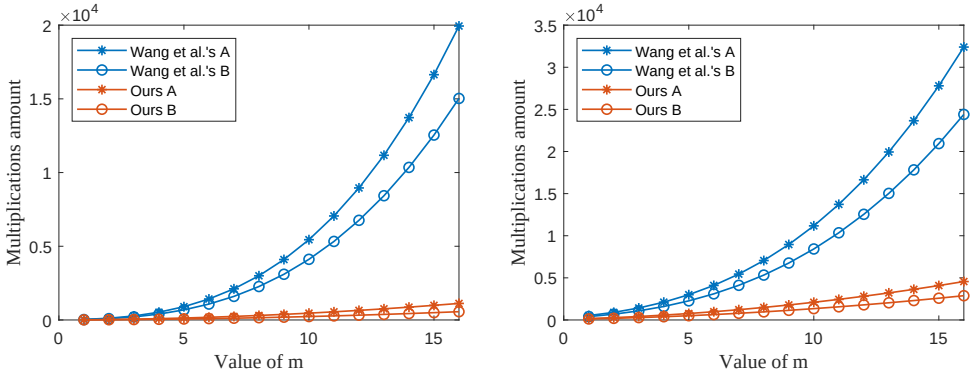
## 3.3 Computational Complexity and Comparisons

In this section, we quantify the computational complexity of our improved framework and present comparisons with other masking schemes. Here we concentrate on the field multiplications as it is usually the most costly part. Firstly, to exhibit the improvement explicitly, we conduct a comparison with the original construction in [WMCS20]. The amount of multiplications for each gadget is reported in Table 1 (the detailed encoding and decoding are attached in Appendix B). Note that Ls Gadget is a multiple of L Gadget and thus the computational cost is also a constant multiple, so it is omitted in the table. Additionally, we also illustrate the trend of the multiplications quantity with increasing $m$ and $k$ in Figure 1. For the sake of brevity, we set $n = k + m$ and only showcase two representative cases for $k$, that is $k = 1$ and $k = 4$. In Figure 1, "A" (resp., "B") indicates the multiplication Gadget (resp., L Gadget), marked by the symbol $*$ (resp., small circles).

**Table 1:** Comparison of the number of field multiplications in different components.

| Component | Scheme in [WMCS20] | Our Improved Scheme |
|---|---|---|
| Encoding | $n(k+m)$ | $m(n-m)$ |
| Multiplication Gadget | $n^2(4k+4m+1)$ | $n^2(k+1)+2nm(n-m)$ |
| L Gadget | $n^2(3k+3m+1)$ | $n^2k+nm(n-m)$ |
| Decoding | $nk$ | $k(n-k)$ |

Notably, it can be explicitly demonstrated from Table 1 and Figure 1 that our improvement is significant, as the computational complexity of our scheme is much lower than that of Wang et al. in any case and the disparity becomes larger with increasing $k$ and




(a) $k = 1$ (b) $k = 4$

**Figure 1:** Comparison of the number of multiplications with increasing $m$ for the original masked gadgets and our improved ones.

$m$. Furthermore, we also supplement the performance comparison regarding clock cycle counts of complete cryptographic implementation in Appendix C for practical verification.

In addition to the longitudinal comparison, we also supplement an extra comparison with an efficient BM scheme, say packed BM, which utilizes the "cost amortization" technique. In fact, the same idea of amortization for mitigating overhead is presented in [WMCS20] and can be applied in our scheme as well. For a fair comparison, we set $n = k + m$ and focus on multiplication gadget in Table 2. As shown in Table 2, the packed BM consumes less computational resources for all components compared. The reasons are, on the one hand, the packed BM scheme aims at Boolean masking only for efficiency while our improved construction is generic for various code-based masking schemes. On the other hand, the idea of "cost amortization" (processing multiple sensitive variables in parallel in the masked domain) for efficiency may be adaptable for that well-designed BM scheme, but fails to reduce overhead (recall that our strategy to mitigate cost is eliminating as many field multiplications as possible) in our scheme, which will be elaborated in Section 3.4. Moreover, we highlight that packed BM is a special case of our improved scheme by instantiating the practical encoder $\mathbf{A}$ as devised in [WGS+20] from an encoding perspective.

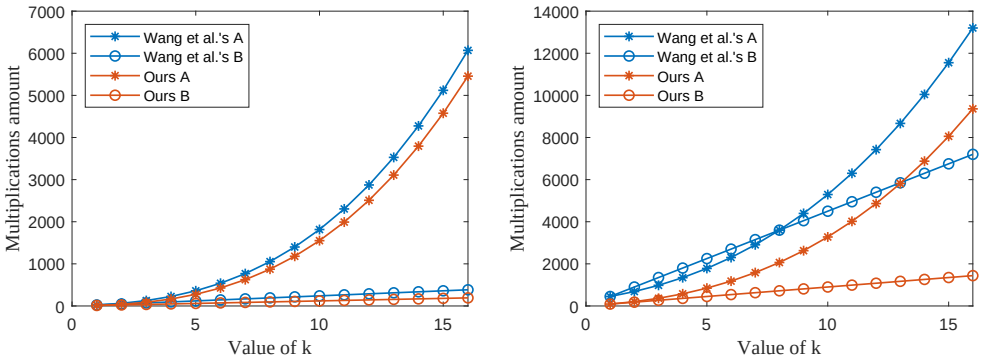
**Table 2:** Computational complexity of field multiplications for components.

| Component | Packed BM in [WGS+20] | Our Improved Scheme |
|---|---|---|
| Encoding | 0 | $mk$ |
| Multiplication Gadget | $(5m^2 + 4m + 2)k + m^2$ | $(3k^2 + 3mk + m + k)(m + k)$ |
| Decoding | 0 | $mk$ |

## 3.4 Discussion about "Cost Amortization"

The concept of "cost amortization" is first proposed by [WMCS20] to improve the performance of code-based masking. That is, $k$ (for $k > 1$) sensitive variables are encoded into one codeword, then all computations performed on the codeword are equivalent to parallel operations on those $k$ initial variables, resulting in packed operations and possible cost amortization. It is shown in [WMCS20] to require less bilinear multiplications and randomness given sufficiently large $k$ and $m$ compared to [ISW03]. However, there is merely no improvement in performance for small $k$ and $m$, since the "cost amortization" technique involves more internal computations. Indeed, the codeword integrating the $k$ sensitive variables should enter into masked gadgets and the beginning operation is an outer product, which converts the input vector(s) over $\mathbb{F}_q^n$ into an $n \times n$ matrix. Unfortunately, the matrix state and related computations will continue through the whole procedure until

the end. It implies that if the length of the input vector (actually $n$) becomes longer, the dimension of the internal matrices will increase accordingly, resulting in a nearly quadratic increase at cost. However, if we set $k = 1$ and perform the gadget computations in a sequential fashion, it only leads to a linear growth at cost. To demonstrate, we provide a straightforward comparison of the multiplication Gadget (fixing that $n = k + m$ for brevity) between the packed operation and the sequential calculation in Figure 2.




(a) $m = 1$ (b) $m = 4$

**Figure 2:** Comparison of the number of multiplications with increasing $k$ for the packed operation and serial computation.

Note that in Wang et al.'s case, we utilize the recommended instance of the generic encoder provided in [WMCS20], such that $\mathbf{G} = [\mathbf{I}_k, \mathbf{O}^{k\times m}]$ and $\mathbf{H}$ is the transpose of a Vandermonde matrix. In Figure 2, "A" marked by symbol $*$ represents the scheme exploiting packed operation, and "B" marked by small circles denotes $k = 1$ so as to perform the multiplication Gadget sequentially. We can discover that when $m$ is small, the packed operation actually shows a negative effect on mitigating the amount of multiplications. However, if $m$ increases, the cost amortization technique could be more efficient. Here in our case, the potential advantage of the packed operation vanishes. The reason is that, in Wang et al.'s case, the internal matrix keeps the same dimension $n$ (recall that $n = k + m$) throughout the overall computation of multiplication Gadget. That is, $k$ and $m$ jointly dominate the dimension length of the internal matrix. Moreover, $m$ is more involved in the multiplications (the corresponding amount is $n^2(k + 4m + 1)$). Hence if $m$ is large enough, the cost introduced by $k$ will be less significant. On the contrary, in our scheme, $k$ plays a more important role in the dimension of the internal matrix (recall that $\mathbf{T}$ is an $n \times k$ matrix) and $k$ is more involved in the multiplication (recall the computational complexity in Table 1). Hence, however large $m$ is, the quadratic overhead introduced by $k$ will always be higher than the linear case. To further illustrate it, we supplement the comparisons regarding actual clock cycles counts in Appendix D.

In summary, the computation involvement of $k$ and $m$ actually affect the feasibility and effectiveness of the "cost amortization" method. Therefore, not all instances of code-based masking are appropriate to utilize this technique from a computational point of view. Finally, it is also indicated in Figure 2 that whether compared with the serial computation or the packed operation of Wang et al.'s scheme (already accelerated by choice of $\mathbf{G}$), our improved scheme provides better performance from a computational perspective, which again validates our improvement.

## 3.5 Efficient Implementations

In addition to improve the theoretical computational framework, we have developed efficient implementations as well. To the best of our knowledge, this is the first attempt at practical implementation of code-based masking. Hence in order for a direct comparison with other

higher-order masking schemes, we set $k = 1$ employing $m = 1$ (with $n = 2$ shares) and $m = 2$ (with $n = 3$ shares), and apply our improved masking approach in the AES-128 implementations. Our target platform is LEGACY STM32F407 whose micro-controller is ARM Cortex-M4 running at 168 MHz. STM32F407 offers a 32-bit architecture and is equipped with 16 general-purpose registers, 512 KBytes of internal SRAM and 1024 MBytes of Flash memory. We select this device due to two reasons. Firstly it integrates the True Random Number Generator (TRNG) and hence it is capable of producing real random numbers. Secondly, its micro-controller ARM Cortex-M4 possesses an efficient and powerful instruction set. Particularly, it features inner barrel shifts, which implies a free cost of shift operations in some cases. For an aim of speed, our implementations are written in assembly code and some specific strategies are leveraged.

**Field Multiplication.** The implementation relates to the field addition and field multiplication in $\mathbb{F}_{2^8}$ (as our implementation is for AES-128, here we suppose $q$ to be $2^8$). Basically, the field addition can be easily addressed with a native XOR instruction, whereas the field multiplication is more challenging as there is no corresponding native instruction. We opt to utilize the Half-Table Multiplication [GR17], instead of the commonly used *log* and *alog* tables [DWBV+96]. The Half-Table method involves 2 look-ups in two $2^{12}$-sized tables. Although the memory size for tables storing increases compared to the *log* and *alog* table (involving two $2^8$-sized tables), it eliminates the conditional statement (check if any of the operands equals zero). Thanks to the elimination, the clock cycles required for one constant field multiplication can be reduced from 22 [BFG15] to 13 only.

The Half-Table method is based on the following equation [GR17]:

$$a \cdot b = b_h x^4 (a_h x^4 + al) + b_l (a_h x^4 + a_l) \mod p(x). \tag{5}$$

where $a_h$, $a_l$, $b_h$, $b_l$ are the 4-degree polynomials so that $a(x) = a_h x^4 + a_l$ and $b(x) = b_h x^4 + b_l$. Therefore, the above equation can be efficiently computed by tabulating the following functions [GR17]:

$$\begin{aligned} (a_h, a_l, b_h) &\mapsto b_h x^4 (a_h x^4 + a_l) \mod p(x), \\ (a_h, a_l, b_l) &\mapsto b_l (a_h x^4 + a_l) \mod p(x). \end{aligned} \tag{6}$$

As illustrated in [GR17], the advantage of inner barrel shifts in ARM instructions can be taken to gain the triplets efficiently. Precisely, for each table access, only two instructions are required at a minimum (more details referred to [GR17]). However, on our target platform (ARM Cortex-M4), the "LDR" instruction is unable to conduct all the types of inner barrel shifts and thus more instructions are required for one look-up. The concrete instructions to obtain the two triples are listed as follows:

```
LSL   $tmp1, $opA,    #4
EOR   $tmp2, $tmp1,   $opB,   LSR #4
LDRB  $res1, [$tab1,  $tmp2]
AND   $tmp2, $opB,    #0xF
EOR   $tmp2, $tmp1
LDRB  $res2, [$tab2,  $tmp2]
```

**Power Function and Affine Transformation.** For speed, we accelerate the power functions and affine transformation by look-up tables. Both power functions and affine transformation can be considered as the linear function (also the L function defined in [WMCS20]), and hence can be evaluated by L Gadget. As mentioned above, L Gadget almost follows the primary computational framework which has been improved in Section 3.2, thus it also contains a transformation from code-based masking to additive

sharings. This transformation is actually beneficial to L Gadget. Thanks to the property of linear functions, a linear transformation on the sensitive variable is consistent with the same linear transformation on its corresponding additive shares. When the conversion to additive sharings is carried out in L Gadget, the entries in the $k$ columns of matrix $\mathbf{T}$ (an $n \times k$ matrix) are actually the additive shares of the corresponding $k$ unmasked sensitive variables (that is $\mathbf{x}_j = \sum_{i=1}^{n} \mathbf{T}[i, j]$ for $1 \leq j \leq k$). We should underline that the input of L Gadget is actually the codeword $\hat{\mathbf{x}}$ (over $\mathbb{F}_{2^8}^n$) of $\mathbf{x}$ (over $\mathbb{F}_{2^8}^k$). As a consequence, the linear function $f$ of $\mathbf{x}_j$ can be computed as performing $f$ on the additive shares in $\mathbf{T}[*, j]$ independently. Since the power functions and affine transformation are both instances of linear transformations, and their application in our protected implementations relates to a bijective mapping from $\mathbb{F}_{2^8}$ to $\mathbb{F}_{2^8}$, the process of $f$ function performing on the additive shares independently can be accelerated by look-up tables. In fact, only four $2^8$-sized tables are required ($.^2$, $.^4$, $.^{16}$ and an affine function) for this part.

**AES Components.** AES is composed of four components: AddRoundKey, SubBytes, ShiftRows and MixColumns (refer to [DR02] for more details). Among them, AddRoundKey, ShiftRows and MixColumns are essentially linear transformations and thus can be directly evaluated by Ls Gadget (constructed in Algorithm 3). It becomes more complex for SubBytes transformation (also denoted as S-box), where a non-linear function is performed on each of the 16 internal states independently. The common method to compute S-box is using a combination of an inverse and an affine transformation both over $\mathbb{F}_2^8$. Rivain et al. [RP10] further propose to compute the inverse by a power function $x \mapsto x^{254}$, which can be decomposed into a quite efficient addition chain of several multiplications and power functions ($.^2$, $.^4$ and $.^{16}$). Therefore, with multiplication Gadget to compute multiplications and L Gadget for performing linear functions (containing both power functions and an affine transformation), SubBytes transformation can be computed in an efficient fashion.

By instantiating the practical encoder $\mathbf{A}$, code-based masking can be instantiated into specific masking schemes. Here we opt to implement BM and IPM by applying our improved scheme (constructed in Section 3). BM is acknowledged as an effective masking scheme with relatively low overhead, while IPM is a typical representation of high-algebraic masking schemes. Principally, both of them possess corresponding precedent efficient implementations. For BM and IPM with $n = 2$ shares and $n = 3$ shares, the corresponding generator matrices $\mathbf{A}$ are depicted in Table 3. Note that $L_2$ and $L_3$ represent the values of the public vector $L$ (of IPM) in the corresponding indices. In addition, the values of $\mathbf{A}[2, 1]$ and $\mathbf{A}[3, 1]$ in BM (corresponding to $L_2$ and $L_3$ respectively in IPM) are 1 and hence the involved multiplications can be removed. This is why the clock cycle counts (summarized in Table 4) are distinct for BM and IPM although with the same $k$, $m$ and $n$, which essentially affect the overhead. We should claim that our implementations all follow the same flow with constant time, thus the speed (evaluated by clock cycles counts) is independent of the input plaintexts and key. The measurement of clock cycles on the implementations for whole AES-128 encryption is summarized in Table 4.

**Table 3:** Various choices of generator matrix $\mathbf{A}$ over $\mathbb{F}_{2^8}^{n \times n}$ for BM and IPM, respectively.

| | BM, $n = 2$ | IPM, $n = 2$ | BM, $n = 3$ | IPM, $n = 3$ |
|---|---|---|---|---|
| $\mathbf{A}$ | $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 \\ L_2 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 \\ L_2 & 1 & 0 \\ L_3 & 0 & 1 \end{pmatrix}$ |

The clock cycle results in the second and third columns are taken directly from [BFG+17],

**Table 4:** Performance comparison by clock cycles for implementations.

| | BM in [BFG+17] | IPM in [BFG+17] | IPM(C) | Our BM | Our IPM |
|---|---|---|---|---|---|
| 2-share | 110569 | 157196 | 812314 | 155062 | 193765 |
| 3-share | 230221 | 372225 | 1730163 | 285025 | 334983 |

which are evaluated on an AVR ATMega163 [4] platform with 32 general-purpose registers. It should be pointed out that more registers for general-purpose implies less load and store instructions (memory access instructions usually require more clock cycles compared to general data processing instructions). To show the difference between the two platforms, we conduct a timing measurement on the implementation of [BFG+17] on our platform and the results are shown in the fourth column (indicated by "IPM(C)"). However this implementation is written in C code (not fully optimized). In the last two columns, we depict the clock cycles of our specific implementations. It is indicated from Table 4 that our implementations are generally less than 1.5 times as slow as the ones in [BFG+17]. Particularly, our implementation even presents a better performance in the case of IPM with 3 shares. Since those two different platforms both have advantages in reducing costs (e.g., AVR ATMega163 has more general-purpose registers, while STM32F407 features inner barrel shifts) and they are running at different operating frequencies, it is actually hard to draw a valid conclusion regarding the comparison. However, our intention is to provide clock cycle counts for code-based masking, which could promisingly be baselines for future research towards efficient implementations. We highlight that our implementations tailored to $n = 2$ shares and $n = 3$ shares in this section are aimed for speed only, and their side-channel resistance needs further in-depth inspections.

# 4 Practical Evaluations

The state-of-the-art investigations on code-based masking (excluding BM) can be clarified into three classes ranging from more theoretical to more practical analyses. First of all, most prior works [BFG15, WSY+16, PGS+17, BFG+17, WMCS20] have been devoted to designing theoretically secure masking gadgets against the formal adversarial model (e.g.the $d$-probing model [ISW03]). Second, several works [PGS+17, CG18, CGC+20, CGC+21, CS21] consider a coding-theoretic approach which connects the concrete security level of code-based masking (or some special instances) to coding properties. In particular, [CGC+21] quantifies the side-channel leakage from an information-theoretic perspective and shows that the dual distance and the (adjusted) kissing number are good indicators of side-channel resistance for code-based encoders. At last and in practice, to the best of our knowledge, only [BFG+17] considers leakage assessment by using t-test to check whether their IPM and BM implementations are leaking.

However, none of the above works consider the side-channel attacks against real-world implementations. In particular, some theoretical advantages like security order amplification essentially derived from encoding [WSY+16, PGS+17, CGC+21] have not been verified in practice, which leaves a huge gap between theory and practice and hinders the practical application of code-based masking. In this section, we intend to take a step forward to establish the relevance between theory and practice by evaluating three representative types of code-based masking schemes:

- **Non-redundant type** in Section 4.2: taking $n = k + m$ and $k = 1$, we focus on BM and IPM as special examples.

[4] AVR ATMega163 micro-controller has no TRNG, hence the random numbers for all types of implementations are provided externally and stored in memory for a fair comparison.

- **Packed type** in Section 4.3: taking $n = k + m$, $m = 1$ and $k \in \{1, 2, 4\}$, which utilizes cost amortization technique.
- **Redundant type** in Section 4.4: taking $n \geq k + m$, $k = m = 1$ and $n \in \{2, 3, 4\}$, where we show the impact of redundancy on side-channel security.

It is worth mentioning that implementations of BM and IPM evaluated in this section are instantiated from the general code-based masking [WMCS20] (e.g., taking the corresponding basic gadgets) and implemented in this work as the above, which differ from the IPM implementation proposed in [BFG+17].

## 4.1 Evaluation Strategy and Experimental Setup

First of all, we will give a brief summary of our evaluation objects and strategy. Then we detail the acquisition settings in our evaluation experiments.

**Evaluation Objects.** The core ingredients for a complete masking scheme are the encoding for randomizing the secrets and the masked computations manipulating the random shares. Since the latter involves more complicated factors, it is common that private computations sometimes fail to reach the security of the encoding function. Hence in order for an extensive evaluation on the practical security of code-based masking, we evaluate both encoding and gadgets computations, which though have been proved equally secure in word-level probing model [WMCS20]. On the one hand, to assess the side-channel resistance of encoding, we target the output of the first SubBytes transformation (or saying the output of L Gadget) in the first AES round. On the other hand, to evaluate the masked computations against side-channel attacks, we consider the seemingly worst-case scenarios by targeting the theoretical weakest part during gadgets computations, that is the matrix $\mathbf{T}$ of L Gadget (also during the first SubBytes in the first AES round).

As discussed above, a back-and-forth switch between code-based masking and BM exists during gadgets computations, possibly inclining to a security loss. And each column (for $k$ in total) of matrix $\mathbf{T}$ in L Gadget is exactly the additive sharing of the corresponding $k$ unmasked input sensitive variables. Even worse, no extra refreshing operation (e.g., XOR with $\hat{\mathbf{R}}_1$ in multiplication Gadget) is executed before such a switch in L Gadget, which is more likely to expose sensitive information. It is worth mentioning that: 1) the claim for the matrix $\mathbf{T}$ as the "weakest part" derives from the theoretical structural analysis since it degrades code-based masking to additive sharing, however it might not be the weakest part from an adversary's perspective, and 2) the matrix $\mathbf{T}$ is not the only "weakest" part in theory, instead the matrix $\mathbf{V}$ (constructed in Algorithm 2) possesses the same security level with $\mathbf{T}$.

**Evaluation Strategy.** The side-channel security of a cryptographic implementation can be assessed in two aspects: 1) How much sensitive information it leaks, and 2) How difficult for an adversary to extract the secret from those leaking information. To detect leakage, we leverage the most widely used leakage assessment tool in the literature, which is Test Vector Leakage Assessment (TVLA) [GGJR+11]. Concerning side-channel attacks, we utilize typical Correlation Power Analysis (CPA) [BCO04] and Template Attack (TA) [CRR02, RO04]. CPA is an effective non-profiled attack that is proven to be optimal if the side-channel measurements linearly depend on the hypothetical leakages [HRG14]. Whilst TA is profiled and regarded as the worst-case attack scenario. Regarding practical attacks, we consider two typical metrics, namely the Success Rate (SR) and Guessing Entropy (GE) [SMY09].

With respect to CPA, we leverage 2nd-order CPA in our evaluation since we mainly target first-order secure masking schemes. Specifically, we first select two sets of samples for each share, respectively, and combine them with a squared difference. Although the

centered product combination is demonstrated to be optimal in [PRB09], the squared difference combination performs better in our experimental scenarios. The combined samples are then correlated with hypothesis leakage under the Hamming weight leakage model for each subkey guess to obtain the attack results.

Regarding TA, in the profiling phase, firstly the collected measurements (totally $90,000$) are aligned by the static alignment method [MOP07]. Then $60,000$ measurements are used to build the templates, while other $30,000$ measurements are utilized to mount attacks. We build 256 Gaussian templates for each share in total, which include all the candidate values of one byte. As a result, we actually take the Hamming distance leakage out of account in our TA evaluations. As a result, we actually do not take transitions into account in our TA evaluations. Byte transitions can be taken into consideration if, for instance, $256^2 = 65536$ templates are profiled on the pair of (initial, final) values. During the attack phase, we leverage an adaption of the Gaussian mixture model as in [CMP18, CS21] by using real measurements. We choose one Point of Interest (POI) for each share and the selection strategy is to designate the one which has the largest and most consistent Signal-to-Noise Ratio (SNR), which is defined for each share as in [DFS15]. Since the masked implementations are running on the same device, and the acquisition environment and settings are the same (in each group of experiments), we assume the environment noise is constant. We therefore try to ensure that the SNR of all shares for different masked implementations in an evaluation are consistent with each other. As a consequence, it may not be the optimal attack for a single masked implementation since there may exist more informative sample points which are not exploited. But we nevertheless proceed this way for a fair comparison among all the protected implementations, since SNR has a significant impact on the attack results. As knowledgeable evaluators, we choose POIs directly by SNR for impartial evaluations, instead of selecting the most informative ones.

**Acquisition.** Our target platform is legacy STM32F407 which has been introduced in Section 3.5. We exploit the `arm-none-eabi-gcc` tool-chain to port our protected implementations (coded in assembly for speed-ups) of AES-128 to this platform. We should underline that our implementation for security evaluation is generic, which can be fed with any legal instance of the practical encoder $\mathbf{A}$ (it can be further optimized as in Section 3.5 for specific encoders). They are constant-time and thus independent of the inputs (plaintexts or key) as well. In the acquisition phase, Electromagnetic (EM) measurements are collected in a contactless fashion by placing a Riscure HP (High Precision) EM probe (SN152, with tip diameter 0.2mm) over the chip package. The probe can pick up EM fields with frequencies up to 4.5 GHz and has adjustable gain. Then the signals are sampled by a Keysight InfiniiVision DSOX3034T oscilloscope.

The acquisition stage for TVLA follows the approach of the non-specific fixed versus random test [GGJR+11, SM15]. Hence, we collect two sets of measurements: one is with fixed plaintexts, and the other is fed with random plaintexts drawn uniformly from $\mathbb{F}_{2^8}^{16}$. The two sets of measurements are obtained randomly interleaved. In total, we collect $100,000$ EM measurements (containing both fixed and random sets) for each TVLA test. Notably, each EM measurement covers the first 2.5 rounds of AES, which is actually a trade-off between the amount of data processing (shorter measurement implies less data) and the computational complexity of the executed encryption since two rounds of AES give full diffusion [BFG+17]. While for side-channel attacks, acquired EM measurements cover the first two SubBytes transformations in the first AES round.

## 4.2 Security Evaluation on Non-Redundant Type

In this section, we concentrate on the non-redundant case when $n = k + m$ and $k = 1$. This case corresponds to a masking scheme that protects one sensitive variable with

$m$ masks. Here we focus on IPM [5], which captivates by its commonly known "security order amplification" [WSY+16, PGS+17]. More precisely, the security order under the bit-probing model [PGS+17] of IPM can be much higher than its word-level security order under the probing model. In particular, Cheng et al. [CGC+20] further provide a sound theoretical explanation for this feature and show how the public vector $L$ of IPM significantly affects its concrete security level. They exploit two coding-theoretic parameters, the dual distance $d_{\mathcal{D}}^{\perp}$ and the kissing number $B_{d_{\mathcal{D}}^{\perp}}$ (see definitions in Section 2) of the code $\mathcal{D}$ to quantify the leakage of IPM, which is validated by simulation experiments. Guided by the theoretical derivations, we complement the last step to verify such a coding-theoretical leakage model by means of physical side-channel analysis, meanwhile gaining an insight into the practical security of code-based masking when $k = n + m$ and $k = 1$.

We consider four instances of IPM with $L_2 \in \{2, 3, 14, 23\}$ (recall Table 3), and they therefore possess the same 1st-order security in word-level probing model. Following the coding-theoretic approach proposed in [CGC+20], we obtain $d_{\mathcal{D}}^{\perp} \in \{2, 3, 3, 4\}$ and $B_{d_{\mathcal{D}}^{\perp}} \in \{5, 6, 1, 4\}$, respectively. The rationale is that the dual distance $d_{\mathcal{D}}^{\perp}$ indicates the concrete bit-probing security order and further smaller $B_{d_{\mathcal{D}}^{\perp}}$ implies a higher security level when the dual distances $d_{\mathcal{D}}^{\perp}$ are the same. To clarify the security level, we also add 1st-order and 2nd-order Boolean masking under the word-level probing model as baselines. Similarly, we have $d_{\mathcal{D}}^{\perp} = 2$, $B_{d_{\mathcal{D}}^{\perp}} = 8$ for 1st-order Boolean masking, and $d_{\mathcal{D}}^{\perp} = 3$, $B_{d_{\mathcal{D}}^{\perp}} = 8$ for 2nd-order Boolean masking. For the sake of brevity, we denote the IPM2, IPM3, IPM14, IPM23 as the IPM with $L_2 = 2, 3, 14, 23$ respectively, and BM1 (resp., BM2) represents 1st-order (resp., 2nd-order) Boolean masking.

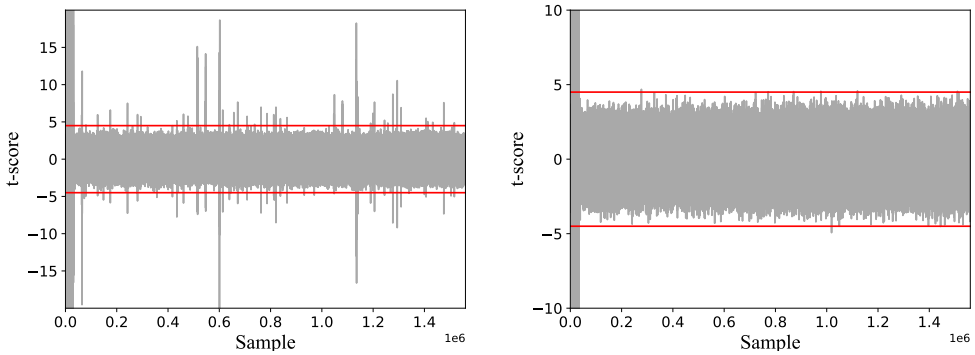
Essentially, security proofs [BFG15, BFG+17, WMCS20] and information-theoretic analysis in [WSY+16, CGC+20] are valid only for the assumed leakage model (usually idealized), but might not be true for the leakage behavior of real devices. In addition, they mostly focus on encoding functions, neglecting the whole encryption process. Therefore, we begin the evaluations with leakage assessment on IPM instances (including BM1).

### 4.2.1 Leakage Assessment

Since the EM measurements cover the first 2.5 AES rounds (elaborated in Section 4.1) in total, this assessment analyzes the leakage behavior of encoding function as well as private computations in the masked domain. Figure 3 depicts the Welch's (two-tailed) t-test results for BM1 and IPM23 (other instances are in Appendix E for brevity). Note that the sampling rate for all instances is set as 156 MHz to ensure that the first 2.5 AES rounds are covered. Actually, this follows the similar acquisition setting for TVLA as in [BFG+17], e.g., 125 MHz in the latter (500,000 samples within 4 ms).

From Figure 3 and Figure 11 in Appendix E, we can observe a great difference between BM and other IPM instances at the same 1st-order word-level security. The implementation protected by BM leaks significantly (many t-test scores exceed the threshold $\pm 4.5$) on quite a lot of time samples. On the contrary, the implementations protected by IPM show much less evidence of leakage than BM, and can be deemed not to leak for this number of measurements. Such difference is consistent with the t-test results (with activated TRNG) in [BFG+17]. Although such t-test in this experiment is too coarse that cannot distinguish the actual security level among those masked implementations, the assessment results demonstrate that the more complex algebraic structure of the encoding function brings a promising alternative to BM since IPM allows reducing both the number of leaking samples and the informativeness of the masked implementations.

[5] In fact, code-based masking for this type when $n = k + m$ and $k = 1$ is equivalent to DSM [PGS+17, CG18]via the equivalence of linear codes, which is more general than IPM.




(a) 1st-order Boolean masking (b) IPM with $L_2 = 23$

**Figure 3:** TVLA (t-test) results for BM1 (left) and IPM23 (right) with TRNG activated and sampling rate at 156 MHz. The red lines mark the $\pm 4.5$ threshold.

### 4.2.2 Attack-based Evaluations

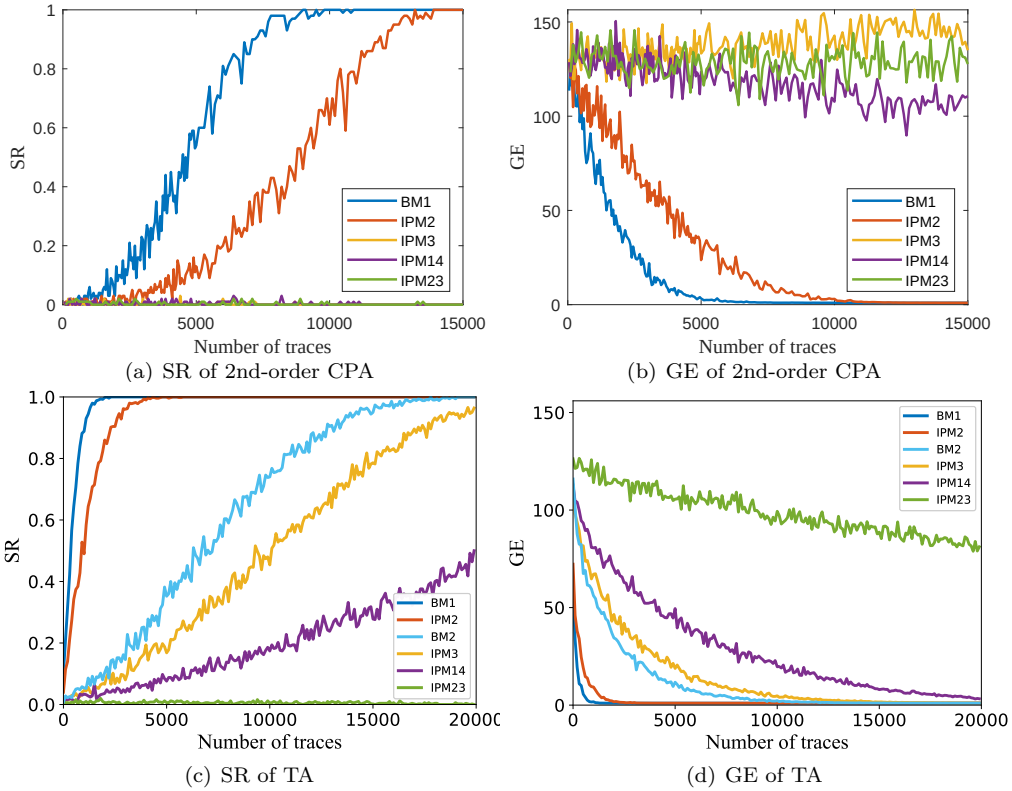
The basic Welch's t-test [6] is able to detect the presence of leakage qualitatively, but it cannot provide more quantitative evaluations. As clarified in Section 4.2.1, the leakage assessment here is unable to prove the "security order amplification" of IPM, let alone indicate the concrete security level of IPM instances. Facing this situation, we utilize two kinds of side-channel attacks [7], namely CPA and TA, representing both non-profiled and profiled types to evaluate the concrete resilience of IPM instances against side-channel attacks in the real world. Firstly, we target the output of L Gadget which possesses the same security property of encoding function. The acquisition settings are clarified in Section 4.1, and specifically in this evaluation, the collected EM measurements encompass $637,500$ samples with a sampling rate of 1.25GHz. The SR and GE results for 2nd-order CPA and TA are illustrated in Figure 4 below. Note that for 2nd-order CPA, we exclude BM2 instance (since it cannot succeed in theory).

From Figure 4(a) and 4(b), we can see that 2nd-order CPA can attack BM1 successfully with less than $10,000$ traces and IPM2 requires more to compromise. However, 2nd-order CPA fails to break up other IPM instances, which are perfectly compatible with theoretical predictions by [CGC+20]. In other words, since the dual distance $d_{\mathcal{D}}^{\perp}$ of BM1 and IPM2 is equal to 2, BM1 and IPM2 can only resist 1st-order CPA but cannot resist 2nd-order CPA (with a squared difference combination). At the opposition, other IPM instances whose $d_{\mathcal{D}}^{\perp} \geq 3$ are able to resist 2nd-order CPA as indicated by [CGC+20] and verified in practice in this work.

Moreover, we shall claim that according to the above-mentioned coding-theoretic parameters (the dual distance $d_{\mathcal{D}}^{\perp}$ and the kissing number $B_{d_{\mathcal{D}}^{\perp}}$), the side-channel resistance of the above IPM (and BM) instances increases in the order of $L_2 = 1, 2, 3, 14, 23$ and BM2 is between IPM2 and IPM3. From Figure 4(c) and Figure 4(d), it is demonstrated that the security levels of selected instances are also consistent with the predictions, which verifies the coding-theoretic leakage quantification for IPM proposed in [CGC+20]. Note that IPM with $L_2 = 23$ is one of the optimal encoders (amongst the best in terms of maximum dual distance and the smallest kissing number) for 2-share IPM [CGC+20] under linear leakage models, and it indeed turns out to provide the best side-channel resistance among the investigated experimental groups. More importantly, it is indicated that even a 2-share IPM (IPM3, IPM14 and IPM23) can better withstand template attacks than

[6] It can be extended to check for univariate higher-order leakage and multivariate analysis if with sufficient computational power.

[7] This is also a supplement and the actual difference between the work of [BFG+17] and ours' regarding the practical evaluation of IPM.



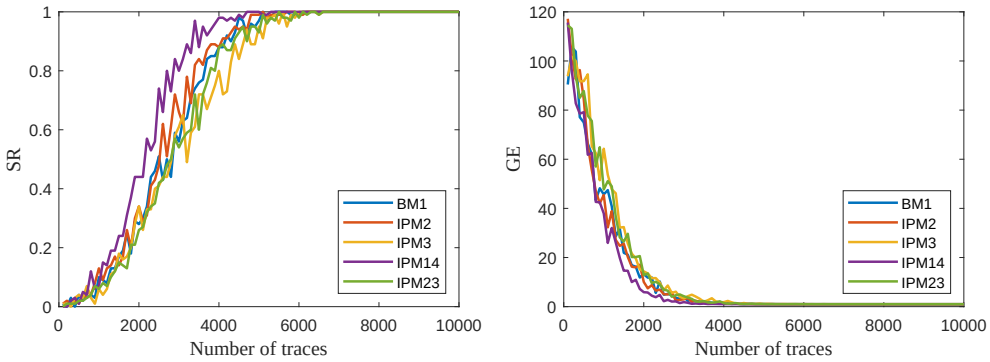

(a) SR of 2nd-order CPA (b) GE of 2nd-order CPA

(c) SR of TA (d) GE of TA

**Figure 4:** SR and GE results for both 2nd-order CPA and TA on non-redundant instances.

the 3-share BM, which is strong support for the "security order amplification" of IPM (also code-based masking of the non-redundant type). Combined with the theoretical predictions, our experimental results complement the practical side-channel analysis in order for a sound evaluation of concrete security for IPM. This should be of special interest for cryptographic designers since they can tackle the specific security level of IPM by two coding-theoretic parameters as in [CGC+20].

As discussed above, there exists a possible security bottleneck in the current computational framework, which locates at the matrix $\mathbf{T}$ of L Gadget. When it comes to $n = k+m$ and $k = 1$ for code-based masking, the matrix $\mathbf{T}$ is virtually the additive sharing of the input sensitive variable, degrading high-algebraic masking schemes to Boolean masking, and resulting in practical security loss. The reason is that BM is practically more prone to attacks than other IPM instances as demonstrated in Figure 4. Hence we launch 2nd-order CPA on IPM instances (including BM1) again, targeting the theoretical weakest part (the matrix $\mathbf{T}$ of L Gadget) during gadgets computations. The results are illustrated in Figure 5. It is clear that all IPM instances can be easily attacked by 2nd-order CPA, substantially different from Figure 4. In addition, all IPM instances have a similar security level to BM1, losing the feature of "security order amplification". This is not surprising since the gadgets are devised originally to keep the consistent word-level security order (instead of bit-level security order). Concerning the distinct security levels between encoding and computations (Figure 4 and 5, resp.) under the same word-level security order, one may query the practical relevance of the theoretical (word-level) probing model. However, the experimental results demonstrate that the switch between code-based masking and additive sharing in the gadgets computations indeed limits the security level of whole protected implementations for the non-redundant type.




(a) SR of 2nd-order CPA (b) GE of 2nd-order CPA

**Figure 5:** SR and GE results for IPM instances.

In summary, we leverage both leakage detection and side-channel attacks to evaluate the practical security of code-based masking for the non-redundant type (when $n = k + m$ and $k = 1$). Concerning encoding functions, we demonstrate that distinct linear codes (with various coding-theoretic properties) for the code-based encoders actually play a dominant role in the concrete security level of the masked implementations (if implemented ideally). We also confirm the "security order amplification" of IPM in practice and verify the applicability of the theoretical approach proposed in [CGC+20]. With respect to private computations, we find and verify the security bottleneck of gadgets computation, which assists in pushing forward the design and improvement of the computational framework for code-based masking (which we leave as our future work).

## 4.3 Security Evaluation for Packed Type

Now we consider the case when $n = k + m$ and $k > 1$ which involves the application of the "cost amortization" technique for code-based masking. As discussed in Section 3.4, the packed type encodes multiple (for $k > 1$) sensitive variables into one codeword and thus the gadgets computations on this codeword allow for manipulating those $k$ sensitive variables in parallel. This type of code-based masking (containing both encoding and gadgets computations) is proved to be $m$th-order secure in the word-level probing model whatever how large $k$ is [WMCS20]. However, intuitively such packed operations inclined to leak more sensitive information. For example, considering BM with $k = 2$, $m = 1$, $n = 3$ (the corresponding encoder is depicted in Table 5), the two (for $k = 2$) sensitive variables have to use one common mask, then such mask potentially involves more operations during computations, increasing the opportunity of exposure. Inspired by such an idea, we investigate the practical security of code-based masking of this type.

We target BM instances since our intention is to study the practical security level of packed implementations with increasing $k$. We consider three 1st-order word-level secure BM instances: $\text{BM}^{k=1}$, $\text{BM}^{k=2}$ and $\text{BM}^{k=4}$ correspond to $k \in \{1, 2, 4\}$ [8], respectively. The corresponding encoders $\mathbf{A}$ over $\mathbb{F}_{2^8}^{(k+m)\times n}$ are shown in Table 5, where all $k$ sensitive variables reuse one common mask.

### 4.3.1 Leakage Assessment

As discussed above, exploiting the shared mask stands a great chance to produce more information leakage during computations. For this reason, we firstly conduct leakage detection using TVLA, with the acquisition settings clarified in Section 4.1. The sampling

[8] To take full advantage of implementation platform (actually by the width of 32-bit registers in ARM Cortex-M4), the value of $k$ should be a multiple of 2.

**Table 5:** Various choices of generator matrices $\mathbf{A}$ over $\mathbb{F}_{2^8}^{(k+m)\times n}$ for BM instances in packed code-based masking with $n = k + m$ and $m = 1$.

| | $\text{BM}^{k=1}$: $k = 1,\ m = 1$ | $\text{BM}^{k=2}$: $k = 2,\ m = 1$ | $\text{BM}^{k=4}$: $k = 4,\ m = 1$ |
|---|---|---|---|
| $\mathbf{A}$ | $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$ |

rate is set to 100 MHz to cover the first 2.5 AES rounds of BM instances particularly for $\text{BM}^{k=4}$ and it largely follows the acquisition setting of [BFG+17]. Figure 6 illustrates the Welch's t-test results for the packed BM implementations. By comparing the t-scores, all three BM instances leak significantly. More interestingly, with increasing $k$, more time samples are inclined to leak. Roughly, the amount of leaking samples of $\text{BM}^{k=2}$ is as twice that of $\text{BM}^{k=1}$, while the number of leaking samples $\text{BM}^{k=4}$ exceeding the thresholds is far more than the former two [9]. Summing up, the shared mask shall as expected participate in more computations, resulting in more information leakage.

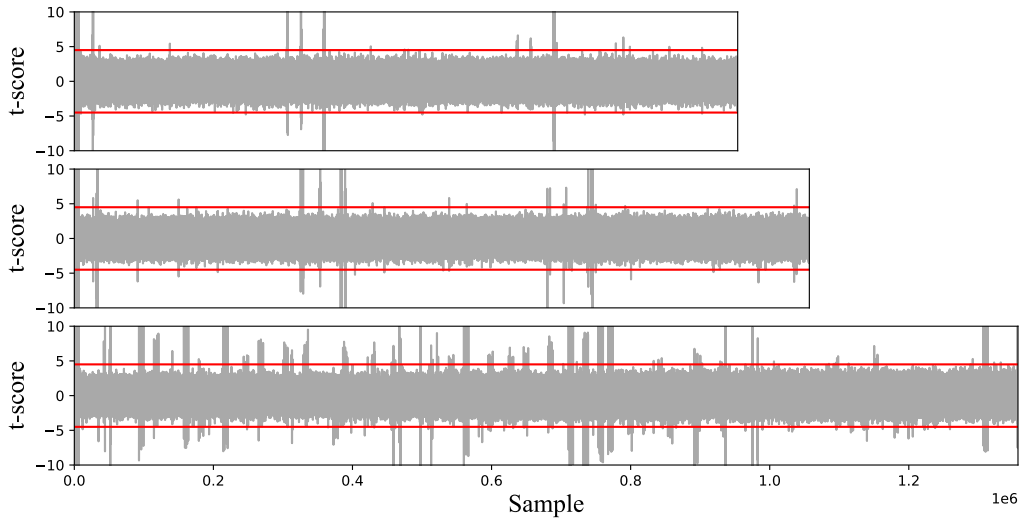



**Figure 6:** TVLA (t-test) results for $\text{BM}^{k=1}$ (top), $\text{BM}^{k=2}$ (middle) and $\text{BM}^{k=4}$ (bottom) with a sampling rate of 100 MHz. The red lines mark the ±4.5 threshold.

### 4.3.2 Attack-based Evaluation

The above leakage assessment demonstrates that packed masking schemes shall produce more information leakage, and we further conduct 2nd-order CPA to investigate the concrete security level of the packed BM instances. To begin with, we highlight that the side-channel security orders (both in word- and bit-probing models) of encoding and the gadgets computations for the selected BM instances are actually consistent [10]. The matrix $\mathbf{T}$ (over $\mathbb{F}_{2^8}^{n\times k}$) of L Gadget (the security bottleneck shown in Section 4.2.2) for

[9] Notably, the associated executing operations of samples are not exactly the same among the three instances due to their different time cost.

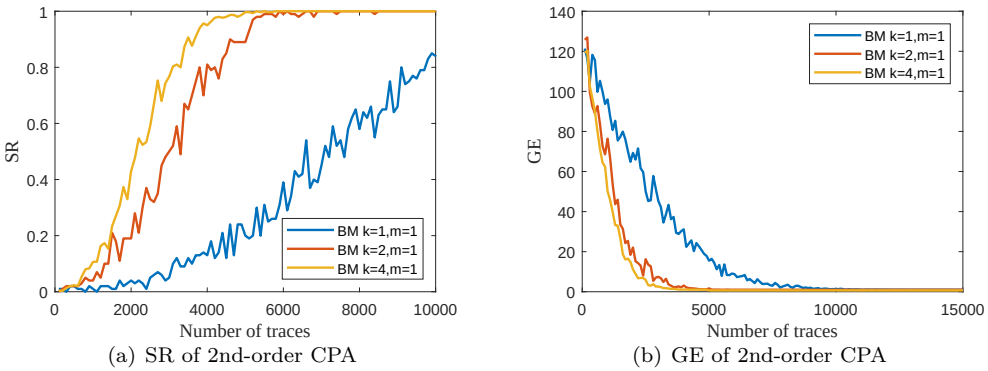
[10] If the packed type is instantiated into IPM instance, gadgets computations shall possibly cause bit-probing security degradation due to the security bottleneck.

the three BM instances are shown in Table 6, where $m_i$ for $1 \leq i \leq 4$ and $r$ are random variables over $\mathbb{F}_{2^8}$. Recall that each column of the matrix $\mathbf{T}$ is an additive sharing for the corresponding unmasked sensitive variable. Hence for each sensitive variable, only $m+1$ shares are effective in the corresponding additive sharing, although there are $n$ shares in total. Furthermore, similar to the encoding, one mask is shared for protecting all $k$ sensitive variables. Therefore, we only target the encoding in this type.

**Table 6:** Illustration of matrix $\mathbf{T}$ over $\mathbb{F}_{2^8}^{n\times k}$ for the packed BM instances.

| | $\mathrm{BM}^{k=1}$: $k=1,\ n=2$ | $\mathrm{BM}^{k=2}$: $k=2,\ n=3$ | $\mathrm{BM}^{k=4}$: $k=4,\ n=5$ |
|---|---|---|---|
| $\mathbf{T}$ | $\begin{pmatrix} m_1 \\ r \end{pmatrix}$ | $\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \\ r & r \end{pmatrix}$ | $\begin{pmatrix} m_1 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 \\ 0 & 0 & m_3 & 0 \\ 0 & 0 & 0 & m_4 \\ r & r & r & r \end{pmatrix}$ |

Next, we conduct 2nd-order CPA on the output of L Gadget (specifically we target the output of the first S-box in the first AES round), concentrating on how the practical side-channel resistance varies with $k$. The output of the S-box is actually a vector over $\mathbb{F}_{2^8}^n$ (containing $k$ variables and $m=1$ mask), then $k$ (out of 16) bytes of the keys can be rebuilt from the same random mask and the other corresponding shares. Therefore, for $\mathrm{BM}^{k=2}$ and $\mathrm{BM}^{k=4}$ we have the ability to recover 2 and 4 subkey bytes, respectively. The target EM measurement sets for each instance have a size of $10,000$ and each measurement contains $781,250$ samples with a sampling rate at 313 MHz [11]. The SR and GE results of 2nd-order CPA on the three instances are illustrated in Figure 7. Notably, since $k$ subkey bytes can be restored for packed instances, the SR and GE are obtained by averaging.




(a) SR of 2nd-order CPA (b) GE of 2nd-order CPA

**Figure 7:** SR and GE results of 2nd-order CPA for BM instances on packed type.

From Figure 7, we can observe that the side-channel resistance of packed instances decreases with increasing $k$, which accords with the leakage behavior as shown in Section 4.3.1. Essentially, the value of $k$ also implies the number of subkeys adversaries can rebuild by knowing only one mask, which could lead to more efficient attacks. However, the attack results deviate from the theoretical analysis in [WMCS20], which shows those three packed instances have the same security orders under the word-level probing model. Hence it again poses a challenge to the practical relevance of the word-level probing model. Interestingly, one possible explanation for this concrete security decrease is from a coding-theoretic perspective [CGC+20] that, although the dual distances (over both $\mathbb{F}_2$

[11] Hence the result for $\mathrm{BM}^{k=1}$ is distinct from the one with the same setting in Section 4.2.2 which samples at 1.25GHz.

and $\mathbb{F}_{2^8}$) of the three packed BM instances are the same: $d_{\mathcal{D}}^{\perp} = 2$; the kissing numbers are largely different: $B_{d_{\mathcal{D}}^{\perp}} \in \{8, 24, 80\}$ for $k \in \{1, 2, 4\}$, respectively. Note that a larger kissing number indicates more possibilities to reconstruct the sensitive variable by utilizing the encodings.

Summing up, by means of both leakage detection and practical attacks, we show that the packed operation leads to more information leakage in practice and accordingly more prone to side-channel attacks. In addition, the theoretical weakest part of gadgets computations shall neither decrease the security level of encoding from a packed view, nor be capable of lifting the resistance by increasing shares. Therefore, with the discussion in Section 3.4 regarding the efficiency of packed operation, the advantage of the cost amortization technique applied in code-based masking [12] promisingly lies in reducing the randomness cost. It utilizes less randomness given sufficient large $k$ and $m$, so that it can be exploited as an alternative when applying to some platforms with constrained randomness resources. However, more attention must be paid when applying the amortization technique because of the concrete security loss in practice originated from the reuse of random numbers.

## 4.4 Security Evaluation for Redundant Type

One compelling merit of code-based masking is its potential against fault injection analysis when equipped with redundancy in encoding, which corresponds to the case when $n > k+m$. Similarly, such redundant type (including both encoding and masked gadgets) is proved to be $m$th-order secure in the word-level probing model [WMCS20] as well. However, recent research shows that more redundancy in code-based masking leads to more leakage in both encoding and operations. In particular, given a fixed $m$, increasing $n$ can only incur more leakage from an information-theoretic perspective [CGC+21], which is also validated by (simulated) attack-based evaluation [CS21]. In this section, we concentrate on the evaluation of the practical security for redundant cases, paving the way for future research with regard to fault injection analysis of code-based masking.

For this purpose, we set three experimental groups with the same $k = 1$ and $m = 1$ but increasing $n$: RE1 with $n = 2$ (without redundancy as the baseline), RE2 with $n = 3$ and RE3 with $n = 4$. Note that they all have 1st-order security under the word-level probing model. Their corresponding encoders $\mathbf{A}$ over $\mathbb{F}_{2^8}^{(k+m)\times n}$ are depicted in Table 7. In fact, we initialize RE1 by IPM23 as already studied in Section 4.2. It is worth mentioning that, firstly, taking $L_2 \in \{23, 29, 51\}$ leads to 2-share IPM instances with the maximized dual distance $d_{\mathcal{D}}^{\perp} = 4$. Secondly, however, the dual distances of the corresponding codes in RE1, RE2 and RE3 are decreasing that $d_{\mathcal{D}}^{\perp} \in \{4, 3, 2\}$. Therefore, we shall verify the impact of more shares on side-channel resistance. Note that the adjusted kissing numbers for three instances are $B'_{d_{\mathcal{D}}^{\perp}} \in \{4, 10, 1\}$, respectively.

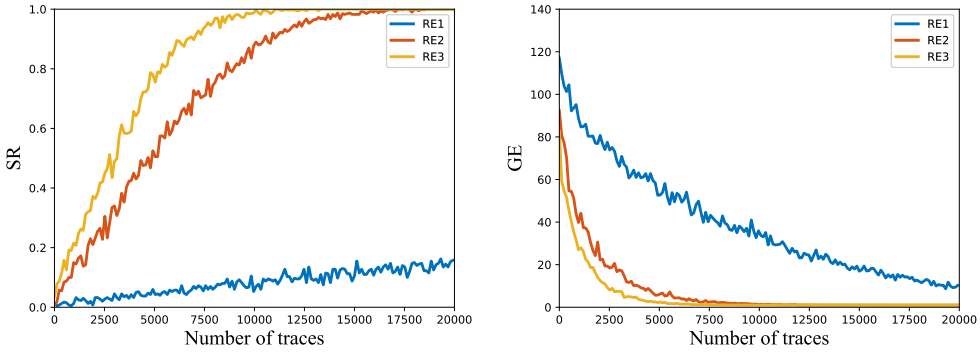
**Table 7:** Various choices of $\mathbf{A}$ over $\mathbb{F}_{2^8}^{(k+m)\times n}$ in redundant cases with $k = 1$ and $m = 1$.

| | RE1: $n = 2$ | RE2: $n = 3$ | RE3: $n = 4$ |
|---|---|---|---|
| | Non-redundant | Redundant | Redundant |
| $\mathbf{A}$ | $\begin{pmatrix} 1 & 0 \\ 23 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 \\ 23 & 29 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 23 & 29 & 51 & 1 \end{pmatrix}$ |

We utilize template attacks on both encoding and the theoretical weakest part in the gadgets computations. Considering matrix $\mathbf{T}$ (over $\mathbb{F}_{2^8}^{n\times k}$) in L Gadget, although it turns into additive sharing with $n$ shares, the $n-1$ shares are all related to only one random

[12] If $m$ is sufficiently large, the cost amortization technique may mitigate the cost by taking the scheme proposed in [WMCS20].

mask. Hence only $m+1$ shares are sufficient to recover the subkey, which implies that the redundant cases degrade from $m$th-order word-level secure code-based masking to $m$th-order word-level secure BM as well in matrix $\mathbf{T}$ of L Gadget, resulting in loss of practical security. We collect three EM measurements sets for those three instances by setting sampling rates to 625 MHz. By targeting the output of L Gadget, the results on all the selected instances are shown in Figure 8 to illustrate how redundancy affects the practical security of code-based masking. It is indicated from Figure 8 that the redundancy indeed leads to a practical security decrease. In particular, the security level of RE2 is significantly lower than RE1, which is compatible with the coding-theoretic properties [CGC+21]: $d_{\mathcal{D}}^{\perp}=3$, $B'_{d_{\mathcal{D}}^{\perp}}=10$ for the former and $d_{\mathcal{D}}^{\perp}=4$, $B'_{d_{\mathcal{D}}^{\perp}}=4$ for the latter, respectively. Moreover, adding one more share of redundancy further reduces the dual distance to $d_{\mathcal{D}}^{\perp}=2$, and again leads to a security decrease in the sense of attacks.

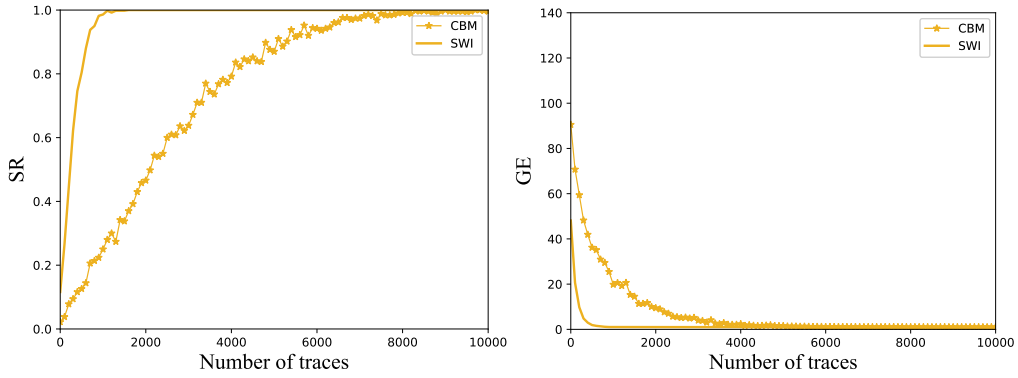



(a) SR of TA (b) GE of TA

**Figure 8:** SR and GE results targeting the output of L Gadget for redundant type.

Regarding the bottleneck part in switching from code-based masking to additive sharing, namely the matrix $\mathbf{T}$ of L Gadget, here we focus on RE3 for the sake of brevity, since our intention is to present the practical security loss for each instance (see Section 4.2 for security loss of RE1 and Appendix F for RE2). The comparison results between the encoding function and the bottleneck part are depicted in Figure 9, where "CBM" (resp., " SWI") indicates the target is the output (resp., the matrix $\mathbf{T}$) of L Gadget. From Figure 9 and recall that it is $d_{\mathcal{D}}^{\perp}=2$, $B_{d_{\mathcal{D}}^{\perp}}=8$ for 1st-order BM, we can observe that the switching part again reduces the practical security level of encoding similar to the results in Section 4.2. In particular, the minimum number of measurements achieving SR$\geq 95\%$ is reduced from about $8,000$ for CBM case to 820 for SWI case (about ten times of reduction). Therefore, the security order amplification brought by the high-algebraic structure of encoding vanishes, because of the additive sharing in matrix $\mathbf{T}$. Recall that the encodings and computations possess the same probing security order, the attack results from Figure 8 and 9 again indicate that the word-level probing model is not sufficient to depict the practical side-channel resistance of masked implementations.

To conclude, we target the redundant type of code-based masking and show that redundancy usually brings in a decline in the practical security level. Our empirical results are consistent with both simulated evaluation in [CS21] and information-theoretic evaluation carried out in [CGC+21] for redundant code-based masking. In addition, we again confirm that the internal switch to additive sharing during gadgets computations reduces the concrete security of code-based encoders with a higher-algebraic structure for this redundant type.




(a) SR of TA (b) GE of TA

**Figure 9:** SR and GE results targeting L Gadget for redundant type.

## 4.5 Further Discussions

In the following, we further discuss the security order amplification in code-based masking and the practical relevance of the probing model.

**Security Order Amplification.** Security order amplification [WSY+16, PGS+17] is commonly known as a positive feature of IPM, and it is demonstrated to be an intrinsic feature for the encoding [BFG+17] but not for the masked operations (e.g., secure multiplications). However, the evaluation results in Sections 4.2 and 4.4 demonstrate that security order amplification emerges in code-based masking as well and can further enhance masked computations (recall that our attack target is the output of L gadget). In fact, security order amplification happens if the numerical degree [CGC+21] of the leakage function is smaller than the bit-probing security order. It is not only a special feature of encoding, but also exists in computations: linear operations are trivial by using encoding; nonlinear functions like S-box are usually done by fully masked additions and multiplications with proper refreshments. Therefore, if the basic gadgets are well-encoded (e.g., under strong non-interference construction), then nonlinear functions should also keep security order amplification. This can also help reason about the practical security loss introduced by the "weakest part" of gadgets computations. Due to the internal switch to additive sharing, the original code-based encoding degrades to Boolean encoding (thus not fully well-encoded), therefore losing the feature of security order amplification and causing the security decrease in practice.

As already pointed out in [BFG+17], security order amplification is not unconditional. That is, security order amplification will vanish if the leakage function is non-linear (precisely, if the numerical degree of the leakage function exceeds the bit-probing security order). Indeed, the leakage function of real devices will not be exactly linear in most practical scenarios. However, the linear parts are usually more dominant than the non-linear parts in the observable leakage [PGS+17]. Therefore if the adversary can only capture and exploit the linear leakage part, security order amplification still plays its advantage. For example, in the evaluations for the non-redundant type of code-based masking (see Figure 4), security order amplification emerges against 2nd-order CPA, which mostly leverages the linear leakage for attack. Therefore, it is still promising to devise fully well-encoded gadgets for masking schemes, preventing the masked implementations from side-channel threats to some extent. At last, our findings and verifications on the security bottleneck part of current gadgets computations could assist in developing fully encoded secure computations for code-based masking, which is still an open problem and we leave it as our future research.

**Two Probing Models.** The three masked gadgets proposed in [WMCS20] and improved in Section 3 all keep the consistent word-level probing security orders with the corresponding code-based encoders. However, it is indicated in our evaluations that the internal switch to additive sharing during gadgets computations causes a security loss (even degradation) in practice. In other words, keeping the same word-level probing security order is unable to guarantee that the masked gadgets are equally secure as the corresponding encoders. Hence a more practice-relevant model is required for gadgets design. From our evaluation results, the security level characterized by the coding-theoretic properties, namely dual distance and (adjust) kissing number defined in the bit-probing model, is more in line with the concrete side-channel resistance in practice. Hence it is recommended to look for masked gadgets which maintain the consistent bit-probing security order with encoding.

# 5 Conclusions and Future Work

In this paper, we target code-based masking and investigate its efficient implementations as well as the practical security against real-world attacks. On the one hand, We propose an improved scheme based on the computational framework of [WMCS20], enabling a far more efficient scheme compared to the original one from a computational complexity perspective. We further apply our improved scheme to several efficient implementations of AES-128. Both theoretical analyses and performance evaluations (by clock cycles) on the practical implementations show that our improvements are significant. On the other hand, we provide an extensive evaluation of the practical security of code-based masking by taking three representative types. For each type, we target both encoding function and private computations. In the sense of encoding, we provide strong evidence for "security order amplification" of code-based masking. In addition, we discover that the "cost amortization" technique shall incur a decline in concrete security level, which deviates from the conclusion drawn in [WMCS20]. We further verify that the redundant code-based encoders indeed bring in security loss against side-channel attacks. Regarding the gadgets computations, we identify a security bottleneck existing in the internal additive sharing during gadgets computations, which usually reduces the practical security level of code-based masking.

Because of the security bottleneck we have identified, the current computational framework for code-based masking usually fails to give full play of the merits featured by code-based encoders. Still, this framework is the only solution for code-based masking with generic encoders. We highlight that our improvements are mainly focused on practical encoder $\mathbf{A}$ which is irrelevant to computations that shall be applicable for strengthened schemes in the future. As such, our future research will concentrate on the construction and verification of a fully encoded computational framework for code-based masking by addressing the back-and-forth switch inside masked gadgets.

## Acknowledgments

This work was also supported in part by National Key R&D Program of China (2020AA A0107700), by National Natural Science Foundation of China (62072398), by Open Fund of State Key Laboratory of Cryptology (MMKFKT202013), by Alibaba-Zhejiang University Joint Institute of Frontier Technologies, by Major Scientific Research Project of Zhejiang Lab (2018FD0ZX01), by Zhejiang Key R&D Plan (2021C01116), by Leading Innovative and Entrepreneur Team Introduction Program of Zhejiang (2018R01005), by Research Institute of Cyberspace Governance in Zhejiang University, by National Key Laboratory of Science and Technology on Information System Security, by State Key Laboratory of Mathematical Engineering and Advanced Computing, and by Key Laboratory of Cyberspace Situation Awareness of Henan Province.

The authors sincerely thank the anonymous reviewers for their valuable comments, which significantly improved the quality of the paper. Besides, the authors acknowledge financial support of the French national bank (BPI) under SECURYZR-V grant (Contract n° DOS0144216/00), a RISC-V centric platform integrating security co-processors.

## References

[BCC+14] Julien Bringer, Claude Carlet, Hervé Chabanne, Sylvain Guilley, and Houssem Maghrebi. Orthogonal direct sum masking. In *IFIP International Workshop on Information Security Theory and Practice*, pages 40–56. Springer, 2014.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer, 2004.

[BDF+17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 535–566. Springer, 2017.

[BFG15] Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. Inner product masking revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–510. Springer, 2015.

[BFG+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 724–754. Springer, 2017.

[CDD+15] Claude Carlet, Abderrahman Daif, Jean-Luc Danger, Sylvain Guilley, Zakaria Najm, Xuan Thuy Ngo, Thibault Porteboeuf, and Cédric Tavernier. Optimized linear complementary codes implementation for hardware Trojan prevention. In *European Conference on Circuit Theory and Design, ECCTD 2015, Trondheim, Norway, August 24-26, 2015*, pages 1–4. IEEE, 2015.

[CG18] Claude Carlet and Sylvain Guilley. Statistical properties of side-channel and fault injection attacks using coding theory. *Cryptography and Communications*, 10(5):909–933, 2018.

[CGC+20] Wei Cheng, Sylvain Guilley, Claude Carlet, Sihem Mesnager, and Jean-Luc Danger. Optimizing inner product masking scheme by a coding theory approach. *IEEE Transactions on Information Forensics and Security*, 16:220–235, 2020.

[CGC+21] Wei Cheng, Sylvain Guilley, Claude Carlet, Jean-Luc Danger, and Sihem Mesnager. Information leakages in code-based masking: A unified quantification approach. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):465–495, 2021.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CMP18] Hervé Chabanne, Houssem Maghrebi, and Emmanuel Prouff. Linear repairing codes and side-channel attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):118–141, 2018.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[CS21] Nicolas Costes and Martijn Stam. Redundant code-based masking revisited. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):426–450, 2021.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.

[DWBV+96] Erik De Win, Antoon Bosselaers, Servaas Vandenberghe, Peter De Gersem, and Joos Vandewalle. A fast software implementation for arithmetic operations in GF($2^n$). In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 65–76. Springer, 1996.

[FMPR10] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In *International Workshop on Selected Areas in Cryptography*, pages 262–280. Springer, 2010.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[GM11] Louis Goubin and Ange Martinelli. Protecting AES with Shamir's secret sharing scheme. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 79–94. Springer, 2011.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 567–597. Springer, 2017.

[GT02] Jovan D Golić and Christophe Tymen. Multiplicative masking and power analysis of aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 198–212. Springer, 2002.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 55–74. Springer, 2014.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Annual International Cryptology Conference*, pages 463–481. Springer, 2003.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[MS77] F. Jessie MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. Elsevier, Amsterdam, North Holland, 1977. ISBN: 978-0-444-85193-2.

[PGS+17] Romain Poussier, Qian Guo, François-Xavier Standaert, Claude Carlet, and Sylvain Guilley. Connecting and improving direct sum masking and inner product masking. In *International Conference on Smart Card Research and Advanced Applications*, pages 123–141. Springer, 2017.

[PR11] Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Transactions on computers*, 58(6):799–811, 2009.

[RO04] Christian Rechberger and Elisabeth Oswald. Practical template attacks. In *International Workshop on Information Security Applications*, pages 440–456. Springer, 2004.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[SLC+21] Patrick Solé, Yi Liu, Wei Cheng, Sylvain Guilley, and Olivier Rioul. Linear Programming Bounds on the Kissing Number of q-ary Codes. In *2021 IEEE Information Theory Workshop (ITW2021)*, Kanazawa, Japan, October 2021.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 495–513. Springer, 2015.

[SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer, 2009.

[WGS+20] Weijia Wang, Chun Guo, François-Xavier Standaert, Yu Yu, and Gaëtan Cassiers. Packed multiplication: How to amortize the cost of side-channel masking? In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 851–880. Springer, 2020.

[WMCS20] Weijia Wang, Pierrick Méaux, Gaëtan Cassiers, and François-Xavier Standaert. Efficient and private computations with code-based masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 128–171, 2020.

[WSY+16] Weijia Wang, François-Xavier Standaert, Yu Yu, Sihang Pu, Junrong Liu, Zheng Guo, and Dawu Gu. Inner product masking for bitslice ciphers and security order amplification for linear leakages. In *International Conference on Smart Card Research and Advanced Applications*, pages 174–191. Springer, 2016.

# A Detailed Algorithms for Improved Gadgets

Here we shall present the detailed algorithms of our improved L Gadget and Ls Gadget which basically follow the adjusted procedure (or saying multiplication Gadget) elaborated in Section 3.2. The algorithms are illustrated in Algorithm 2 and Algorithm 3.

**Algorithm 2:** Improved L Gadget

**Input:** Codewords $\hat{\mathbf{x}} = \mathbf{x}\mathbf{A}$ of $\hat{\mathbf{x}} \in \mathbb{F}_q^n$ and $\mathbf{x} \in \mathbb{F}_q^k$

**Output:** $\hat{\mathbf{z}} \in \mathbb{F}_q^n$ such that $\mathbf{z} = f(\mathbf{x})$ and $f$ is a linear function for $\mathbb{F}_q^k \mapsto \mathbb{F}_q^k$

Initialize $\mathbf{R}_2$ uniformly over $\mathbb{F}_q^{n\times m}$
$\hat{\mathbf{R}}_2 = \mathbf{R}_2\mathbf{H}$
**for** $i = 1$ **to** $n$ **do**
  **for** $j = 1$ **to** $k$ **do**
    $\mathbf{S}[i, j] = \hat{\mathbf{x}}[i]$
  **end**
  **for** $j = k + 1$ **to** $n$ **do**
    $\mathbf{S}[i, j] = 0$
  **end**
**end**
$\mathbf{M}_i = \mathbf{M}_i^*[*, 1 : k]$, where $\mathbf{M}_i^* = (\tilde{\mathbf{A}}[i, *] \otimes \tilde{\mathbf{A}})\mathbf{E}$ and $\tilde{\mathbf{A}} \stackrel{def}{=} [\mathbf{A}^{-1}[*, 1 : k], \mathbf{O}^{n\times m}]$
**for** $i = 1$ **to** $n$ **do**
  $\mathbf{T}[i, *] = \mathbf{S}[i, *] \times \mathbf{M}_i$
**end**
**for** $i = 2$ **to** $n$ **do**
  $\mathbf{V}[i, *] = f(\mathbf{T}[i, *])$
**end**
$\mathbf{V}[1, *] = f(\mathbf{T}[1, *]) \oplus (n - 1)\mathbf{c}$, while $\mathbf{c}$ is a constant in $\mathbb{F}_q^k$ associated to $f$, and $(n - 1)\mathbf{c} = \sum_{i=1}^{n-1} \mathbf{c}$
$\mathbf{K} = [\mathbf{V}, \mathbf{O}^{n\times(n-k)}] \oplus \hat{\mathbf{R}}_2$
**for** $j = 1$ **to** $n$ **do**
  $\hat{\mathbf{z}}[j] = \sum_{i=1}^{n} \mathbf{K}[i, j]$
**end**
**return** $\hat{\mathbf{z}}$

We should claim that the improved gadgets possess the same security property as the original ones proposed in [WMCS20], since the corresponding improvements (detailed in Section 3.2) shall not introduce security loss. That is, if the practical encoder $\mathbf{A}$ is $d$-privacy secure over the word-level probing model, the improved L Gadget (constructed in

**Algorithm 3:** Improved Ls Gadget

**Input:** Codewords $\hat{\mathbf{x}}_1 = \mathbf{x}_1\mathbf{A}, \cdots, \hat{\mathbf{x}}_l = \mathbf{x}_l\mathbf{A}$ for $\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_l \in \mathbb{F}_q^n$ and $\mathbf{x}_1, \cdots, \mathbf{x}_l \in \mathbb{F}_q^k$

**Output:** $\hat{\mathbf{z}}_1, \cdots, \hat{\mathbf{z}}_{l'} \in \mathbb{F}_q^n$ such that $\mathbf{z}_1, \cdots, \mathbf{z}_{l'} = f(\mathbf{x}_1, \cdots, \mathbf{x}_l)$ and $f$ is a linear function for $\mathbb{F}_q^{kl} \mapsto \mathbb{F}_q^{kl'}$

1 Call Algorithm 2 part from line 3 to line 14 $l$ times for each inputs $\mathbf{x}_1, \cdots, \mathbf{x}_l \in \mathbb{F}^k$, resulting $\mathbf{T}_1, \cdots, \mathbf{T}_l$
2 $\mathbf{T} = [\mathbf{T}_1, \cdots, \mathbf{T}_l]$
3 Call Algorithm 2 part from line 15 to line 18 with input $\mathbf{T}$, resulting in $\mathbf{V}$ ($\mathbf{V} = [\mathbf{V}_1, \cdots, \mathbf{V}_{l'}]$)
4 $\mathbf{V}_1, \cdots, \mathbf{V}_{l'} \leftarrow \mathbf{V}$
5 Call Algorithm 2 part from line 19 to line 22 $l'$ times for each inputs $\mathbf{V}_1, \cdots, \mathbf{V}_{l'}$ with newly constructed $\hat{\mathbf{R}}_2$ by Algorithm 2 from line 1 to line 2 for each time, resulting in $\hat{\mathbf{z}}_1, \cdots, \hat{\mathbf{z}}_{l'}$
6 **return** $\hat{\mathbf{z}}_1, \cdots, \hat{\mathbf{z}}_{l'}$

Algorithm 2) and improved Ls Gadget (constructed in Algorithm 3) also feature $d$-privacy security in the word-level probing model. The proof for these two gadgets in [WMCS20] still applies.

# B Encoding and Decoding

Now we will present the related encoding and decoding functions by utilizing the practical encoder $\mathbf{A}$ constructed in Section 3.1. Let $\boldsymbol{x}$ and $\boldsymbol{r}$ be vectors over $\mathbb{F}_q^k$ and $\mathbb{F}_q^m$, respectively, where $\boldsymbol{x}$ represents the sensitive variables that should be protected and $\boldsymbol{r}$ consists of random masks uniformly distributed in $\mathbb{F}_q$. Encoding equals the product $[\boldsymbol{x}, \boldsymbol{r}]\mathbf{A}$, which can be computed by Equation 7. As we can see, the sparsity of $\mathbf{A}$ enables only $a_{ij}$ to participate in the multiplication and hence the computational complexity of multiplication operations for encoding can be reduced from $(k+m) \times n$ (for generic encoder) to $m \times (n-m)$.

$$\begin{aligned}[\boldsymbol{x}, \boldsymbol{r}]\mathbf{A} =& [\boldsymbol{x}[1] + \sum_{j=1}^{m} r[j] \times a_{j1}, \cdots, \boldsymbol{x}[k] + \sum_{j=1}^{m} r[j] \times a_{jk}, \\ & \sum_{j=1}^{m} r[j] \times a_{j(k+1)}, \cdots, \sum_{j=1}^{m} r[j] \times a_{j(n-m)}, \boldsymbol{r}[1], \cdots, \boldsymbol{r}[m]].\end{aligned} \tag{7}$$

We suppose that $\hat{\boldsymbol{x}}$ is the codeword of $\boldsymbol{x}$ (namely $\hat{\boldsymbol{x}} = [\boldsymbol{x}, \boldsymbol{r}]\mathbf{A}$). Thus decoding can be evaluated by a product of $\hat{\boldsymbol{x}}$ and $\mathbf{A}^{-1}[*, 1:k]$, so that we get:

$$\hat{\boldsymbol{x}}\mathbf{A}^{-1}[*, 1:k] = [\hat{\boldsymbol{x}}[1] + \sum_{j=k+1}^{n} \hat{\boldsymbol{x}}[j] \times a'_{j1}, \cdots, \hat{\boldsymbol{x}}[k] + \sum_{j=k+1}^{n} \hat{\boldsymbol{x}}[j] \times a'_{jk}]]. \tag{8}$$

where $a'_{ij}$ denotes the entry in the lower $(n-k) \times k$ sub-matrix of $\mathbf{A}^{-1}[*, 1:k]$. Similarly, the number of multiplication operations for decoding decreases from $n \times k$ (for generic encoder) to $(n-k) \times k$ as the top $k \times k$ sub-matrix of $\mathbf{A}^{-1}[*, 1:k]$ is an identity matrix.

# C Clock Cycles Comparison for Whole implementation

Here we shall provide the performance comparison regarding clock cycle counts between the original scheme [WMCS20] and our improved scheme. Firstly, we apply those two

schemes to the AES-128 implementations. In order to enable a fair comparison, both schemes are implemented on the same platform (introduced in Section 3.5) and follow the same optimization strategy (excluding the essential difference between their masking schemes). The trend of clock cycle counts with increasing $m$ (fixing $k = 1$) for the masking schemes is depicted in Figure 10. It is indicated from Figure 10 that the clock cycle counts of our scheme are much lower than those of [WMCS20] for all the cases and the gap becomes larger with increasing $m$. Actually, the performance trend is resembling that of Figure 1(a), which further provides strong validation that our improvement is significant from a computational perspective.

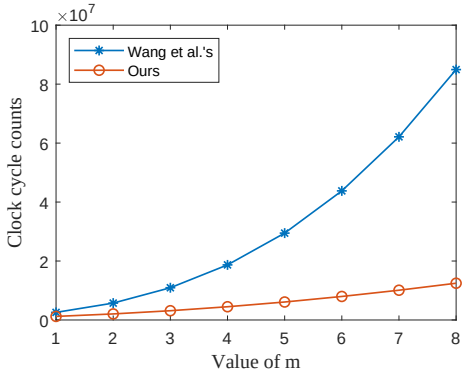



**Figure 10:** Comparison of the number of clock cycles with increasing $m$ and fixed $k = 1$ for the original masked implementation and our improved ones.

# D Clock Cycles for Cost Amortization

Here we present the physical clock cycles counts of both well-constructed instances (introduced in 3.4) in [WMCS20] and our improved scheme to illustrate the practical performance of the "cost amortization" technique. Both schemes are applied to AES-128 implementations and the associated platform is introduced in Section 3.5. To exhibit the efficiency of cost amortization, we choose to fix $m = 1$ and develop different $k$, and the related clock cycle counts are recorded in Table 8 below.

**Table 8:** Performance comparison indicated by clock cycles for AES-128 implementations.

| | $k = 1$, $m = 1$ | $k = 2$, $m = 1$ | $k = 4$, $m = 1$ |
|---|---|---|---|
| Scheme in [WMCS20] | 2576186 | 2867778 | 4596884 |
| Our scheme | 1212684 | 1402647 | 2310900 |

It can be explicitly indicated from Table 8 that the "cost amortization" technique actually increases the overhead since the number of clock cycles becomes larger with increasing $k$ when $m$ is small, which validates our analysis in Section 3.4. We should claim that the implementations for both schemes are the general type that can be fed with any legal tuples of $k$, $m$ and $n$, and they are not fully accelerated as in Table 4. Therefore, the clock cycle results in the second column differ from the ones in Table 4 of Section 3.5.

# E Leakage Assessment results

Here the TVLA results for IPM2, IPM3 and IPM14 are provided below in Figure 11.

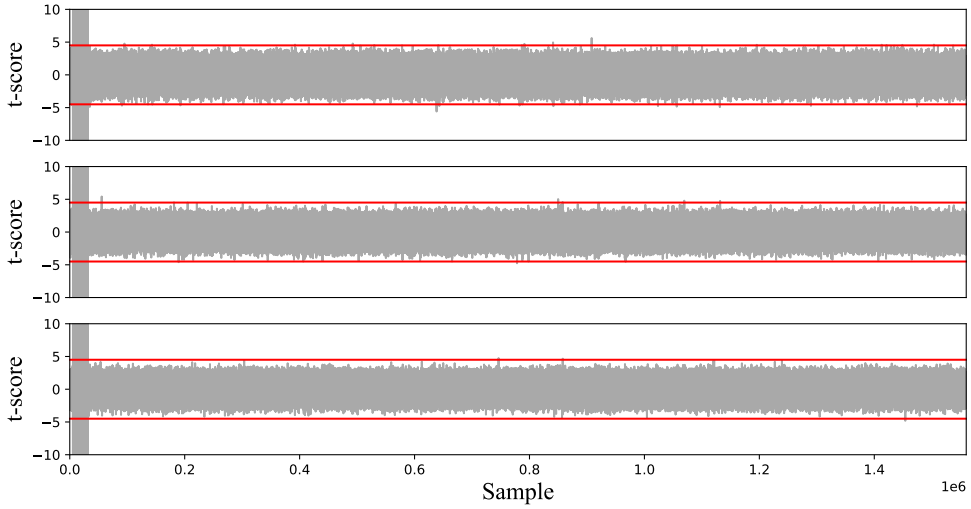
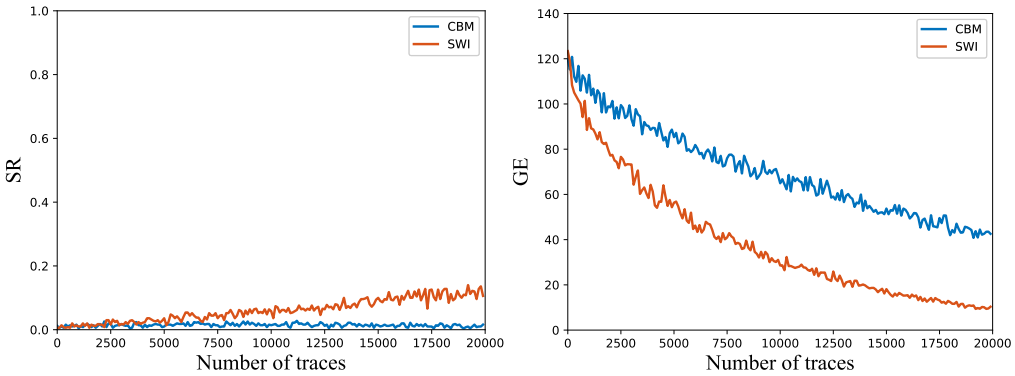



**Figure 11:** TVLA (t-test) results for IPM2 (top), IPM3 (middle) and IPM14 (bottom) with TRNG activated and sampling rate at 156 MHz. The red lines mark the ±4.5 threshold.

# F Security Loss of RE2

In this part, we provide the attack results of RE2 (introduced in Section 4.4) to depict the security loss introduced by the security bottleneck (the back-and-forth switch between code-based masking and additive sharing). Figure 12 depicts the SR and GE results of template attacks (see details in Section 4.4). Similarly, "CBM" (resp., " SWI") indicates that the target is the output (resp., the matrix $\mathbf{T}$) of L Gadget. It can be indicated from Figure 12 that the CBM case is more difficult to compromise than SWI case. This therefore implies that the matrix $\mathbf{T}$ of L Gadget again reduces the security level of encoding, which is consistent with the observations in Figure 9 for RE3. Note that SNR in this evaluation differs from that of TA in Figure 8. Hence although both targets are the encoding RE2 (orange line in Figure 8 and blue line in Figure 12), the results present different attack efficiency. Precisely, the one with higher SNR in Figure 8 (plotted in orange) is easier to compromise, e.g., 20,000 traces are enough to succeed, whilst exploiting POI with lower SNR in Figure 12 (plotted in blue) results in a much lower success rate in recovering subkey by utilizing up to 20,000 traces. This also provides strong proof that SNR indeed significantly affects the attack's difficulty in practice.




(a) SR of TA (b) GE of TA

**Figure 12:** SR and GE results targeting L Gadget for RE2.