Provable Order Amplification for Code-based Masking: How to Avoid Non-linear Leakages due to Masked Operations

Weijia Wang*, Yu Yu†‡ and François-Xavier Standaert§
*Universit catholique de Louvain,
EPL, ICTEAM, ELEN, Crypto Group,
B-1348, Louvain-la-Neuve, Belgium
Email: weijia.wang@uclouvain.be
†School of Electronic Information and Electrical Engineering,
Shanghai Jiao Tong University, China
Email: yyuu@sjtu.edu.cn
‡ Westone Cryptologic Research Center
§Universit catholique de Louvain,
EPL, ICTEAM, ELEN, Crypto Group,
B-1348, Louvain-la-Neuve, Belgium
Email: fstandae@uclouvain.be

***Abstract*—Code-based masking schemes have been shown to provide higher theoretical security guarantees than the Boolean masking. In particular, one interesting feature put forward at CARDIS 2016 and then analyzed at CARDIS 2017 is the so-called security order amplification: under the assumption that the leakage function is linear, it guarantees that an implementation performing only linear operations will have a security order in the bounded moment leakage model larger than $d-1$, where d is the number of shares.**

The main question regarding this feature is its practical relevance. First of all, concrete block ciphers do not only perform linear operations. Second, it may be that actual leakage functions are not perfectly linear (raising questions regarding what happens when one deviates from such assumptions). In this paper, we show that the issue of only linear operations can be provably avoided, and that it is possible to obtain security order amplification for any functionality to implement. We then show that (not so) slightly non-linear leakage functions do not annihilate the nice properties (i.e., that the code-based schemes we consider remain interesting compared to the Boolean masking). We conclude with a performance evaluation of the proposals, showing that the performance overheads are moderate for a reasonable number of shares (we studied when the number of the shares $d = 2, 3, 4$). Additionally, our results could be specified to the case of provable security for low entropy masking, which can be considered as a side bonus of our contributions. We give some preliminary results on how to construct low entropy masking schemes with provable high security order against linear leakage.

***Index Terms*—Side-channel attack; Masking; Order amplification; Bounded moment model; Probing model**

I. Introduction

Masking aims to provide a leakage-resilient implementation for the cryptographic algorithm, by encoding each sensitive variable into several shares. Typically, the security of this technique largely relies on the noise of the side-channel measurements. As proved in [1], [2], [3], [4], [5], given sufficient noise, the leakage of Boolean masking scheme exponentially decreases with the number of shares. However, this security guarantee does not always hold in the low noise scenario, and various high order attacks exploiting leakages of multiple intermediate variables (shares) have been proposed (see, e.g., [6], [7], [8], [9] for incomplete list of related studies). Thus, some recent literature tries to mitigate this problem by increasing the algebraic complexity of the shares (e.g., applying linear transformations to the shares of Boolean masking), resulting in various code-based masking schemes. Examples include inner product masking [10], [11], [12], polynomial masking [13], [14], [15], leakage squeezing [16], [17], [18] and affine masking [19]. Further, it is observed by [11] and further studied by [20] that this strategy (increasing the algebraic complexity of shares) provides a higher security order in linear leakage beyond the guarantees of the Boolean masking with the same number of shares, which is referred to as order amplification. However, there are some more promising but challenging tasks on this topic.

First, to the best of our knowledge, how to construct a full-fledged masking scheme (for both linear and non-linear operations) with order amplification is non-trivial and remains an open problem, because intermediate results in a masked operation may leak a non-linear function of the inputs. Second, the order amplification observed in [11] was only confirmed by the specific leakage function (Hamming weight), and it is necessary to provide a formal proof for any linear leakages. Finally, as the development of chip industry moves towards smaller technologies, the leakage of cryptographic implementations in nanoscale devices tends to be non-linear, and thus the analysis of order amplification under non-linear leakage is also very meaningful.

A. Our contributions

In this paper, we provide a formal study on the order amplification by giving the following contributions:

First, we give the first masking scheme with provable secure order amplification, i.e., for a certain number of shares, its security order in linear leakage is provably higher than that of the Boolean masking. Meanwhile, to reduce the amount of randomness used in masked multiplication, we provide another randomness-efficient (masked multiplication) Algorithm by putting an additional constraint on the sequential in which the operations (of the Algorithm) are carried out. Furthermore, we revisit the bit-probing model [20], which facilitates the security proof in the bounded moment model of the masking scheme in the presence of any linear leakage functions. Thus, the security order amplification of our proposed masking scheme can be derived from the security in the bit-probing model. Additionally, we show that the bit-probing model is also a good tool to design and analyze the low entropy masking and we provide a possible construction of low entropy masking with higher security order under linear leakages.

Second, we provide an analysis in the presence of non-linear leakage. It shows that, although non-linearity can break the order amplification in the high noise regime, the code-based masking still keeps a high level of security under restricted noise rates. Additionally, it is possible to reduce the non-linearity by some dedicated designs, in order to achieve order amplification for a larger range of noise levels.

Finally, as for the performance, we give some discussions on the efficient implementations of our new masking. In terms of software implementations, we discuss how to implement the Boolean matrix multiplication efficiently in cases of using different instruction sets. On the other hand, we also investigate the possibility of the parallel computation of the masked Boolean matrix multiplication (even with reduced randomness), and present its hardware implementation. At last, the software evaluations of the masking scheme based on the protection of AES are given.

II. Backgrounds

A. Notations

Let lowercases (e.g., i, x) denote integer variables or binary vectors, and let capital letters (e.g., A) be the Boolean (binary) matrices. $A(i,:)$ denotes the i-th row of matrix A, $x(i)$ denotes the i-th element of vector x, and $A(:,i)$ denotes the i-th column of matrix A. Let $A(i:j,k)$ (resp., $A(k,i:j)$) be the elements of k-th column (resp., k-th row) and i-th to j-th row (resp., i-th to j-th column). Let the bold lowercases (e.g., $\boldsymbol{x} = (x_1, x_2, \ldots)$) denote a vector whose elements are binary vectors, and let the bold capital letters be vectors of Boolean matrices (e.g., $\boldsymbol{A} = (A_1, A_2, \ldots)$)). Let E_m denote an $m \times m$ identity matrix, and A^{-1} and A^{T} denote the (generalized) inverse and transpose of the matrix A respectively. Finally, for Boolean matrices A and B, we denote their product and tensor product by $A \times B$ and $A \otimes B$ respectively, and for binary vectors x and y, we denote their product in Galois field as $x \cdot y$. The concatenation of two matrices A_1 and A_2 in column direction is denoted as $(A_1; A_2)$. For any integer $n \geq 1$, we use $[1, n]$ to denote the set of $\{1, \ldots, n\}$

B. Encoding of code-based masking

In this paper, we consider the encoding that applies linear transformations to the shares of the Boolean masking. That is, for an m-bit secret variable (say, x), we have $x = (A_1 \times x_1) \oplus (A_2 \times x_2) \oplus \ldots \oplus (A_d \times x_d)$, where $\{x_i\}_{i\in[1,d]}$ are shares and $\{A_i\}_{i\in[1,d]}$ are the public linear transformation matrices. Algorithm 1 shows the encoding, and it should be noted that this encoding can be seen as the generalized version of the encodings in masking schemes from some recent literature. For example, the inner product masking [10] is equivalent to the code-based masking whose encoding takes the matrices $\{A_i\}_{i\in[1,d]}$ corresponding to the Galois field multiplication (rather than any nonsingular Boolean matrices), and the matrix product masking [11] applies linear transformation only to one share.

Algorithm 1 Enc: encoding of the code-based masking

Require: an m-bit secret variable x, public nonsingular Boolean matrices $\boldsymbol{A} = \{A_1, \ldots, A_d\}$ and their inverses $\boldsymbol{A}^{-1} = \{A_1^{-1}, \ldots, A_d^{-1}\}$

Ensure: $\mathrm{Enc}(x) = \boldsymbol{x} = (x_1, \ldots, x_d)$ as the masked variables

1: **for** $i := 1$; $i < d$; $i{+}{+}$ **do**
2: Generate a random m-bit value x_{i+1}.
3: **end for**
4: $x_1 := A_1^{-1} \times (x \oplus \bigoplus_{i=2}^{d} (A_i \times x_i))$

It should be noted that, Massey et al. have proved that any linear sharing / masking may be represented by an encoding with a linear error correcting code [21]. For example, the Boolean masking can be represented by an encoding with a parity check code. The link between the property of the masking / sharing and the corresponding code has been studied, e.g. in [22], [15], [16]. However, in this paper, we focus on the constructions of secure masked operations, rather than the encoding of code-based maskings.

C. The bounded moment model in [23]

Recently, the bounded moment model was introduced in [23], which formalizes a weaker notion of security frequently used in the side-channel literature. Recall that in [23], the mixed moment at orders $o_1, o_2, \ldots, o_d$ for d-order masking (number of shares is d) are defined as $\prod_{i=1}^{d}(x_i - \mathrm{E}(x_i))^{o_i}$, where $\mathrm{E}(\cdot)$ is the expectation, $x_1, \ldots, x_d$ are shares and $o_i \in \{0, 1, \ldots, d\}$. The degree of a mixed moment is defined by $o = \sum_{i=1}^{d} o_i$. Thus, the security order of the bound moment model equals to the degree of the lowest key-dependent mixed moments (denoted by o_{min}).

The security in the bounded moment model is defined as follows.

Definition 1 (security in the bounded moment model): An implementation is secure at order o in the bounded moment model if the degree of the lowest key-dependent mixed moments of all the intermediate variables is at least o.

D. The probing model and bit-probing model

The probing model was proposed in [3], and it has been widely used in the security definitions for various masking schemes. Following is the definition of security in the probing model, where "perfect simulation" means that the answers of the simulator is identically distributed to the output of adversary.

Definition 2 (security in the probing model): An implementation is secure at order o in the probing model, if any o variables (e.g., the shares) in it can be perfectly simulated without any knowledge of the secret input.

The bit-probing model was introduced in [20] which simulates the bits (rather than the variables). Following is the definition of the security in the bit-probing model.

Definition 3 (security in the bit-probing model): An implementation is secure at order o in the bit-probing model, if any o bits in it can be perfectly simulated without any knowledge of the secret input.

We can see that the security in the probing model implies the security in the bit-probing model, but not vice versa. This is because that the simulator of the bit-probing model can simply call the simulator of the probing model. Meanwhile, we can take the encoding of the code-based masking as a counterexample showing that the security in the probing model does not imply the security in the bit-probing model with the same order (the security order in the bit-probing model is larger than that in the probing model in this case). Poussier et al. studied the connection between the bit-probing model and the bounded moment model in [20, Proposition 2]. We recall Poussier et al's result in Lemma 1, where a leakage function $\mathrm{L} : \mathbb{F}_2^l \to \mathbb{R}$ maps the value of an l-bit intermediate variable to a real number.

Lemma 1: For any leakage function of degree up to $\mathfrak{d}$, any implementation is secure at order $\lfloor \frac{\hat{d}}{\mathfrak{d}} \rfloor$ on the bounded moment model if it is secure at order $\hat{d}$ in the bit-probing model.

Lemma 1 shows that, under linear leakage assumption (i.e., the degree of the leakage function is 1, and the bits in a variable leak independently), the bit-probing model can be seen as a generalization of the bounded moment leakage model. That is, the security in the bit-probing model implies the security in the bounded moment model with the same order. Thanks to this generalization, we can investigate the security order of our constructions in Section III.

E. Security order amplification of the shares

One advantage of the code-based masking is the order amplification, which means that some code-based maskings may indicate a higher security order in linear leakage beyond the guarantees of the Boolean masking with the same number of shares. [11] gives some preliminary results suggesting that these phenomena are related to the minimum Hamming weights of the matrices $\{A_i\}_{i\in[1,d]}$ and $\{A_i^{-1}\}_{i\in[1,d]}$. Besides, [20] establishes a more concrete connection based on the dual distance between order amplification and the code properties of the masking scheme. We give the definition of order amplification:

Definition 4 (security order amplification): If a masking scheme has a property of order amplification, then its security order in the bounded moments model is larger than that in the probing model.

To give a more clear view of the order amplification, we compute the security order of the shares for code-based masking with 2 shares, the matrices of the code-based masking are chosen from the identity matrix E, $A' = ((1100; 0011; 1010; 1101)$ and $A'' = (1110; 1101; 1011; 0111)$. Assuming that the leakage function is the Hamming weight, table I shows the security orders of the shares in different models for different matrices and the property of order amplification.

However, despite the existence of order amplification for the shares with specific leakage function (Hamming weight), it is more challenging to provide a full-fledged masking scheme (including linear and non-linear operations) and a formal proof for any linear leakage functions. Specifically, for two masked variables, $\boldsymbol{x}$ and $\boldsymbol{y}$, known schemes cannot avoid the leakage of $x_i \cdot y_j$ for $i, j \in [1, d]$, which includes a non-linear function of x_i and y_i and may break the order amplification (because of the non-linear function of the input shares).

III. Constructions of masked operations

In this section, we present the construction of masked operations with security order amplification. We use d and $\hat{d}$ to denote the number of shares the amplified security order for the d-order encoding, and the latter one equals to the security order in the bounded moment model for linear leakage (by definition) and the bit-probing model (by Lemma 1). Clearly, we have $\hat{d} \geq d - 1$.

A. Mask refreshing

We first present the mask refreshing operation that will be used as the building block for the other operations. Mask refreshing is a re-randomized procedure to re-encode the secret variables, and our scheme is a modified version of the parallel refreshing in [23] by applying the linear transformations to the shares. Algorithm 2 is the masking refresh, and Figure 1 presents its process. For the reason of security, Algorithm 2 should be invoked for at least d times for a refreshing.

Algorithm 2 Refresh: mask refreshing

Require: shares $\boldsymbol{x} = \{x_i\}$ and $\boldsymbol{A} = \{A_i\}_{i\in[1,d]}$
Ensure: $\boldsymbol{y} = \{y_i\}_{i\in[1,d]}$
1: generate m-bit random numbers $\boldsymbol{r} = \{r_i\}$ for $i \in [1, d]$
2: $r'_i = A^{-1}_{(i-1) \mod d} \times A_i \times r_i$
3: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
4: $\quad x'_i := x_i \oplus r_i$
5: **end for**
6: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
7: $\quad y_i := x'_i \oplus r'_{(i+1) \mod d}$
8: **end for**

TABLE I
THE SECURITY ORDERS OF THE SHARES FOR DIFFERENT MATRICES

masking type	A_1	A_2	security order in the bounded moment model	security order in the probing model	order amplification
Boolean	-	-	1	1	no
Code-based	E	A'	1	2	yes
Code-based	E	A''	1	3	yes

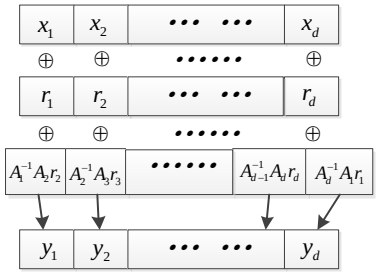

Fig. 1. Mask refreshing.

B. Linear operations

1) Addition.: The addition of two masked variables $\boldsymbol{x}$ and $\boldsymbol{y}$ is the same as what Boolean masking do. Algorithm 3 gives the masked addition of $\boldsymbol{x} = (x_1, \ldots, x_d)$ and $\boldsymbol{y} = (y_1, \ldots, y_d)$ that are encodings of two secret values x and y respectively.

Algorithm 3 SecAdd: addition of two masked variables

Require: two masked variables $\boldsymbol{x} = (x_1, ..., x_d)$ and $\boldsymbol{y} = (y_1, ..., y_d)$
Ensure: encoding of $x \oplus y$ (namely, $(\mathrm{Enc}(x \oplus y) = \boldsymbol{z} = (z_1, ..., z_d))$
1: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
2: $\quad z_i := x_i \oplus y_i$
3: **end for**

2) Linear transformation.: Another more complicated linear operation is the linear transformation, which can be represented by left-multiplying a matrix, i.e., $\mathrm{L}(x) = y = L \times x$, where $\mathrm{L}(\cdot)$ is the linear transformation, and matrix L is the corresponding linear transformation matrix. A straight-forward construction in the masked domain is left-multiplying each share of $\boldsymbol{x}$ by a pre-computed matrix, that is, $y_i = L_i \times x_i$, where the pre-computed matrix is $L_i = A_i^{-1} \times L \times A_i$. However, this construction may break the security order amplification. This is because that a bit of y_i may be related to multiple bits of x_i (i.e., y_i may be the output of a non-linear function of x_i), which means that the bits of x_i may not be independent even when one bit of y_i leaks.

In Algorithm 4 we present a new construction for the linear transformation, and we show its process in Figure 2. Basically, we insert the linear transformation into a refreshing operation. We refer to Section IV for the formal security proof. To prove the correctness of Algorithm 4. We have:

$$
\begin{aligned}
&\bigoplus_{i=1}^{d} A_i \times y_i \\
=&\bigoplus_{i=1}^{d} A_i \times \big(t_i \oplus (\bigoplus_{j=1}^{d'} r'_{i,j})\big) \\
=&\bigoplus_{i=1}^{d} A_i \times \big(A_i^{-1} \times L \times A_i \times u_i \oplus (\bigoplus_{j=1}^{d'} r'_{i,j})\big) \\
=&\bigoplus_{i=1}^{d} A_i \times \big(A_i^{-1} \times L \times A_i \times u_i \oplus (\bigoplus_{j=1}^{d'} r'_{i,j})\big) \\
=&\bigoplus_{i=1}^{d} A_i \times \big(A_i^{-1} \times L \times A_i \times (x_i \oplus \bigoplus_{k=0}^{d'} r_{i,k}) \oplus (\bigoplus_{j=1}^{d'} r'_{i,j})\big) \\
=&\bigoplus_{i=1}^{d} A_i \times (A_i^{-1} \times L \times A_i \times x_i) \\
=&L \times \bigoplus_{i=1}^{d} A_i \times x_i \\
=&L \times x \ .
\end{aligned}
$$

Algorithm 4 MaskedLinear: linear transformation to a masked variable

Require: $\boldsymbol{x} = (x_1, \ldots, x_d)$, $\boldsymbol{A} = \{A_i\}_{i\in[1,d]}$, linear transformation matrix L, security order for linear leakage $\hat{d}$
Ensure: linear transformation of $\boldsymbol{x}$ (i.e., $\boldsymbol{y} = (y_1, \ldots, y_d)$)
1: $L_i = A_i^{-1} \times L \times A_i$
2: $d' = \hat{d} - d + 1$
3: generate m-bit random numbers $\boldsymbol{r} = \{r_{i,k}\}_{i\in(1,\ldots,d),k\in(1,\ldots,d')}$
4: $r'_{i,k} = A_{(i-1) \bmod d}^{-1} \times A_i^{-1} \times L_i \times r_{i,k}$
5: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
6: $\quad u_i := x_i \oplus r_{i,1} \oplus \ldots \oplus r_{i,d'}$ ▷ First part of refreshing
7: **end for**
8: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
9: $\quad t_i := L_i \times u_i$ ▷ Linear transformation
10: **end for**
11: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
12: $\quad y_i := t_i \oplus r'_{i,1} \oplus \ldots \oplus r'_{i,d'}$ ▷ Second part of refreshing
13: **end for**

C. Non-linear operations

For the non-linear operations, we use the multiplication in $\mathrm{GF}(2^m)$ as a case study, which can be used as the building

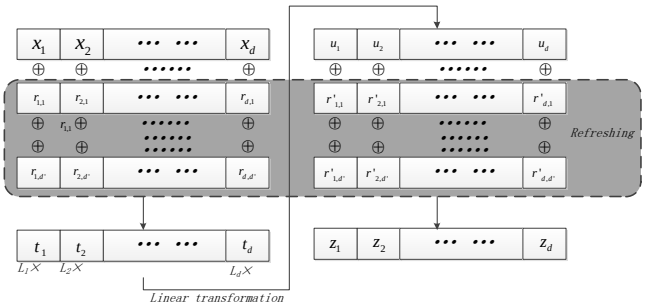

Fig. 2. Masked linear transformation.

block for many Sboxes such as the one in AES. For simplicity, we first consider the case that two inputs are independently masked, which can usually achieved by refreshing one of the inputs before the multiplication.

As discussed in [11], for any m-bits variables x and y, the multiplication in $\mathrm{GF}(2^m)$ can be rewritten as

$$x \cdot y = G \times (x \otimes y) \ , \tag{1}$$

where G is an $m \times m^2$ Boolean matrix. It should be noted that other non-linear operations of degree 2 can use a very similar construction and we refer to [11] for the "tensor product" representation of the bitand.

Without loss of generality, we consider the case for the mask order $d = 2$ (which can be easily generalized to a higher order). The variables x and y, are encoded as $\boldsymbol{x} = (x_1, x_2)$ and $\boldsymbol{y} = (y_1, y_2)$ respectively, and we have $x = (A_1 \times x_1) \oplus (A_2 \times x_2)$ and $y = (A_1 \times y_1) \oplus (A_2 \times y_2)$. Then the multiplication in the masked domain can be written as:

$$\begin{aligned}
&(A_1 \times x_1 \oplus A_2 \times x_2) \quad \cdot \quad (A_1 \times y_1 \oplus A_2 \times y_2) \\
=&\big((A_1 \times x_1) \cdot (A_1 \times y_1)\big) \ \oplus \ \big((A_1 \times x_1) \cdot (A_2 \times y_2)\big) \ \oplus \\
&\big((A_2 \times x_2) \cdot (A_1 \times y_1)\big) \ \oplus \ \big((A_2 \times x_2) \cdot (A_2 \times y_2)\big) \\
=&A_1 \times c_{1,1} \ \oplus \ A_1 \times c_{1,2} \ \oplus \\
&A_2 \times c_{2,1} \ \oplus \ A_2 \times c_{2,2} \ ,
\end{aligned} \tag{2}$$

where we define $c_{i,j}$ as:

$$c_{i,j} = A_i^{-1} \times (A_i \times x_i) \cdot (A_j \times y_j) \ . \tag{3}$$

Next, we need to compute $c_{i,j}$ without leaking neither $A_i \times x_i$ nor $A_j \times y_j$. By (1) and (3), we have:

$$c_{i,j} = A_i^{-1} \times G \times \big((A_i \times x_i) \otimes (A_j \times y_j)\big) \ . \tag{4}$$

Applying the following distributive property of tensor product:

$$(A_i \times x_i) \otimes (A_j \times y_j) = (A_i \otimes A_j) \times (x_i \otimes y_j) \ , \tag{5}$$

$c_{i,j}$ can be rewritten as:

$$c_{i,j} = \big(A_i^{-1} \times G \times (A_i \otimes A_j)\big) \times \underline{(x_i \otimes y_j)} \ . \tag{6}$$

Therefore, in Equation (6) the secret (i.e., $x_i \otimes y_j$) and public (i.e., $A_i^{-1} \times G \times (A_i \otimes A_j)$) parts are separated. We can pre-compute the public matrices: $M_{i,j} = A_i^{-1} \times G \times (A_i \otimes A_j)$, and have: $c_{i,j} = M_{i,j} \times (x_i \otimes y_j)$. However, as discussed in Section II-E, $c_{i,j}$ leaks a non-linear function of the inputs x_i and y_j, and thus the directly computing of $c_{i,j}$ may break the security order amplification.

1) Achieving provable security using more randomness: It should be noted that the variable $u_{i,j} = x_i \otimes y_j$ is a linear function of x_i or y_j [1] (which is a different situation from $c_{i,j}$). Thus, as shown in Figure 3, we design the mask multiplication in an alternative way to achieve the order amplification:

First (Step 1 and 2), we compute tensor products: $u_{i,j} = x_i \otimes y_j$, and then refresh them using the random bits generated from Algorithm 6, which is based on the scheme in [24] and [3], but in the tensor product domain, and the shares $\{u'_{i,i}\}_{i \in [1,d]}$ should also be refreshed. To this point, (because of the refreshing) any d^2-1 of the variables $u'_{i,j}$ is independent of both x and y.

Second (Step 3), we can get the $t_{i,j}$ simply by multiplying the pre-computed matrix: $t_{i,j} = M_{i,j} \times u'_{i,j}$. To this point, $t_{i,j}$ may indeed leak a non-linear function of $u'_{i,j}$. But (thanks to the refreshing in Algorithm 6) we have d^2 shares in this step, and any $d^2 - 1$ of them are independent of the input x or y. Thus, we still have the order amplification property.

At last (Step 4), similar to the construction in [24], we compress the variables $\{t_{i,j}\}_{i,j \in [1,d]}$ to get the final outputs. That is, the output for 2 shares is $z_1 = t_{1,1} \oplus t_{1,2}$ and $z_2 = t_{2,2} \oplus t_{2,1}$.

Algorithm 5 presents the process of masked multiplication mentioned above. To prove the correctness of Algorithm 5. We have:

$$\begin{aligned}
&\bigoplus_{i=1}^{d} A_i \times z_i \\
=&\bigoplus_{i=1}^{d} A_i \times (\bigoplus_{j=1}^{d} t_{i,j}) \\
=&\bigoplus_{i=1}^{d} A_i \times (\bigoplus_{j=1}^{d} M_{i,j} \times u'_{i,j}) \\
=&\bigoplus_{i=1}^{d} A_i \times \big(\bigoplus_{j=1}^{d} M_{i,j} \times (u_{i,j} \oplus \bigoplus_{k=1}^{d'} r_{i,j,k})\big) \\
=&\bigoplus_{i=1}^{d} A_i \times \big(\bigoplus_{j=1}^{d} M_{i,j} \times ((x_i \otimes y_j) \oplus \bigoplus_{k=1}^{d'} r_{i,j,k})\big) \\
=&\bigoplus_{i=1}^{d} A_i \times \big(\bigoplus_{j=1}^{d} M_{i,j} \times (x_i \otimes y_j) \oplus M_{i,j} \times \bigoplus_{k=1}^{d'} r_{i,j,k}\big) \\
=&\bigoplus_{i=1}^{d} A_i \times \big((\bigoplus_{i=1}^{d} A_i \times x_i) \cdot (\bigoplus_{i=1}^{d} A_i \times y_i)\big) \ ,
\end{aligned}$$

where $r_{i,j,k}$ denotes the random variable in the k-th call of the RefreshGen, for $k \leq d'$.

2) How to reduce the randomness?: As presented in the last sub-section, we can see that the main idea to guarantee the security at amplified order is that $\boldsymbol{t}$ is masked independently of the tensor product $\boldsymbol{u}$. To achieve this, Algorithm 5 applies the refreshing to $\boldsymbol{u}$, which uses $(d^2/2 + d - 1)m^2 d'$ random bits. It may not be practical since the cost of the random number generators are usually expensive [25]. In this sub-section, we

[1] We certainly need to assume that x and y are independently masked, which can also be achieved by using refreshing, i.e., d calls of Algorithm 2.

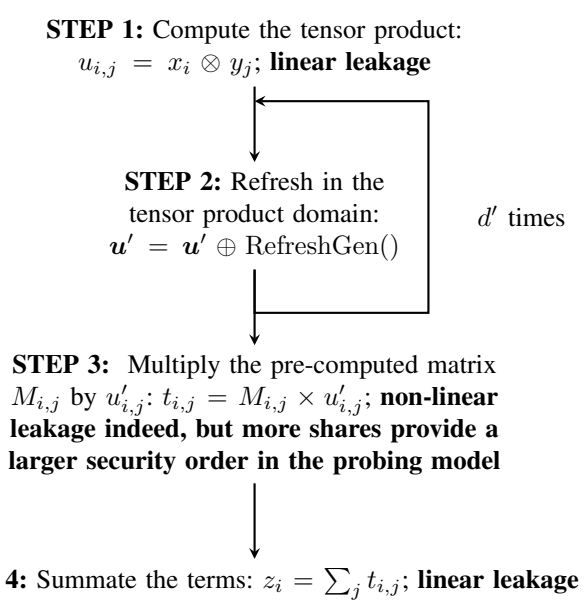

Fig. 3. Masked multiplication

Algorithm 5 MaskedMul: masked multiplication

Require: $\boldsymbol{x} = (x_1, \ldots, x_d)$, $\boldsymbol{y} = (y_1, \ldots, y_d)$, $\boldsymbol{M} = \{M_{i,j}\}$ for $i, j \in [1, d]$, security order for linear leakage $\hat{d}$
Ensure: encoding of $x \cdot y$ (i.e., $\boldsymbol{z} = (z_1, \ldots, z_d)$)

$d' = \hat{d} - d + 1$
for $i := 1$; $i \leq d$; i++ **do** ▷ STEP 1
  for $j := 1$; $j \leq d$; j++ **do**
    $u_{i,j} := (x_i \otimes y_j)$
  end for
end for
$\boldsymbol{u}' := \{u_{i,j}\}_{i,j \in [1,d]}$;
for $i := 1$; $i \leq d'$; i++ **do** ▷ STEP 2
  $\boldsymbol{u}' := \boldsymbol{u}' \oplus \text{RefreshGen}()$;
end for
for $i := 1$; $i \leq d$; i++ **do** ▷ STEP 3
  for $j := 1$; $j \leq d$; j++ **do**
    $t_{i,j} := M_{i,j} \times u'_{i,j}$
  end for
end for
for $i := 1$; $i \leq d$; i++ **do** ▷ STEP 4
  $z_i := \sum_{j=1}^{d} t_{i,j}$
end for

provide a randomness-efficient scheme with still preserving the order amplification property.

A naive way (with less randomness) to compute $t_{i,j}$ is to first calculate $c_{i,j} = M_{i,j} \times u_{i,j}$ and then to refresh them. But it may break the order amplification property, due to the issue of the straight-forward linear transformation described in Section III-B2. That is, a bit of $c_{i,j}$ may relates to multiple bits of $u_{i,j}$, which means that the bits of $u_{i,j}$ may not be independent even when one bit of $c_{i,j}$ leaks. Hence we need to compute the refreshed value of $M_{i,j} \times u_{i,j}$ without leaking any linear combinations of $u_{i,j}$ (except itself). We can abstract this problem as the computing of the product between an $m \times m^2$

Algorithm 6 RefreshGen: randomness generation in the tensor product domain

Require:
Ensure: a random matrix $\{r_{i,j}\}_{i,j \in [1,d]}$

generate m^2-bit random numbers $\boldsymbol{r} = \{r_{i,j}\}$ for $i, j \in [1, d]$ and $i < j$
for $i := 2$; $i \leq d$; i++ **do**
  for $j := 1$; $j < i$; j++ **do**
    $r_{i,j} = (M_{i,j}^{-1} \times A_i^{-1} \times A_j \times M_{j,i}) \times r_{j,i}$
  end for
end for
generate m^2-bit random numbers $r_{i,i}$ for $i \in [1, d-1]$
$u_{d,d} := M_{d,d}^{-1} \times A_d^{-1} \times \sum_{i=1}^{d-1} (A_i \times M_{i,i} \times r_{i,j})$

matrix M and a vector t and adding a vector of random values $\boldsymbol{r}$: $t = (M \times u) \oplus r$, without any non-linear leakages of u.

We first represent the matrix M as the concatenation of m matrices of size of $m \times m$: $M = (M_{(1)}, \ldots, M_{(m)})$. Similarly, the vector t can also be represented as the concatenation of m (column) vector of size of m: $t = (u_{(1)}; \ldots; u_{(m)})$. Then, the multiplication $M \times u$ can be rewritten as $(M_{(1)}, \ldots, M_{(m)}) \times (u_{(1)}; \ldots; u_{(m)}) = (M_{(1)} \times u_{(1)}) \oplus \ldots \oplus (M_{(m)} \times u_{(m)})$. By combining the refreshing with the above multiplication (to avoid any non-linear leakages), we have:

$$\begin{aligned} t =& (M \times t) \oplus r \\ =& (r \oplus (M_{(1)} \times u_{(1)})) \oplus (M_{(2)} \times u_{(2)}) \oplus \ldots \oplus \\ & (M_{(m)} \times u_{(m)}) \ . \end{aligned} \tag{7}$$

As shown in Figure 4, Equation 7 can be calculated in a secure and efficient manner. We apply the concept of Trichina masked AND gate [26], [27] to compute the $t_{i,j}$. [2] The internal values $v_1, \ldots, v_m$ are calculated one by one: the first internal value is calculated as $v_1 = (r \oplus (M_{(1)} \times u_{(1)})) = M_{(1)} \times (M_{(1)}^{-1} \times r \oplus u_{(1)})$, and the second one is $v_2 = v_1 \oplus (M_{(2)} \times u_{(2)}) = M_{(2)} \times (M_{(2)}^{-1} \times v_1 \oplus u_{(2)})$. Other internal values $v_3, \ldots, v_m$ can be calculated in a similar way and the last internal value v_m is the output. Figure 4 illustrates the process of the computation. It should be noted that the security order of the modified scheme in the bit-probing model is still d.

Furthermore, we illustrate that by applying of so-called "masked shares multiplication trick" [28], [29], [30], the masked multiplication can be more efficiently adapted to the situation when the inputs are not independently masked. For two variables x and y, and a random variable r, the masked shares multiplication trick computes the product of x and y by XORing r (i.e., $xy \oplus r$) in the way that one probing to the computation only relates to one variable from x and y. To achieve this, the masked shares multiplication trick computes the tuple $(\alpha_1, \alpha_2) = (xr \oplus r, x(r \oplus y))$ whose sum is indeed $xy \oplus r$. Algorithm 7 and 8 give the process of the above masked multiplication.

[2] It should be noted that the Trichina masked AND gate in previous literature is a first-order secure masking scheme, but our construction applies its idea that how to sequentially summate several terms with randomness in a secure manner.

Cautionary note: Although we still need to invoke the mask refreshing before the multiplication, only a single call of Algorithm 2 is sufficient, which is more efficient than the case without the masked shares multiplication trick. The latter case requires d calls of Algorithm 2.

Algorithm 7 MatrixMul: more efficient masked multiplication with less randomness

Require: $m \times 1$ variables x and y, an $m \times m^2$ matrix M and $m \times 1$ vector r
Ensure: $t = (M \times (x \otimes y)) \oplus r$
1: $(M_{(1)}, \ldots, M_{(m)}) := M$
2: $(u_{(1)}; \ldots; u_{(m)}) := u$
3: $v_0 = 0$
4: **for** $i := 1$; $i \leq m$; $i{+}{+}$ **do**
5: **if** $i = 1$ **then**
6: $v_{i-1} := v_{i-1} \oplus r$
7: **end if**
8: $v_i := M_{(i)} \times \mathrm{MSM}(x, y(i), M_{(i)}^{-1} \times v_{i-1})$
9: **end for**
10: $u := v_m$

Algorithm 8 MSM: masked shares multiplication trick

Require: two variables x and y, a random variable r
Ensure: $z = r \oplus x \cdot y$
1: $\alpha_1 = (y \cdot r) \oplus r$
2: $\alpha_2 = y \cdot (x \oplus r)$
3: $z = \alpha_1 \oplus \alpha_2$

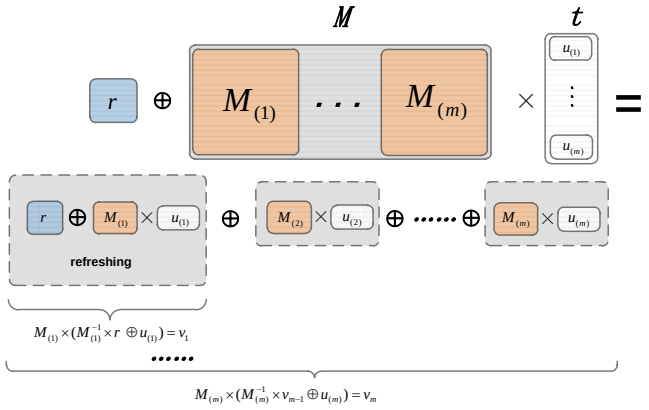

Fig. 4. Matrix multiplication with refreshing.

Algorithm 7 reduces the number of random bits by means of sequential evaluation of the linear transformation. Thus, as shown in Algorithm 9 we can construct a randomness-efficient masked multiplication by replacing the Steps 1, 2 and 3 of Algorithm 5 with the counterparts of Algorithm 7.

To prove the correctness of Algorithm 7. We have:

$$
\begin{aligned}
&\bigoplus_{i=1}^{d} A_i \times z_i \\
=&\bigoplus_{i=1}^{d} A_i \times (\bigoplus_{j=1}^{d} t_{i,j}) \\
=&\bigoplus_{i=1}^{d} A_i \times (\bigoplus_{j=1}^{d} M_{i,j} \times u_{i,j}) \\
=&\bigoplus_{i=1}^{d} A_i \times (\bigoplus_{j=1}^{d} M_{i,j} \times v_{i,j,m}) \\
=&\bigoplus_{i=1}^{d} A_i \times \big(\bigoplus_{j=1}^{d} M_{i,j} \times ((x_i \times y_j) \oplus r_{i,j})\big) \\
=&\bigoplus_{i=1}^{d} A_i \times \big((\bigoplus_{i=1}^{d} A_i \times x_i) \cdot (\bigoplus_{i=1}^{d} A_i \times y_i)\big) \ ,
\end{aligned}
$$

where $v_{i,j,m}$ is the output of Algorithm 7 for the computation of $x_i \cdot y_j \oplus r_{i,j}$

Algorithm 9 MaskedMul2: multiplication of two masked variables with less randomness

Require: $\boldsymbol{x} = (x_1, \ldots, x_d)$, $\boldsymbol{y} = (y_1, \ldots, y_d)$, $\boldsymbol{M} = M_{i,j}$ for $i, j \in [1, d]$
Ensure: encoding of $x \cdot y$ (i.e., $\boldsymbol{z} = (z_1, \ldots, z_d)$)
1: $\boldsymbol{y} = \mathrm{Refresh}(\boldsymbol{y})$
2: Generate m-bit random numbers $\{r_{i,j}\}_{i,j \in [1,d]}$ for $i \leq j$
3: $r_{i,j} := (A_i \times A_j) \times r_{j,i}$ for $i > j$
4: Generate m-bit random numbers $\{r_{i,i}\}_{i \in [1,d-1]}$
5: $r_{d,d} := A_d^{-1} \times \sum_{i=1}^{d-1} (A_i \times r_{i,i})$
6: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
7: **for** $j := 1$; $j \leq d$; $j{+}{+}$ **do**
8: $t_{i,j} := \mathrm{MatrixMul}(x_i, y_j, M_{i,j}, r_{i,j})$
9: **end for**
10: **end for**
11: **for** $i := 1$; $i \leq d$; $i{+}{+}$ **do**
12: $z_i := \sum_{j=1}^{d} t_{i,j}$
13: **end for**

IV. Security analysis in linear leakage

By using the bit-probing model and Lemma 1, we can prove the (amplified) security order of code-based masking under the bounded moment model. For (a simple) example, the encoding of 4-bit variable $x = (A \times x_1) \oplus x_2 \oplus \ldots \oplus x_d$, where A is a 4×4 matrix:$(0111; 1011; 1101; 1110)$. We can prove that every $(d+1)$-tuple of bits from its shares (i.e., $\{x_1, \ldots, x_d\}$) can be perfectly simulated without any knowledge of x, and thus the security order of the encoding for the bounded moment model under linear leakage is $d + 1$. This is because that (thanks to the linear code to the first share) to know only one bit of x, the adversary at least needs to know 3 bits of x_1 and 1 bit of each of $\{x_i\}_{i \in [2,d]}$.

In the rest of this section, we prove the order amplification of the linear and non-linear operations introduced in Section

III, i.e., Algorithm 4, 5 and 9. In the following theorem, we give the (amplified) security order of the operations under linear leakages.

Theorem 1: Given that the number of shares is d, and the security order of the encoding is $\hat{d}$ in the bounded moment model under linear leakages, the security orders of Algorithm 4, 5 and 9 equal to $\hat{d}$, $\min(\hat{d}, d(\hat{d}-d+1), d^2)$ and d respectively in the bounded moment model under linear leakages.

A. Security proof of masked linear transformation in Algorithm 4

We prove that every $\hat{d}$-tuple of bits in the masked linear transformation can be perfectly simulated without any knowledge of its secret input x. As $\boldsymbol{u}$ is a linear transformation of $\boldsymbol{t}$, the leakage of the bits from $\boldsymbol{t}$ (resp., $\boldsymbol{u}$) are not independent any more conditioned on even a single bit of leakage from $\boldsymbol{u}$ (resp., $\boldsymbol{t}$), Thus, the leakage of $\boldsymbol{t}$ (resp., $\boldsymbol{u}$) is non-linear in the leakage of $\boldsymbol{u}$ (resp., $\boldsymbol{t}$). Similarly, the linear transformation of the random variables $\boldsymbol{r}$ and $\boldsymbol{r}'$ are non-linear in leakage of themselves. Hence, in view of the security analysis, the process of masked linear transformation can be seen as a mask refreshing but some internal variables ($\boldsymbol{t}$, $\boldsymbol{u}$, $\boldsymbol{r}$, and $\boldsymbol{r}'$) have non-linear leakages. Thus, for any $i \in [1, d]$ and $j \in [1, d]$, the leakage of intermediate variable t_i (u_i, $r_{i,j}$ or $r'_{i,j}$) can be treated in an all-or-nothing manner: either all bits of this variable are leaked or nothing leaks at all (about this variable). That is, in case of even a single bit leakage about $\boldsymbol{t}$, $\boldsymbol{u}$, $\boldsymbol{r}$ and $\boldsymbol{r}'$ we just treat it as a complete leakage of the corresponding variable. For example, for any $i \in [1, d]$, the leakage of any bit from t_i can be treated as the (whole) leakage of the u_i. Let the numbers of leaking bits from $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{t}$ and $\boldsymbol{u}$ be d_x, d_y, d_t and d_u respectively, and let the sum of leaking bits from both $\boldsymbol{r}$ and $\boldsymbol{r}'$ be d_r. The simulator answers all adversary queries (value of leaking bits) based on the evaluation of the masked linear transformation when fed uniform and independent bits as input. To prove this simulator works, we give our analysis for different cases:

1) If $0 \leq d_r < d'$, then the input $\boldsymbol{x}$ and output $\boldsymbol{y}$ are still independently masked, and thus every $\hat{d}$-tuple of bits can be perfectly simulated without knowledge of x.
2) If $d_t = 0$ and $d_r \geq d'$, then we have that $d_x + d_y \leq \hat{d} - d_r \leq \hat{d} - d' \leq \hat{d} - (\hat{d} - d + 1) = d - 1$. In this case, Algorithm 4 can be seen as a mask refreshing, and thus every $\hat{d}$-tuple of bits can be perfectly simulated without knowledge of x.
3) If $d_t > 0$ and $d_r \geq d'$, then we have that $d_x + d_y \leq \hat{d} - d_r - d_t \leq d - d_t - 1$. In this case, we can see that every $\hat{d}$-tuple of bits can be perfectly simulated without knowledge of x.

Moreover, from the analysis we can see that every t-tuple of bits can be simulated from either t bits of the $\boldsymbol{x}$ or $t-(\hat{d}-d+1)$ variables of $\boldsymbol{x}$, for any $t \leq \hat{d}$.

B. Security proof of masked multiplication in Algorithm 5

Suppose that x and y are encoded independently, we prove that every $o_{min} = \min(\hat{d}, d(\hat{d} - d + 1), d^2)$-tuple of bits in the masked multiplication can be perfectly simulated without knowledge of x and y. Let the numbers of leaking bits from $\boldsymbol{z}$ be d_z, and we use the same definitions of d_t, d_u, d_r, d_x, d_y as that of Section IV-A. The simulator answers all adversary queries (value of leaking bits) based on the evaluation of the masked linear transformation when fed uniform and independent bits as input. In order to prove this simulator works, we give our analysis for different cases:

1) If $d_r \leq d' - 1$ bits, we can see that the refresh step is leakage free. In this case, we separate our analysis as follows:
   a) If the rest leaking bits come from the step 1, we have $d_t = 0$, $d_z = 0$ and $d_u + d_x + d_y \leq o_{min} \leq \hat{d}$. Since the input x and y are independently masked, every leaking bits can be perfectly simulated without knowledge of x or y.
   b) If the rest bits come from the step 3, we have $d_x = 0$, $d_y = 0$, $d_z = 0$ and $d_u + d_t \leq o_{min} \leq d^2$. Thanks to the refreshing, we have more shares in this step and every leaking bits is independent of the input x or y.
   c) If the rest bits come from the step 4, then we have $d_x = 0$, $d_y = 0$, $d_u = 0$ and $d_z + d_t \leq o_{min} \leq \min(d^2, \hat{d})$, similar to the case 1b every leaking bits can be perfectly simulated without knowledge of x or y.
   d) Combining 1a, 1b and 1c, we have that every o_{min} bits from the step 1, 3 and 4 of multiplication can be perfectly simulated without knowledge of x or y.
2) If not (i.e., $d_r \geq d'$), the refreshing is not leakage free, leaving at most $o_{min} - d'$ bits leaking from other variables. Then, the analysis is as follows:
   a) If $d_z = 0$, we have that $d_t + d_x + d_y + d_u + d_t \leq o_{min} - d' \leq \hat{d} - d' = d - 1$. Thus all the leaking bits can be perfectly simulated because of the independent masking of x and y.
   b) If not, we separate our analysis as follows:
      i) If $d_r < dd'$, then every $d - 1$ bits from $\boldsymbol{z}$ can be perfectly simulated without knowledge of x or y. Combined with 2a, we can see that all the leaking bits can be perfectly simulated without any knowledge of x and y.
      ii) If $d_r \geq dd'$, we can see that at least 1 bit from $\boldsymbol{z}$ may relate to x or y. Therefore, an upper bound on the number of the leaking bits in Algorithm 5 is $dd' = d(\hat{d} - d + 1)$.

Moreover, from the analysis we can see that every t-tuple of bits can be simulated from either t bits or $t - (\hat{d} - d + 1)$ variables of each of the inputs $\boldsymbol{x}$ and $\boldsymbol{y}$, or, for any $t \leq \min(\hat{d}, d(\hat{d} - d + 1), d^2)$.

C. Security proof of the masked shares multiplication trick in Algorithm 8

We provide the security proof of the masked shares multiplication where the input x and r are $m \times 1$ bit vectors, and y is

over GF(2), which is called by Algorithm 9. We aim to prove that any single bit in Algorithm 8 can be perfectly simulated from only 1 bit of x or y. Let d_x, d_y, d_r, d_{α_1} and d_{α_2} be the number of leaking bits of x, y, r, the computations of α_1 and α_2 respectively. The simulator answers all adversary queries (value of leaking bits) based on the evaluation of the masked linear transformation when fed uniform and independent bits as input. In order to show this simulator works, we give our analysis for different cases:

1) If $d_x = 1$ or $d_y = 1$, then we have $t_r = 0$, $d_{\alpha_1} = 0$ and $d_{\alpha_2} = 0$, and then one bit in x or y can be perfectly simulated from only 1 bit of x or y.
2) If $d_r = 1$, then we have $d_x = 0$, $d_y = 0$, $d_{\alpha_1} = 0$ and $d_{\alpha_2} = 0$, and then any bits in r can be perfectly simulated without knowing about x or y.
3) If $d_{\alpha_1} = 1$, then we have $d_x = 0$, $d_y = 0$, $d_r = 0$ and $d_{\alpha_2} = 0$, and then we have that any bits in α_1 can be perfectly simulated from y and r.
4) If $d_{\alpha_2} = 1$, then we have $d_x = 0$, $d_y = 0$, $d_r = 0$ and $d_{\alpha_1} = 0$, and then any 1 bit in α_2 can be perfect perfectly simulated from one bit of x or y.

D. Security proof of masked multiplication in Algorithm 9

Algorithm 9 calls Algorithm 7 and 8. The security proof of Algorithm 9 (which invokes Algorithm 7) is very similar to that of Algorithm 5, and we use the same definitions of d_t, d_r, d_x, d_y and d_z. Like $\boldsymbol{r}$, the linear transformation of the random variables $\boldsymbol{v}$ are non-linear in leakage of themselves. Thus, for any $i \in [1, d]$, the intermediate variable v_i is also leaked in an all-or-nothing manner, and we let the numbers of leaking bits from $\boldsymbol{v}$ be d_v. It should be noted that, for any $i \in [1, d]$, the leakage of any bit from v_i can be treated as the (whole) leakage of the v_i.

We can see that Algorithm 9 is secure at order $d - 1$ in the probing model. In Algorithm 7, the internal variables $\{v_i\}_{i\in[1,d]}$ are masked by random variable r, and thus any 1 bit from internal variables $\{v_i\}_{i\in[1,d]}$ and r can be perfectly simulated by 1 bit of x or y (in Algorithm 7). We can abstract the leakage of v_i and r by using the leakage of x and y in Algorithm 7: the probing of 1 bits in Algorithm 7 can be treated as the probing of 1 bit from x or y, and the probing of more than 1 bit in Algorithm 7 can be treated as the probing of (all bits of) x and y. The simulator answers all adversary queries (i.e., values of d leaking bits) based on the evaluation of the masked linear transformation when fed uniform and independent bits as input. In order to prove this simulator works, we give our analysis for different cases:

1) If the masking refresh process leaks at least 1 bit, then we divide our analysis as follows:
   a) If Algorithm 7 does not leak, we can see that $(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{t}$ are independent. Thus, every d-tuple of bits relate to at most d bits of input, which can be perfectly simulated without any knowledge of $\boldsymbol{x}$ or $\boldsymbol{y}$.
   b) If not, then let the number of calls of Algorithm 7 leak 1 bit be d_{mul}, and we have $d_{mul} \leq d - 1$. Let the number of the calls of Algorithm 7 leak more than 1 bit be d'_{mul}, and we have $2d'_{mul} + d_{mul} \leq d - 1$. At the same time, by the analysis in Section IV-C, the leaking bits in the summing process only related to d_r variables from $\boldsymbol{x}$ and d_t variables from $\boldsymbol{y}$, and d_{mul} variables from x or y. We can see that the total number of related input shares is $d_r + d_t + 2d'_{mul} + d_{mul} < d$. This is because that the leak of randomness only in the calling of Algorithm 7. Therefore, every d-tuple of bits of variables can be perfectly simulated without any knowledge of x and y.
2) If the masking refresh process does not leak, then $\boldsymbol{x}$ and $\boldsymbol{y}$ are independently masked, and we divide our analysis as follows:
   a) If Algorithm 7 does not leak, we can see that $(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{t}$ are independent. Thus, every d-tuple of bits relate to at most d bits of input, which can be perfectly simulated without any knowledge of $\boldsymbol{x}$ or $\boldsymbol{y}$.
   b) If not, then let the number of calls of Algorithm 7 leak 1 bit be d_{mul}, and we have $d_{mul} \leq d$. Let the number of the calls of Algorithm 7 leak more than 1 bit be d'_{mul}, and we have $2d'_{mul} + d_{mul} \leq d$. At the same time, by the analysis in Section IV-C, the leaking bits in the summing process only related to d_r variables from $\boldsymbol{x}$ and d_t variables from $\boldsymbol{y}$, and d_{mul} variables from x or y. We can see that the total number of related input shares is $d_r + d_t + d'_{mul} + d_{mul}$. We further separate the discussion into cases:
      i) If $d'_{mul} > 0$, then we can see that $d_r + d_t + d'_{mul} + d_{mul} < d$ since clearly $d_r + d_t + d'_{mul} + d_{mul} \leq d$.
      ii) If $d'_{mul} = 0$ and $d_t > 0$, then the total number of related input shares is $d_r + d_t + d'_{mul} + d_{mul} - 1 < d$.
      iii) If $d'_{mul} = 0$ and $d_t = 0$, then we can see that the total number of related input bits is at most d.

      Therefore, every d-tuple of bits of variables can be perfectly simulated without any knowledge of x and y. Moreover, from the analysis we can see that every t-tuple of bits can be simulated from either t bits or $t - 1$ variables of the inputs $\boldsymbol{x}$ and $\boldsymbol{y}$, for any $t \leq d$.

E. Security discussion on the composition of operations

As for the whole masking scheme, we give an informal security analysis for the composition of multiple masked operations.

Our analysis is similar to the works in [3], [1], [10], [11]. Assuming the security order in the bounded moment model for a single operation is at least $\tilde{d}$, we first only provide an analysis showing that the secure order of the composed masked operations is $\tilde{d}$ in the bounded moment model under linear leakages. Namely, the distribution of any tuple of $\tilde{d}$ or less intermediate bits in the masked cipher is independent

of any plaintext or key. This requires that, for a sequence of operations, the adversary could learn d_i intermediate bits for each operation, as long as $\sum_i d_i \leq \tilde{d}$. As shown in Figure 5, we consider w masked operations $\mathcal{F} = (f_1, \ldots, f_w)$ in sequence. Suppose that the adversary probes d_i intermediate bits in the i-th operation f_i and let ϕ_{w-1} be the inputs of the last operation f_w, then we can see that d_w probes of f_w relate to either at most d_w bits of ϕ_{w-1}, or at most $d_w - (\hat{d} - d + 1)$ variables of ϕ_{w-1}. Since ϕ_{w-1} is in turn the outputs of f_{w-1}, probing of d_w intermediate bits of f_w can be perfectly simulated from either d_w bits or $d_w - (\hat{d} - d + 1)$ variables of the inputs of f_{q-1}. This relies on the strategy that one refreshing is added before each multiplication. By adding the probing of d_{w-1} bits of f_{w-1}, the probing of $(d_w + d_{w-1})$ bits of f_w and f_{w-1} can be perfectly simulated from either $(d_w + d_{w-1})$ bits or $(d_w + d_{w-1} - (\hat{d} - d + 1))$ variables of the input of f_{w-1}. At last, by induction, we can conclude that the probing of $\sum_i d_i$ bits of the sequence of the operations can be perfectly simulated from either $\sum_i d_i$ bits or $(\sum_i d_i) - (\hat{d} - d + 1)$ variables of the inputs of the whole masked operations.

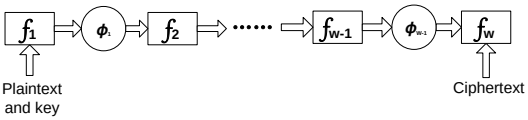

Fig. 5. A sequence of operations in consideration.

It should be noted that, we rely on the general strategy of adding one refresh before each multiplication in the setting of multiple inputs and outputs (see for example the "greedy strategy" discussed in [28, Section 3.1]). We leave the optimization of the number of necessary refresh gadgets for our scheme to compose as an interesting open problem.

As discussed in [10, Section 5.2], to handle the situation that adversary learns up to $\tilde{d}$ bits in each execution of the masked cipher (and thus he probes many values in a multiple-run setting), the masking refreshing should be carried out $\tilde{d}$ times on the secret key whenever the encryption / decryption starts over again.

F. Theoretical comparison to some previous related masking schemes

In order to give a summary of the theoretical contributions of our construction, we compare our masking scheme to some typical schemes in table II. We use d to denote the number of shares in the table. And the leakage rate denotes the value of security order divided by the complexity. For the comparison, we mainly consider the following related works:

- Ishai et al. proposed the first provable secure masking scheme (aka., private circuits) in the probing model, which is widely known as the ISW scheme [3]. The complexity of the ISW scheme is $O(d^2)$, and its leakage rate is $O(1/n)$. Besides the theoretical complexity, the ISW scheme is quite efficient because of its simplicity, ease of implementation and comparably low performance overhead.
- Andrychowicz et al. proposed a scheme with a leakage rate $O(1/\log(d))$, and its complexity is $O(\log(d)d^2)$ [31].
- Goudarzi1 et al. proposed an improved scheme with a leakage rate $O(1/\log(d))$ and complexity $O(\log(d)d)$ based on the number theoretic transform [32].
- Balasch et al. proposed a scheme with has the same complexity and theoretical security as ISW scheme but was illustrated to be more secure in practice [10].

V. Security analysis in practical scenarii

As observed in [33], [34], the leakage of different intermediate bits might be correlated, which can result in non-linearity of the leakage function. In general, leakages will become more complex and hard to interpret with technology scaling. Despite its inevitability, small (time or space) distance is one of the most important reasons for the significance of the coupling effect. Thus, it is possible to decrease the non-linearity of the leakage function through some dedicated designed implementation of the masked operations.

For software implementations, to mitigate the coupling effect, we can insert some dummy bits in between the intermediate bits. As shown in Figure 6, given that the intermediate bits are $(x(1), \ldots, x(m))$, each of which is separated by c dummy bits. Doing this, we can enlarge the distance (in space) between each bit and thus mitigate the coupling effect. Based on this strategy, Algorithm 10 presents a modified encoding that outputs shares with less non-linear leakages. The expand function $\mathrm{g} : \mathrm{F}_2^m \to \mathrm{F}_2^{cm}$ maps the i-th bit of the input to the $\big((i-1)c+1\big)$-th bit of the output, for $i \in [1, m]$, and set other bits of the output as constants. Then we apply the expand function on each shares: $\bar{x}_i = \mathrm{g}(x_i)$, for $i = [1, d]$. For hardware implementations, a designer can reduce the coupling effect by a reasonable arrangement of the netlist, in order to enlarge the distance of the gates or wires whose corresponding variables are encoded together.

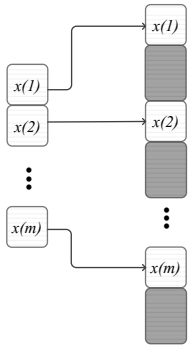

Fig. 6. A separate-by-space storage of intermediate bits in order to reduce the non-linearity of a leakage function

Furthermore, we show how the security order is affected by the non-linearity of the leakage function. We consider the polynomial representation of the leakage function. As stated in [35] and [36], any leakage function $\mathrm{F}(\cdot)$ on input $z \in \mathbb{F}_2^m$ can be represented in the following algebraic normal form:

$$\mathrm{F}(Z_i) = \alpha_0 + \sum_{u \in \mathbb{F}_2^m} \alpha_u Z_i^u + \varepsilon \ ,$$

TABLE II
THE SECURITY ORDERS OF THE SHARES FOR DIFFERENT MATRICES

masking schemes	complexity	leakage model	leakage rate	more secure in practice	provable order amplification
the scheme of [3]	$O(n^2)$	probing model	$O(1/n)$	no	no
the scheme of [31]	$O(n^2 \log(n))$	probing model	$O(1/\log(n))$	no	no
the scheme of [32]	$O(n \log(n))$	random probing model	$O(1/log(n))$	no	no
the scheme of [10]	$O(n^2)$	probing model	$O(1/log(n))$	yes	no
our scheme	$O(n^2)$	probing model, bit-probing model	$O(1/n)$	yes	yes

Algorithm 10 NewEnc: encoding with less non-linear leakages

Require: m-bit secret variable x, invertible matrices $\boldsymbol{A}$
Ensure: NewEnc$(x) = \bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_d)$ as the masked variables with smaller non-linearity leakage

1: **for** $i := 2; i \leq d$; $i{+}{+}$ **do**
2: generate an m-bit random value x_i
3: $\bar{x}_i := \mathrm{g}(A_i^{-1} \times x_i)$
4: **end for**
5: $x_1 := A_1^{-1} \times (x \oplus \bigoplus_{i=2}^{d} x_i)$
6: $\bar{x}_1 := \mathrm{g}(x_1)$

where coefficients $\alpha_u \in \mathbb{R}$, $Z_i = Z_{i,k^*}$, Z^u denotes $\prod_{j=1}^{m} Z(j)^{u(j)}$, $Z(j)$ (resp., $u(j)$) is the j-th bit of Z (resp., u), and ε denotes probabilistic noise. In order to analyze the leakage with different non-linear terms, we set the value of each coefficient α_u of order d as λ^d, where λ is in range of 0 to 1 whose value positively relates to the non-linearity of the leakage. That is, the leakage function is $\mathrm{F}(Z_i) = 1 + \sum_{u \in \mathbb{F}_2^m} \lambda^{\mathrm{HW}(u)} Z_i^u + \varepsilon$, where $\mathrm{HW}(u)$ denotes the Hamming weight of u.

In Figure 7, we present the mutual information of leakage functions for different λs. We can see that, as expected, the leakage (in terms of the mutual information) of the code-based masking is much less than that of the Boolean masking in all settings (i.e., in any noise level and λ). Furthermore, in view of the limited noise level, Figure 7 shows that the lowest the λ has (i.e., less non-linearity), the higher security order (reflected by the slope of the curves) is.

We remark that for cryptographic implementations in the real world, the leakage functions typically have small degrees or are dominated by linear terms. As a piece of supporting evidence, Poussier et al. give a linear regression-based analysis on the leakage function for 32-bit ARM Cortex-M4 microcontrollers [20, Section 6.2]. Linear regression [37] allows estimating how the manipulated data leaks at the bit level, and thus it is a suitable tool to estimate the linearity of the leakages. The results in [20] show that the linear terms of the leakage function in practice are significantly more dominant than the higher-order ones.

VI. Pushing the Limits of the Low Entropy Masking

In this sub-section, we show that the design of low entropy maskings [38] can also benefit from the bit-probing model. In low entropy maskings, an m-bit secret variable x is encoded into two shares: $\mathrm{Enc}(x) = (x_1, x_2)$, and (as the main difference from full masking schemes) the space of one share (either x_1 or x_2) is significantly smaller than 2^m. In this case, it is obvious that neither x_1 nor x_2 are independent of the secret variable x. But under linear leakage, it is still possible to achieve high order security. For (a very simple) example, let the encoding be: $x = x_1 \oplus x_2$ and $x_2 \in \{0^m, 1^m\}$, where 0^m (resp., 1^m) is the value that all of its bits are 0s (resp., 1s), then it can be seen that any 1 bit in the shares can be perfectly simulated without any knowledge of x. Thus, we can still achieve 1st order security in the bit-probing model. In this section, we conduct a study on the construction of better encodings for the low entropy masking with higher order security under linear leakages.

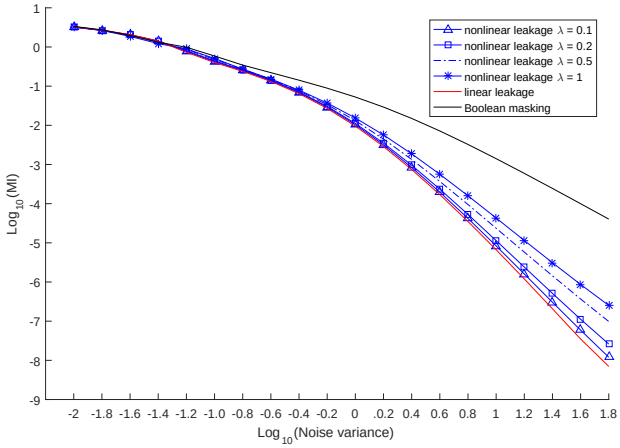

Fig. 7. Mutual information in function of the noise variance for different non-linear leakage functions (parameterized by λ).

In our construction, for $n < m$, the share x_2 is uniformly sampled from $\mathbb{F}_2^n$, and we have $x = x_1 \oplus (B \times x_2)$, where B is an $m \times n$ binary matrix. We can see that the security order (in linear leakage) is determined mainly by the choice of matrix B. To achieve the security in higher order, we give the following proposition:

Proposition 1: For $n < m$, we define the encoding of an m-bits variable x as $x = x_1 \oplus B \times x_2$, where x_2 is uniformly sampled from $\mathbb{F}_2^n$ and B is an $m \times n$ Boolean matrix. Then, the security order of this encoding in the bit-probing model is $h - 1$, where h is the minimum number of the linearly independent rows in the $(m+n) \times n$ matrix $C = (B; E_n)$.

Proof of Proposition 1. We aim to prove that every $(h-1)$-tuple of bits in the shares (x_1, x_2) can be perfectly simulated without any knowledge of the secret variable x. Let the

leaking bits from x_1 and x_2 be $\{x_1(i)\}_{i\in\mathcal{I}_1}$ and $\{x_2(j)\}_{j\in\mathcal{I}_2}$ respectively, where $\mathcal{I}_1$ and $\mathcal{I}_2$ are the subsets of $[1, m]$. Let the numbers of leaking bits from x_1 and x_2 be d_1 and d_2 respectively. We prove the existence of a simulator that can perfectly simulate the leakage of $h-1$ bits. The simulator answers all adversary queries by uniform bits. In order to prove this simulator works, our analysis goes as follows:

1) If $d_1 = 0$, every $(h-1)$-tupe bits can be perfectly simulated without any knowledge of the secret variable x, since x_2 is uniformly sampled and independent of x.
2) If not, the leaking bits come from both x_1 and x_2, and thus they can be represented as $\{x_1(i) = x(i) \oplus B(i,:) \times x_2\}_{i\in\mathcal{I}_1}$ and $\{x_2(j) = E_n(j,:) \times x_2\}_{j\in\mathcal{I}_2}$. We can see that any linear combination of at most $h-1$ leaking bits are still masked by at least one bit in x_2. Therefore any $h-1$ bits can be perfectly simulated without any knowledge of the secret variable x.

VII. Performance Evaluations

A. On improving the performance of the matrix multiplications

As matrix multiplications dominate the performance of masked multiplication, we discuss their performance and some possible improvements. Based on them, we furthermore discuss how to implement the matrix multiplication in hardware efficiently.

1) On the software implementation of matrix multiplications: Our discussions consider two types of matrix multiplication: the one presented in Algorithm 5 with the matrix with a size of $m \times m^2$ bits and the version with refreshing in Algorithm 7, which involves smaller matrices (i.e., m $m \times m$ matrices). The latter one can be implemented in software by using precomputed tables, i.e., for the multiplication between a fixed bit matrix A and an m-bit vector x, we can simply build a table for matrix A that maps the value of x to the value of $A \times x$.

Algorithm 5 is dominated by multiplication between a binary vector in tensor space and a fixed (pre-computed) $m \times m^2$ matrix. Wang et al. [11] introduce three different implementations of this matrix multiplication, which provide the trade-off between time / memory complexity and the size of instruction set of the processor: the time / memory complexities of the matrix multiplication can largely benefit from the 'popcnt' instruction, meanwhile we can still achieve good time complexities at the cost of some (reasonable) memory complexity without 'popcnt' instruction.

Another direction is that the improvement of the matrix multiplication relates to the shortest Linear Straight-Line Program (SLP) problem over $\mathrm{GF}(2)$. The SLP problem aims to find a program with the smallest number of lines that computes the multiplication between a fixed matrix and a vector, where every program line is of a certain form. Although Boyar et al. showed that this problem is NP-hard [39], there are still some heuristic methods can be used to solve some small instances [40].

2) Adjustment of the matrix multiplication for hardware implementation.: Algorithm 7 is essentially sequentially summing up internal values v_i, and thus it appears that Algorithm 7 can only be implemented serially. But it is possible to slightly modify it in order to achieve parallelism for the hardware implementation. Suppose that m is divisible by k and $\sum_{i=1}^{k} r_i = r$, in the following we present a modification of Algorithm 7 which can be computed in $\frac{m}{k}$ circles. As shown in Figure 8, in the first circle, the internal values $v'_1, \ldots, v'_k$ are calculated, where $v'_i = M_{(i)} \times (u_{(i)} \oplus M^{-1}_{(i)} \times r_i)$ for $i \in [1, k]$. Then, in the second circle, the internal values $v'_{1+k}, \ldots, v'_{2k}$ are calculated as $v'_{i+k} = M_{(i)} \times (u_{(i+k)} \oplus (M^{-1}_{(i+k)} \times v'_i \oplus r_i))$ for $i \in [1, k]$. Similarly, in j-th circle ($j \leq \frac{m}{k}$), the internal values $v'_{1+jk}, \ldots, v'_{(j+1)k}$ are calculated as $v'_{i+jk} = M_{(i+jk)} \times (u_i \oplus (M^{-1}_{(i+jk)} \times v'_{i+(j-1)k}))$ for $i \in [1, k]$, In the last circle, the sum of $v'_{m-k+1}, \ldots, v'_m$ is calculated and outputted. Similar to the case of software implementation, we can also apply the masked shares multiplication trick to the hardware implementation above. It is worth noting that, like the software implementation, the security order of this hardware implementation is also d.

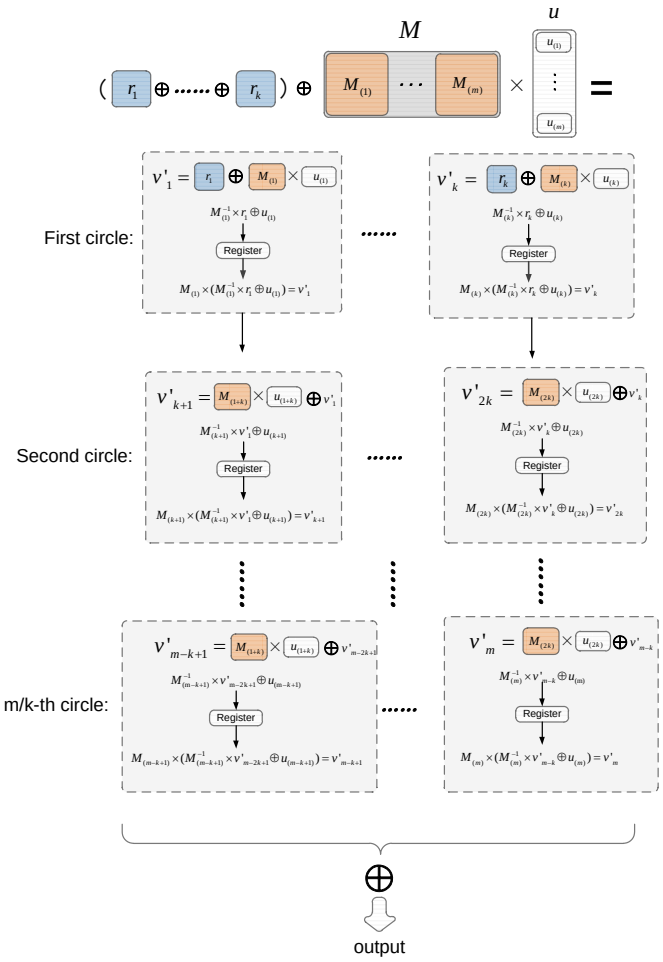

Fig. 8. Hardware adjusted matrix multiplications with refreshing.

B. Evaluation results

In order to compare the efficiency of our proposed masking scheme with Boolean masking, we apply them to protect the AES (without key expansion). We choose the efficient

version of multiplication, i.e., the Algorithm 9, in which the refreshing on one of the inputs can be omitted because every multiplication in the AES Sbox is placed after a linear operation, which already contains a refreshing. We implement the code-based masking for $d = 2, 3, 4$ and $m = 8$. The Boolean masking (the scheme of [4] with the SNI refreshing [41] and the masked shares multiplication trick) is also implemented. We implement the scheme in C language and run them on an Atmega 2560 processor. We use lookup tables wherever possible to decrease the time complexity. In terms of efficiency, there is significant room for improvement if implemented in assembly language. But we only analyze the relative performance slow-down compared to the Boolean masking.

We summarize the performances of our AES implementations in Table III. Besides, Table IV shows the performances of the masked Sbox using Boolean masking and code-based masking. The main performance loss of the code-based masking comes from the computation of Sbox, which includes several multiplications. In the Boolean masking, the performance of multiplication in $\mathrm{GF}(2^8)$ can benefit from lookup tables. That is, for any x_1 and x_2 in $\mathrm{GF}(2^8)\setminus\{0\}$, there exists e_1 and e_2 in $\mathrm{GF}(2^8)$ such that $x_1 = 3^{e_1}$, $x_2 = 3^{e_2}$ and their product equals $3^{e_1+e_2}$, where lookup tables between x_i and e_i can be built to accelerate the computation. This method is known as the exp-log multiplication in [42], and the authors of [42] also introduced other tabulated methods, such as Karatsuba multiplication and half-table multiplication. We choose the exp-log multiplication as the baseline for comparison because it has a good trade-off between table size and time complexity. Unfortunately, all the known acceleration methods in [42] do not work for the code-based masking, and the implementation of the latter one can only rely on Algorithm 7 (by tabulating the matrix multiplication of the smaller matrix $M_{(i)}$). This is the main reason for the performance loss.

TABLE III
PERFORMANCES OF OUR IMPLEMENTATIONS OF AES.

masking type	d	clock cycles	penalty factor to the Boolean masking with the same order	randomness
Boolean	2	524288	1	6400 bits
Boolean	3	856064	1	19200 bits
Boolean	4	1268736	1	38400 bits
Our Masking	2	1581056	3.02	23040 bits
Our Masking	3	3038208	3.55	44800 bits
Our Masking	4	4959232	3.91	71680 bits

Nevertheless, we emphasize that, in some cases (e.g., when the noise is low), it is worth sacrificing a certain amount of performance for a more secure implementation. This is because an adversary usually can achieve high SNR leakages from software implementation, which may enable him to make successful higher-order side-channel attacks [43]. Moreover, in the case of hardware, Section VII-A2 shows that our new masking is easy to be implemented in parallel. Thus, we believe that the better performance of our masking could be achieved in hardware implementation, and we leave its evaluation as a future study.

TABLE IV
PERFORMANCES OF OUR IMPLEMENTATIONS OF AES SBOX.

masking type	d	clock cycles	penalty factor to the Boolean masking with the same order	randomness
Boolean	2	2523	1	32 bits
Boolean	3	4289	1	96 bits
Boolean	4	6670	1	192 bits
Our Masking	2	8782	3.48	128 bits
Our Masking	3	16653	3.88	256 bits
Our Masking	4	27338	4.10	416 bits

VIII. CONCLUSION

In this paper, we provide a formal and systematic analysis of the order amplification in the bounded moment model. We present a construction of code-based masking scheme with provable order amplification, which can be implemented efficiently in both software and hardware. The proof of the new masking scheme in the bounded moment model can be achieved based on the bit-probing model [20]. Furthermore, our analysis shows that even in the presence of non-linear leakage, the security order amplification still holds for a limit level of noise. Another very interesting point is the provable higher order security of low entropy masking under linear leakages, in which we give some preliminary results and leave the rest as future work.

ACKNOWLEDGMENT

Yu Yu is supported by the National Natural Science Foundation of China (Grant Nos. 61872236, 61572192), the National Cryptography Development Fund MMJJ20170209, and Anhui Initiative in Quantum Information Technologies (Grant No.AHY150100). François-Xavier Standaert is a Senior Research Associate of the Belgian Fund for Scientific Research. This work has been funded in parts by the European Union through the H2020 project 731591 (REASSURE), the ERC project 724725 (SWORD) and the CHIST-ERA project SECODE.

REFERENCES

[1] G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, and P. Strub, "Verified proofs of higher-order masking," in *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*, 2015, pp. 457–485.

[2] A. Duc, S. Faust, and F. Standaert, "Making masking security proofs concrete - or how to evaluate the security of any leaking device," in *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*, 2015, pp. 401–429.

[3] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, 2003, pp. 463–481.

[4] M. Rivain and E. Prouff, "Provably secure higher-order masking of AES," in *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, 2010, pp. 413–427.

[5] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, 1999, pp. 398–412.

[6] E. Prouff, M. Rivain, and R. Bevan, "Statistical analysis of second order differential power analysis," *IEEE Trans. Computers*, vol. 58, no. 6, pp. 799–811, 2009.

[7] B. Gierlichs, L. Batina, B. Preneel, and I. Verbauwhede, "Revisiting higher-order DPA attacks:," in *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, 2010, pp. 221–234.

[8] F. Standaert, N. Veyrat-Charvillon, E. Oswald, B. Gierlichs, M. Medwed, M. Kasper, and S. Mangard, "The world is not enough: Another look on second-order DPA," in *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, 2010, pp. 112–129.

[9] T. S. Messerges, "Using second-order power analysis to attack DPA resistant software," in *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, 2000, pp. 238–251.

[10] J. Balasch, S. Faust, and B. Gierlichs, "Inner product masking revisited," in *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*, 2015, pp. 486–510.

[11] W. Wang, F. Standaert, Y. Yu, S. Pu, J. Liu, Z. Guo, and D. Gu, "Inner product masking for bitslice ciphers and security order amplification for linear leakages," in *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, 2016, pp. 174–191.

[12] J. Balasch, S. Faust, B. Gierlichs, C. Paglialonga, and F. Standaert, "Consolidating inner product masking," in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, 2017, pp. 724–754.

[13] L. Goubin and A. Martinelli, "Protecting AES with shamir's secret sharing scheme," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, 2011, pp. 79–94.

[14] E. Prouff and T. Roche, "Higher-order glitches free implementation of the AES using secure multi-party computation protocols," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, 2011, pp. 63–78.

[15] H. Chabanne, H. Maghrebi, and E. Prouff, "Linear repairing codes and side-channel attacks," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2018, no. 1, pp. 118–141, 2018.

[16] H. Maghrebi, S. Guilley, and J. Danger, "Leakage squeezing countermeasure against high-order attacks," in *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication - 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings*, 2011, pp. 208–223.

[17] C. Carlet, J. Danger, S. Guilley, and H. Maghrebi, "Leakage squeezing: Optimal implementation and security evaluation," *J. Mathematical Cryptology*, vol. 8, no. 3, pp. 249–295, 2014.

[18] C. Carlet, J. Danger, S. Guilley, H. Maghrebi, and E. Prouff, "Achieving side-channel high-order correlation immunity with leakage squeezing," *J. Cryptographic Engineering*, vol. 4, no. 2, pp. 107–121, 2014.

[19] G. Fumaroli, A. Martinelli, E. Prouff, and M. Rivain, "Affine masking against higher-order side channel analysis," in *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, 2010, pp. 262–280.

[20] R. Poussier, Q. Guo, F. Standaert, C. Carlet, and S. Guilley, "Connecting and improving direct sum masking and inner product masking," in *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, 2017, pp. 123–141.

[21] J. L. Massey, "Minimal codewords and secret sharing," p. 246C249, 04 1993.

[22] G. Castagnos, S. Renner, and G. Zémor, "High-order masking by using coding theory and its application to AES," in *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, 2013, pp. 193–212.

[23] G. Barthe, F. Dupressoir, S. Faust, B. Grégoire, F. Standaert, and P. Strub, "Parallel implementations of masking schemes and the bounded moment leakage model," in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, 2017, pp. 535–566.

[24] H. Groß, S. Mangard, and T. Korak, "An efficient side-channel protected AES implementation with arbitrary protection order," in *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, 2017, pp. 95–112. [Online]. Available: https://doi.org/10.1007/978-3-319-52153-4_6

[25] D. Goudarzi, A. Journault, M. Rivain, and F. Standaert, "Secure multiplication for bitslice higher-order masking: Optimisation and comparison," in *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, 2018, pp. 3–22.

[26] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, 2015, pp. 764–783.

[27] E. Trichina, "Combinational logic design for AES subbyte transformation on masked data," *IACR Cryptology ePrint Archive*, vol. 2003, p. 236, 2003.

[28] G. Cassiers and F. Standaert, "Improved bitslice masking: from optimized non-interference to probe isolation," *IACR Cryptology ePrint Archive*, vol. 2018, p. 438, 2018. [Online]. Available: https://eprint.iacr.org/2018/438

[29] J. Coron, E. Prouff, M. Rivain, and T. Roche, "Higher-order side channel security and mask refreshing," in *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, 2013, pp. 410–424. [Online]. Available: https://doi.org/10.1007/978-3-662-43933-3_21

[30] C. Carlet, E. Prouff, M. Rivain, and T. Roche, "Algebraic decomposition for probing security," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, 2015, pp. 742–763. [Online]. Available: https://doi.org/10.1007/978-3-662-47989-6_36

[31] M. Andrychowicz, S. Dziembowski, and S. Faust, "Circuit compilers with o(1/\log (n)) leakage rate," in *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, 2016, pp. 586–615. [Online]. Available: https://doi.org/10.1007/978-3-662-49896-5_21

[32] D. Goudarzi, A. Joux, and M. Rivain, "How to securely compute with noisy leakage in quasilinear complexity," in *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, 2018, pp. 547–574. [Online]. Available: https://doi.org/10.1007/978-3-030-03329-3_19

[33] M. Renauld, F. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, "A formal study of power variability issues and side-channel attacks for nanoscale devices," in *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, 2011, pp. 109–128.

[34] T. D. Cnudde, B. Bilgin, B. Gierlichs, V. Nikov, S. Nikova, and V. Rijmen, "Does coupling affect the security of masked implementations?" in *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, 2017, pp. 1–18.

[35] C. Whitnall, E. Oswald, and F. Standaert, "The myth of generic DPA...and the magic of learning," in *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, 2014, pp. 183–205.

[36] C. Carlet, "Boolean functions for cryptography and error correcting codes," *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, vol. 2, pp. 257–397, 2010.

[37] W. Schindler, K. Lemke, and C. Paar, "A stochastic model for differential side channel cryptanalysis," in *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, 2005, pp. 30–46.

[38] V. Grosso, F. Standaert, and E. Prouff, "Low entropy masking schemes, revisited," in *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, 2013, pp. 33–43.
[39] J. Boyar, P. Matthews, and R. Peralta, "On the shortest linear straight-line program for computing linear forms," in *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, 2008, pp. 168–179.
[40] J. Boyar and R. Peralta, "A new combinational logic minimization technique with applications to cryptology," in *Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings*, 2010, pp. 178–189.
[41] S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud, "Randomness complexity of private circuits for multiplication," in *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, 2016, pp. 616–648.
[42] D. Goudarzi and M. Rivain, "How fast can higher-order masking be in software?" in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, 2017, pp. 567–597.
[43] A. Battistello, J. Coron, E. Prouff, and R. Zeitoun, "Horizontal side-channel attacks and countermeasures on the ISW masking scheme," in *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, 2016, pp. 23–39.

François-Xavier Standaert was born in Brussels, Belgium in 1978. He received the Electrical Engineering degree and PhD degree from the Université catholique de Louvain, respectively in 2001 and 2004. In 2004-2005, he was a Fulbright visiting researcher at Columbia University, Department of Computer Science, Network Security Lab and at the MIT Medialab, Center for Bits and Atoms. In 2006, he was a founding member of IntoPix s.a. From 2005 to 2008, he was a post-doctoral researcher of the UCL Crypto Group and a regular visitor of the two aforementioned laboratories. Since 2008, he is associate researcher of the Belgian Fund for Scientic Research (F.R.S.-FNRS) and professor at the UCL Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM). In 2010, he was program co-chair of CHES (which is the agship workshop on cryptographic hardware). In 2011, he was awarded a Starting Independent Research Grant by the European Research Council. In 2016, he has been awarded a Consolidator Grant by the European Research Council. His research interests include cryptographic hardware and embedded systems, low power implementations for constrained environments (RFIDs, sensor networks, ...), the design and cryptanalysis of symmetric cryptographic primitives, physical security issues in general and side-channel analysis in particular.

Weijia Wang was born in China in 1988. He received his PhD degree from Shanghai Jiao Tong University. He is currently a post-doctoral researchers at the UCL Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM). His research interests include side channel analysis, leakage resilient, cryptographic implementations and hardware security.

Yu Yu was born in China in 1981. He received his BSc from Fudan University in 2003 and his PhD from Nanyang Technological University in 2006 respectively. He is currently a Professor at Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include theoretical aspects of cryptography such as leakage-resilient cryptography and post-quantum cryptography.