IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2020, No. 2, pp. 128–171. DOI:10.13154/tches.v2020.i2.128-171

Efficient and Private Computations with Code-Based Masking

Weijia Wang, Pierrick Méaux, Gaëtan Cassiers
and François-Xavier Standaert

Crypto Group, ICTEAM Institute, UCLouvain, Louvain-la-Neuve, Belgium.
firstname.lastname@uclouvain.be

Abstract. Code-based masking is a very general type of masking scheme that covers Boolean masking, inner product masking, direct sum masking, and so on. The merits of the generalization are twofold. Firstly, the higher algebraic complexity of the sharing function decreases the information leakage in "low noise conditions" and may increase the "statistical security order" of an implementation (with linear leakages). Secondly, the underlying error-correction codes can offer improved fault resistance for the encoded variables. Nevertheless, this higher algebraic complexity also implies additional challenges. On the one hand, a generic multiplication algorithm applicable to any linear code is still unknown. On the other hand, masking schemes with higher algebraic complexity usually come with implementation overheads, as for example witnessed by inner-product masking. In this paper, we contribute to these challenges in two directions. Firstly, we propose a generic algorithm that allows us (to the best of our knowledge for the first time) to compute on data shared with linear codes. Secondly, we introduce a new amortization technique that can significantly mitigate the implementation overheads of code-based masking, and illustrate this claim with a case study. Precisely, we show that, although performing every single code-based masked operation is relatively complex, processing multiple secrets in parallel leads to much better performances. This property enables code-based masked implementations of the AES to compete with the state-of-the-art in randomness complexity. Since our masked operations can be instantiated with various linear codes, we hope that these investigations open new avenues for the study of code-based masking schemes, by specializing the codes for improved performances, better side-channel security or improved fault tolerance.

Keywords: Side-channel attacks · Masking · Linear Codes · Code-based Masking

1 Introduction

Masking is one of the most investigated countermeasures against side-channel attacks [Koc96, KJJ99]. From a high-level overview, the masking approach randomly encodes (via a mapping called encoder) each secret-dependent sensitive variable into n shares, such that any d shares are independent of the sensitive variables, for an integer $d < n$.[1] The cryptographic implementation is built such that any tuple of d intermediate variables brings no information about the secret, typically by implementing each part of the computation with a so-called masked gadget. Even though all the intermediates variables are leaky, in practice, the leakages are noisy and the complexity of extracting useful information about the secret increases exponentially in d [CJRR99, PR13, DDF14].

Code-based masking is a very general type of masking scheme. Beside the Boolean masking (or additive masking) that is its simplest instance, the other ones with higher

[1] In many common cases, we have $d = n - 1$, but it is not always true, *e.g.,* for code-based masking.

Received: 2019-10-15 Accepted: 2019-12-15 Published: 2020-03-02

algebraic complexity of the sharing (than Boolean masking) include polynomial masking [GM11, PR11, GSF13, CMP18], leakage squeezing [MGD11, CDGM14, CDG+14], Inner Product (IP) masking [BFG15, BFG+17, CCG+19] and (orthogonal) Direct Sum Masking (DSM) [BCC+14, CG16, PGS+17]. It has been shown in [BFGV12] that IP masking can be viewed as a generalization of simpler encoders such as used in Boolean, affine and polynomial maskings. Furthermore, the work in [PGS+17] shows that DSM can be viewed as a generalization of the encoders of IP masking. Therefore, to the best of our knowledge, DSM has the most generalized encoder up to now.

The first merit of code-based masking is about security. The higher algebraic complexity of its encodings can provide an improved concrete security against side-channel attacks. For instance, it decreases the information leakages observed in "low noise conditions" [BFG15, BFGV12, FMPR10, GM11, PR11]. Also, it can improve the "statistical security order" (or security order in the bounded moment leakage model [BDF+17]) in case of linear leakage functions [CDG+14, GSP13, WSY+16], which is also known as the "order amplification". The second merit is about fault resistance. For example, DSM [BCC+14] offers protection against both side-channel and fault attacks for the encoded variables, by using an encoding function mixing randomness and sensitive data. Recently, Cheng et al. [CCG+19] also devised a new inner product masking scheme, which enables fault detection.

However, one important limitation of the code-based masking schemes is that computing with them is challenging. In particular, it is pointed out in [PGS+17] that an efficient multiplication algorithm for DSM is still an open challenge, and known IP masking schemes are not as efficient as Boolean masking schemes.

1.1 Contributions

Our first contribution is to present efficient algorithms that perform masked linear and nonlinear operations for the generic encoder, which are both proven to satisfy the Strong Non-Inference (SNI) notion [BBD+16]. The generic encoder is a generalization of the former code-based masking which relaxes the "direct sum" constraint of DSM.[2] The proposed masked operations are compatible with any choice of linear code that is adopted in the generic encoder, and thus we believe they have a good potential for improvement by specializing them to specific linear codes. For example, there is a large room to improve the (randomness and computational) cost of our scheme by choosing linear codes with sparse generating matrices or computation-friendly structures.

The second contribution relates to cost amortization, which has a similar spirit as the work of polynomial masking with packed secret sharing technique [GSF13]. It also shares some goals with works like [FPS17, IKL+13] which focus on reusing the randomness over different sub-parts of a masked implementation. Our code-based masking is able to encode multiple secrets together into one codeword and compute masked operations over these secrets in parallel, which amortizes both the computational and randomness cost. In order to analyze this property, we apply the new masking scheme to protect the AES block cipher, and try to take advantage of the fact that the S-boxes of the AES block cipher are computed over 16 internal states in parallel. We show that the randomness cost of implementations of the full AES-128 can be asymptotically smaller than state-of-the-art solutions [BBP+16, BGR18, CS19], especially when the order of security increases. Besides, a computational complexity evaluation shows that the computational cost of the new scheme is comparable to the state-of-the-art [BBP+16].

[2] The relaxation from the "direct sum" constraint of the DSM allows better fault resistance. We also show that the probing security of the generic encoder can be larger than the one claimed by DSM.

1.2 Organization

We begin this paper by presenting the background and preliminaries including private circuits, security notions and linear codes (see Section 2). We introduce the generic encoder in Section 3 and, in Sections 4 and 5 propose the corresponding masked operations. Section 6 illustrates the property of cost amortization and the application to AES. In the last section, we conclude the paper with several promising future research directions.

2 Preliminaries

2.1 Notations

We denote the finite field of order q as $\mathbb{F}_q$, and we represent field elements by lower-case letters. $\oplus$ and $\ominus$ denote the addition and subtraction over the finite field. We use $\sum$ for the summation over any field and extend these notations to vector spaces defined over these fields. For a natural number n we denote with $[n]$ the set of integers from 1 to n, both included.

Let calligraphies (*e.g.,* $\mathcal{I}$) denote sets, and $|\mathcal{I}|$ denote the cardinal of the set $\mathcal{I}$. Let bold capital letters (*e.g.,* $\mathbf{A}$) be matrices over $\mathbb{F}_q^{r\times c}$, for row and column counts being r and c respectively. $\mathbf{A}[i,*]$ (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 $\mathbf{A}$. Let $\mathbf{A}^{-1}$ and $\mathbf{A}^{\mathrm{T}}$ denote the (generalized) inverse and transpose of $\mathbf{A}$ respectively. For a $r\times c$ matrix $\mathbf{A}$ and a set $\mathcal{I}\subseteq[r]$ (resp., $\mathcal{J}\subseteq[c]$), $\mathbf{A}[\mathcal{I},*]$ (resp., $\mathbf{A}[*,\mathcal{J}]$) denotes the submatrix of $\mathbf{A}$ made up of the rows (resp., columns) indexed by $\mathcal{I}$ (resp., $\mathcal{J}$). For matrices $\mathbf{A}$ and $\mathbf{B}$, we denote their product as $\mathbf{A}\times\mathbf{B}$, or in short $\mathbf{AB}$ in non-ambiguous cases. For two matrices $\mathbf{A}$ and $\mathbf{B}$, $[\mathbf{A},\mathbf{B}]$ (resp., $[\mathbf{A};\mathbf{B}]$) is the concatenation of the columns (resp., rows) of $\mathbf{A}$ and $\mathbf{B}$. Let bold lower cases (*e.g.,* $\mathbf{x}$) be the vectors over $\mathbb{F}_q^{|\mathbf{x}|}$, where $|\mathbf{x}|$ denotes the length of the vector, $\mathbf{x}[i]$ denotes the i^{th} element of $\mathbf{x}$, and $\mathbf{x}[i\!:\!j]$ denotes the vector made up of the i^{th} to j^{th} elements of $\mathbf{x}$. Unless otherwise noted, we assume the vectors are row vectors in this paper, and the column vectors are denoted as $\mathbf{x}^{\mathrm{T}}$. We use $\mathbf{e}_i$ to denote a canonical vector: its i^{th} element is 1 and all of its other elements are 0s.

2.2 Tensor product

For two matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ in $\mathbb{F}_q^{r_1\times c_1}$ and $\mathbb{F}_q^{r_2\times c_2}$ respectively, we define their tensor product as the matrix over $\mathbb{F}_q^{r_1r_2\times c_1c_2}$:

$$\mathbf{X}_1\otimes\mathbf{X}_2=\begin{bmatrix}\mathbf{X}_1[1,1]\mathbf{X}_2 & \dots & \mathbf{X}_1[1,c_1]\mathbf{X}_2\\ \vdots & \ddots & \vdots\\ \mathbf{X}_1[r_1,1]\mathbf{X}_2 & \dots & \mathbf{X}_1[r_1,c_1]\mathbf{X}_2\end{bmatrix}.$$

Similarly, the tensor product of two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathbb{F}_q^{\ell_1}$ and $\mathbb{F}_q^{\ell_2}$ respectively is a vector in $\mathbb{F}_q^{\ell_1\ell_2}$ defined as:

$$\mathbf{x}_1\otimes\mathbf{x}_2=\big[\mathbf{x}_1[1]\mathbf{x}_2[1],\dots,\mathbf{x}_1[1]\mathbf{x}_2[\ell_2],\dots,\mathbf{x}_1[\ell_1]\mathbf{x}_2[1],\dots,\mathbf{x}_1[\ell_1]\mathbf{x}_2[\ell_2]\big].$$

We are also interested in the tensor product of $\mathbf{x}_1^{\mathrm{T}}$ and $\mathbf{x}_2$ (sometimes called the outer product of $\mathbf{x}_1$ and $\mathbf{x}_2$), which results in a matrix in $\mathbb{F}_q^{\ell_1\times\ell_2}$ defined as:

$$\mathbf{x}_1^{\mathrm{T}}\otimes\mathbf{x}_2=\begin{bmatrix}\mathbf{x}_1[1]\mathbf{x}_2[1] & \dots & \mathbf{x}_1[1]\mathbf{x}_2[\ell_2]\\ \vdots & \ddots & \vdots\\ \mathbf{x}_1[\ell_1]\mathbf{x}_2[1] & \dots & \mathbf{x}_1[\ell_1]\mathbf{x}_2[\ell_2]\end{bmatrix}.$$

In the rest of this sub-section, we provide several properties of the tensor product that will be used in different proofs through the paper.

Proposition 1. *Let* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *be two vectors and let* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *be two matrices such that* $|\mathbf{x}_1|$ *equals the row count of* $\mathbf{A}_1$ *and* $|\mathbf{x}_2|$ *equals the row count of* $\mathbf{A}_2$*, then we have:*

$$(\mathbf{x}_1 \otimes \mathbf{x}_2)(\mathbf{A}_1 \otimes \mathbf{A}_2) = (\mathbf{x}_1\mathbf{A}_1) \otimes (\mathbf{x}_2\mathbf{A}_2).$$

For a vector $\mathbf{v} \in \mathbb{F}_q^{r^2}$ and an index i such that $1 \leq i \leq r$, let $\mathbf{v}_i \stackrel{\text{def}}{=} \mathbf{v}[1+r(i-1):ri]$, and for a matrix $\mathbf{M} \in \mathbb{F}_q^{r^2 \times c}$, let $\mathbf{M}_i \stackrel{\text{def}}{=} \mathbf{M}[1+r(i-1):ri, *]$.

Proposition 2. *Let* $\mathbf{v} \in \mathbb{F}_q^{r^2}$ *and* $\mathbf{M} \in \mathbb{F}_q^{r^2 \times c}$*, then* $\mathbf{vM} = \sum_{i=0}^{r}(\mathbf{v}_i\mathbf{M}_i)$.

Proposition 3. *A matrix* $\mathbf{X} \in \mathbb{F}_q^{r \times c}$ *can be rewritten as* $\sum_{i=1}^{c}(\mathbf{X}[*, i] \otimes \mathbf{e}_i)$.

2.3 Equivalence of distributions

Later in this section we will see that our security ibvestigations involve the study of joint probabilistic distributions of a set of variables over $\mathbb{F}_q$. In contrast, from the error correcting code perspective it is more convenient to consider the properties of combinations of codewords' elements, where each combination gives a single element. The purpose of the following lemma is to connect these two approaches, proving that considering the joint distribution of d variables or the distributions of all the linear combinations of these variables is equivalent. More precisely, it shows that a discrete joint distribution over a field is characterized by the knowledge of all the linear combinations (with values in the field) of its marginal distributions.

Lemma 1 (Distributions equivalence)**.** *Let* X_i *for* $i \in [n]$*, be random variables defined on a probability space where* Ω *consists in the elements of* $\mathbb{F}_q$*, the joint distribution is fully characterized by the set of distributions:*

$$\left\{ \sum_{i=1}^{n} \boldsymbol{\beta}_i X_i \mid \boldsymbol{\beta} \in \mathbb{F}_q^n \right\}.$$

Thereafter, in different parts of the paper, we use this equivalence to connect the results from the error-correcting code part to the results from the probing security part. This lemma is an extension of the XOR-Lemma which takes care of the case $q = 2$. This equivalence is sometimes implicitly assumed, however, since we did not find a reference or a simple proof, we give a non-trivial proof. As the result in itself, and the techniques used to prove it, are far away from the main contributions of this paper, we defer the additional definitions required and the detailed proof to Appendix A.

2.4 Circuits and security definitions

We consider a circuit C as a directed acyclic graph whose vertices are gates and edges are wires. We will assume that the gates are elementary operations over $\mathbb{F}_q$ and wires carry values in $\mathbb{F}_q$. A randomized circuit is a circuit augmented with random gates. A random gate is a gate that produces a random variable in $\mathbb{F}_q$ and sends it along its output wire. We call variables carried by the wires of a circuit C as intermediate variables of C. For a circuit C with input $\mathbf{a} \in \mathbb{F}_q^{\phi}$, we use $\mathsf{C}(\mathbf{a}) = \mathbf{b}$ to denote that $\mathsf{C}(\mathbf{a})$ returns the output $\mathbf{b} \in \mathbb{F}_q^{\phi'}$, and for a set $\mathcal{P}$ of intermediate variables (usually called probes), $\mathsf{C}_{\mathcal{P}}(\mathbf{a})$ returns the values of probes when $\mathbf{a}$ is fed as the input of C.

We call a vector of variables (say, $\mathbf{x}$) over $\mathbb{F}_q$ to be independent of the other vector of variables (say, $\mathbf{y}$) if $\Pr(\mathbf{x} = \alpha \mid \mathbf{y} = \beta) = \Pr(\mathbf{x} = \alpha)$ for any value α of $\mathbf{x}$ and any value β of $\mathbf{y}$, where the probability is taken over the random coins used to generate these vectors.

Definition 1 (Private circuit compiler [ISW03])**.** *A private circuit compiler for a circuit* C *with input in* $\mathbb{F}_q^{\phi}$ *and output in* $\mathbb{F}_q^{\phi'}$ *is defined by a triple* $(\mathsf{I}, \mathsf{T}, \mathsf{O})$ *where*

- $\mathsf{I} : \mathbb{F}_q^{\phi} \to \mathbb{F}_q^{\varphi}$ *is an input encoder that randomly maps the input in* $\mathbb{F}_q^{\phi}$ *to the input sharing in* $\mathbb{F}_q^{\varphi}$*.*
- T *is a circuit transformation whose input is circuit* C*, and output is a randomized circuit* C'*, whose input is the input sharing* $\mathbb{F}_q^{\varphi}$*, and the output in* $\mathbb{F}_q^{\varphi'}$ *is called output sharing.*
- $\mathsf{O} : \mathbb{F}_q^{\varphi'} \to \mathbb{F}_q^{\phi'}$ *is a decoder that maps the output sharing in* $\mathbb{F}_q^{\varphi'}$ *to the output of* C *in* $\mathbb{F}_q^{\phi'}$*.*

We say that $(\mathsf{I}, \mathsf{T}, \mathsf{O})$ *is a private circuit compiler and* C' *is a d-private circuit (or d-probing secure, where d is called the security order) if the following requirements hold:*

- *Correctness: for any input* $\mathbf{a} \in \mathbb{F}_q^{\phi}$*,* $\Pr \mathsf{O}(\mathsf{C}'(\mathsf{I}(\mathbf{a}))) = \mathsf{C}(\mathbf{a}) = 1$*.*
- *Privacy: for any input* $\mathbf{a} \in \mathbb{F}_q^{\phi}$ *and any set of probes* $\mathcal{P}$ *such that* $|\mathcal{P}| \leq d$*,* $\mathsf{C}'_{\mathcal{P}}(\mathsf{I}(\mathbf{a}))$ *are independent of the input* $\mathbf{a}$*.*

A usual approach to construct a private circuit compiler is to use subroutines that can be composed to compile any circuit. We define an encoder that enables to encode any vector of a fixed length (denoted k) into a vector of a bigger fixed length (denoted n). Then, the input encoder I consists of several instances of this (fixed-length) encoder. Analogically, the output decoder O consists of several instances of a fixed-length decoder. The circuit transformation is handled by showing that some simple circuits (typically gates) can be transformed into a corresponding gadget, and that any circuit can be expressed as the composition of these simple circuits. Then, the correctness and the privacy of the circuit transformation hold if it is compatible with each gadget, and with the combination.

More concretely, a private circuit compiler can be realized by combining a set of encoders, a set of basic gadgets and a set of decoders, which we defined below.

Definition 2 (Encoder, Codeword, Sharing, Valid Sharing and Share)**.** *An encoder* $\mathsf{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ *is a probabilistic algorithm that maps a vector in* $\mathbb{F}_q^k$ *to a vector in* $\mathbb{F}_q^n$*. The latter vector in* $\mathbb{F}_q^n$ *is called* codeword *or* valid sharing*. A* sharing *is a vector in* $\mathbb{F}_q^n$*, and the elements of a codeword or sharing are called* shares*. Moreover, an encoder is called d-private encoder if and only if the joint distribution of any d shares are independent of the input of the encoder, where the probability is over the random coins from the encoder.*

Note that the name 'sharing' is quite often used in the literature on masking, whereas the name 'codeword' comes from coding theory. To be consistent with both communities, we herein define codeword and valid sharing as equivalent. A codeword or valid sharing must be a sharing, but not vice versa.

Definition 3 (Decoder)**.** *A decoder for some encoder* $\mathsf{Enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$ *is a deterministic function* $\mathsf{Dec} : \mathbb{F}_q^n \to \mathbb{F}_q^k \cup \perp$ *such that for any* $x \in \mathbb{F}_q^k$*,* $\mathsf{Dec}(\mathsf{Enc}(x)) = x$ *with probability 1.*

In the rest of the paper, we usually use bold lower cases with hat superscript (*e.g.,* $\hat{\mathbf{x}}$) to denote sharings, $\hat{\mathbf{x}}[i]$ to denote the ith share of a sharing $\hat{\mathbf{x}}$, and i to denote the index of the share, for any $i \in [|\hat{\mathbf{x}}|]$.

We illustrate the definition with the example of Boolean masking and $k = 1$. To encode a variable $x \in \mathbb{F}_2$ into a sharing $\hat{\mathbf{x}} \in \mathbb{F}_2^n$, $n-1$ random variables $r_1, \ldots, r_{n-1}$ (uniformly distributed in $\mathbb{F}_2^{n-1}$) are generated for the encoding process: $\hat{\mathbf{x}}[i+1] = r_i$ for any $i \in [n-1]$, and the first share $\hat{\mathbf{x}}[1] = x \oplus \sum_{i=1}^{n-1} r_i$. Furthermore, we can see that any $n-1$ shares in $\hat{\mathbf{x}}$ are independent of x, and thus the encoder is an $(n-1)$-private encoder.

The circuit transformation T is realized from a set of basic gadgets $\{\mathsf{G}\}$ (gadget is defined below) and a set of gadget composition rules. The latter one maps circuits C and $\{\mathsf{G}\}$ to a directed acyclic graph C' whose vertices are gadgets in $\{\mathsf{G}\}$ and edges carry sharings. As a basic gadget is a randomized circuit, the output C' of a gadget composition function can also be regarded as a circuit whose gates are the gates inside basic gadgets and wires are either shares of sharings or wires inside basic gadgets.

Definition 4 (Gadget)**.** *A gadget is a randomized circuit whose input and output are sharings. We say that a gadget* G *ensures correctness for a function* $f : \mathbb{F}_q^l \to \mathbb{F}_q^{l'}$ *relatively to the encoders* Enc_i *for* $i \in [l]$ *and decoders* Dec_j *for* $j \in [l']$*, if* $\Pr\left[\mathsf{Dec}_i(\hat{\mathbf{z}}_i) = z_i)\right] = 1$*, for* $i \in [l']$ *where* $(\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_{l'}) = \mathsf{G}\big(\mathsf{Enc}_1(\mathbf{x}_1), \ldots, \mathsf{Enc}_l(\mathbf{x}_l)\big)$ *and* $(z_1, \ldots, z_{l'}) = f(x_1, \ldots, x_l)$*. Furthermore, all* $\hat{\mathbf{z}}_i$ *must be codewords.*

Note that a recursive composition of gadgets is also a gadget. To achieve a d-private circuit, each gadget should be a d-private circuit, which, however, is not sufficient: the composition of t-probing secure gadgets is not necessarily t-probing secure.[3] Barthe et al. introduced stronger notions of security that enable composability in [BBD+16], which are refined by several following works, *e.g.,* [BBP+16, BGR18]. In the rest of the paper, we separate the probes of a gadget into output probes and internal probes as follows:

- Output probes: output variables.
- Internal probes: intermediate variables except for the output probes.

First of all, we recall the simulatability framework introduced in [BBP+16]. Intuitively, a set of probes is simulatable from some of the input shares if there exists a simulator that can generate simulated values of the probes that have the same statistical distribution as their real values. We additionally define the notion of simulation under valid input sharings for a gadget, since the original definition was introduced in the case of Boolean masking, where any sharing is valid.

Definition 5 (Simulatability [BBP+16])**.** *Let* $\mathcal{P}$ *be a set of probes of a circuit (resp., gadget)* C *with input variables* $\mathcal{X}$*, and let* $\mathcal{X}' \subseteq \mathcal{X}$*. A simulator is a randomized function* S*:* $\mathbb{F}_q^{|\mathcal{X}'|} \to \mathbb{F}_q^{|\mathcal{P}|}$*. A distinguisher is a randomized function* D*:* $\left(\mathbb{F}_q^{|\mathcal{P}|}, \mathbb{F}_q^{|\mathcal{X}|}\right) \to \{0, 1\}$*. The set of probes* $\mathcal{P}$ *can be simulated (resp., simulated under valid input sharings) with input* $\mathcal{X}'$ *if and only if there exists a simulator* S *such that for any distinguisher* D *and any input* $x \in \mathbb{F}_q^{|\mathcal{X}|}$ *(resp., any valid input sharings), we have:*

$$\Pr\left[\mathsf{D}\left(\mathsf{C}_{\mathcal{P}}\left(x\right), x\right) = 1\right] = \Pr\left[\mathsf{D}\left(\mathsf{S}\left(x|_{\mathcal{X}'}\right), \mathcal{X}\right) = 1\right],$$

where the probability is over the random coins in C*,* S *and* D *and* $x|_{\mathcal{X}'}$ *denotes the elements of* x *corresponding to the inputs in* $\mathcal{X}'$*.*

t-Non-Inference (NI) and t-Strong Non-Inference (SNI) are two security notions for gadgets that are used together to support composition (see Lemma 4), where the former one is relatively weaker, but allows to build more efficient gadgets. For example, gadgets for linear operations can be easily and efficiently constructed following NI [ISW03, CS19], and gadgets for multiplication operations could save around half the amount of randomness when relaxing the SNI requirement into NI [BBP+16].

Definition 6 (t-Non-Inference (NI) [BBD+16])**.** *We say that a gadget with* l *input sharings and 1 output sharing is* t*-NI (resp.,* t*-NI for valid input sharings), if any* t_{int} *internal variables and* t_{out} *output shares such that* $t_{int} + t_{out} = t$ *can be simulated (resp., simulated under valid input sharings) with input shares indexed by* $\mathcal{I}_1, \ldots, \mathcal{I}_l$ *for* $|\mathcal{I}_1| \leq t, \ldots, |\mathcal{I}_l| \leq t$*.*

[3] In this paper, we use d and t to denote the security order for encoders and gadgets, respectively.

Definition 7 (t-Strong Non-Inference (SNI) [BBD+16])**.** *We say that a gadget with l input sharings and 1 output sharing is t-SNI (resp., t-SNI for valid input sharings), if any t_{int} internal variables and t_{out} output shares such that $t_{int} + t_{out} = t$ can be simulated (resp., simulated under valid input sharings) with input shares indexed by $\mathcal{I}_1, \ldots, \mathcal{I}_l$ for $|\mathcal{I}_1| \leq t_{int}, \ldots, |\mathcal{I}_l| \leq t_{int}$.*

Next, we cover some composition properties of NI and SNI gadgets. Those properties still hold when NI and SNI are replaced with their *valid input sharing* counterparts, assuming all gadgets are correct are compatible (that is, connected inputs and outputs have compatible associated encoders and decoders).

The work in [BBD+16] links the notions of NI and probing security, which is recalled and rephrased in Lemma 2.

Lemma 2 (NI and t-private encoders implies probing security)**.** *A t-NI for valid input sharings gadget is a t-private circuit if all of its input sharings come from d-private encoders for $t \leq d$, and are independently encoded.*

Proof. For a t-NI gadget G with l input sharings $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_l$, by definition of t-NI, any set of probes, say $\mathcal{P}$, such that $|\mathcal{P}| \leq t$, can be simulated with the input shares indexed by $\mathcal{I}_1, \ldots, \mathcal{I}_l$, for $|\mathcal{I}_1| \leq |\mathcal{P}|, \ldots, |\mathcal{I}_l| \leq |\mathcal{P}|$. By the definition of t-private encoders, $\hat{\mathbf{x}}_i[\mathcal{I}_i]$ is independent of the secrets for any $i \in [l]$, and thanks to the independent encoding of the input sharings, $\hat{\mathbf{x}}_1[\mathcal{I}_1], \ldots, \hat{\mathbf{x}}_l[\mathcal{I}_l]$ are independent of the secrets as well. At last, by the definition of the simulatability, the probes $\mathcal{P}$ can be regarded as a random function of $\hat{\mathbf{x}}_1[\mathcal{I}_1], \ldots, \hat{\mathbf{x}}_l[\mathcal{I}_l]$, and thus they are independent of the secrets, which meets the definition of t-private circuit. □

The work of [BBD+16] illustrates the composability of t-NI and t-SNI gadgets, which requires the t-SNI refresh gadget that is defined as follows:

Definition 8 (t-SNI Refresh gadget)**.** *A t-SNI refresh gadget is a t-SNI gadget with one input sharing and one output sharing that ensures correctness for the identity function.*

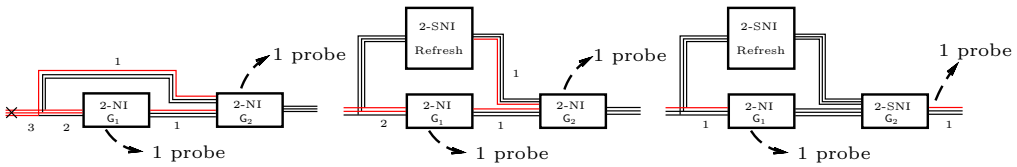
We recall the composability of t-NI and t-SNI gadgets in Lemma 3, which was introduced in [BBD+16], and then appears in many studies, *e.g.,* [BBP+16, BGR18].

Lemma 3 (Composability of t-NI and t-SNI gadgets)**.** *A composition of gadgets is t-NI if all gadgets are t-NI or t-SNI, based on the following composition rule: each sharing is used at most once as input of a gadget other than t-SNI refresh gadget. Moreover, a composition of gadgets is t-SNI if it is t-NI and the output sharings are from t-SNI gadgets.*

We give a simple example in Figure 1 to illustrate the gadgets composition introduced in Lemma 3. Say that we aim at composing two gadgets G_1 and G_2, where the former one has one input and output sharing, the latter one has two inputs and one output sharing, and there is one probe for each gadget.

- Figure 1-(a) shows an improper composition (that violates the composition rule in Lemma 3) where the input sharing is directly linked to the input of both G_1 and G_2. To simulate the probe in G_2, two input shares from G_2 are needed, and to simulate the two shares and the probe in G_1, 3 shares from the input of G_1 are needed, which therefore violates the definition of 2-NI.

- Figure 1-(b) follows the composition rule in Lemma 3 by adding a 2-SNI refresh gadget before one of input sharing of G_2, and thus only two shares from the input of G_1 are needed to simulate the two probes, which fits the definition of 2-NI.



(a) improper composition (b) 2-NI case (c) 2-SNI case

Figure 1: A simple example to illustrate the composability in Lemma 4 with 3 shares. Shares required for simulation are drawn in red.

- To give a composition example that results in SNI, Figure 1-(c) further replaces G_2 with a 2-SNI gadget. The probe in G_2 (1 output probe) can be simulated without any input share, and thus both G_1 and the refresh gadget have no probe that is propagated from G_2. Also, as G_1 is 2-NI, only one input share of G_1 is needed to simulate the probe (1 internal probe) in G_1, giving that only one input share is needed to simulated the two probes (1 from internal variables and another 1 from output shares) in the composed gadget, which fits the definition of 2-SNI.

We further present a specific composability of only t-SNI gadgets in Lemma 4, which is useful later in Section 6.3 for the application to AES S-boxes.

Lemma 4 (Composability of t-SNI gadgets)**.** *Let $\mathsf{G}_1, \ldots, \mathsf{G}_\ell$ be t-SNI gadgets. A composition of them is t-SNI if the input sharings of any gadget come from different (other) gadgets, and there is only one output sharing.*

Proof. Let G be a composite gadget. A composition of ℓ gadgets can be seen as a directed acyclic graph whose vertices are gadgets and edges transfer sharings. As every directed acyclic graph has a topological ordering, we can order the gadgets as $\mathsf{G}_1, \ldots, \mathsf{G}_\ell$, such that G_i comes before G_j for $i < j$. Let the number of internal probes of G that belong to G_j be t_j for $j \in [\ell]$, and let the number of output probes of G be t_O (which is also the output probes of G_ℓ, since there is only one output sharing).

By recurrence, for $j > 1$, assuming that the sub-part of G consisting of $\mathsf{G}_j, \ldots, \mathsf{G}_\ell$ (denoted as $\mathsf{G}^{(j)}$) is t-SNI, we prove that the sub-part consisting of $\mathsf{G}_{j-1}, \ldots, \mathsf{G}_\ell$ (denoted as $\mathsf{G}^{(j-1)}$) is also t-SNI. The base case $\mathsf{G}^{(\ell)} = \mathsf{G}_\ell$ is t-SNI by assumption.

Let the input sharings of $\mathsf{G}^{(j)}$ be $\mathcal{S}^{(j)}$. Let $(\mathcal{S}_1^{(j)}, \mathcal{S}_2^{(j)})$ be a partition of $\mathcal{S}^{(j)}$ such that the sharings in $\mathcal{S}_1^{(j)}$ are outputs of G_{j-1} ($|\mathcal{S}_1^{(j)}| \leq 1$ since G_{j-1} has one output), which implies that $\mathcal{S}_2^{(j)} \subset \mathcal{S}^{(j-1)}$. We next show that a set of t_1 internal probes in G_{j-1}, t_2 internal probes in $\mathsf{G}^{(j)}$ and t_O output probes (such that $t_1 + t_2 + t_O \leq t$) can be simulated using at most $t_1 + t_2$ shares of each input sharing in $\mathcal{S}^{(j-1)}$. The t_2 and t_O probes can be simulated using at most t_2 shares of each sharing in $\mathcal{S}^{(j)}$ (using the $\mathsf{G}^{(j)}$ t-SNI simulator). The t_1 probes, along with at most t_2 shares of each sharing in $\mathcal{S}_1^{(j)}$ can be simulated with at most t_1 shares of each sharing in $\mathcal{S}^{(j-1)}$ (using the G_{j-1} simulator), and since $\mathcal{S}_2^{(j)} \subset \mathcal{S}^{(j-1)}$, the t_2 shares of each sharing in $\mathcal{S}_2^{(j)}$ can be simulated with t_2 shares of each sharing in $\mathcal{S}^{(j-1)}$. □

2.5 Code-based encoder / decoder and linear codes

In this part, we give some basic notions of linear codes. We fix notations relatively to code-based encoders and decoders, and we relate code-based encoders to other encoders of previous works.

Definition 9 (Code-based encoder)**.** *Let k, m and n be three positives integers such that $n \geq k + m$, we define the encoder $\mathsf{Enc}_{\mathbf{A}}$ relatively to the matrix $\mathbf{A} \in \mathbb{F}_q^{(m+k)\times n}$ as*

the following algorithm. On input $\mathbf{x} \in \mathbb{F}_q^k$*, a vector* $\mathbf{r}$ *is chosen following the uniform distribution over* $\mathbb{F}_q^m$*. The product* $[\mathbf{x}, \mathbf{r}]\mathbf{A}$ *is computed, and the resulting sharing is given as output and denoted* $\hat{\mathbf{x}}$*. We refer to:*

- $\mathbf{x}$ *as the encoded vector,*
- $\hat{\mathbf{x}}$ *as the sharing, and as shares for its coefficients,*
- k *as the number of encoded variables,*
- m *as the number of random variables,*
- n *as the length of the sharing.*

Definition 10 (Code-based decoder)**.** *Let* k, m *and* n *be three positives integers such that* $n \geq k + m$*, we define the decoder* $\mathsf{Dec}_{\mathbf{A}}$ *relatively to the matrix* $\mathbf{A} \in \mathbb{F}_q^{(k+m)\times n}$ *as the following deterministic function. On input sharing* $\hat{\mathbf{x}} \in \mathbb{F}_q^n$*,* $\mathsf{Dec}_{\mathbf{A}}$ *outputs the vector* $\mathbf{x}$ *such that* $\exists \mathbf{r} \in \mathbb{F}_q^m$ *such that* $[\mathbf{x}, \mathbf{r}]\mathbf{A} = \hat{\mathbf{x}}$ *(and outputs* $\perp$ *if* $\mathbf{r}$ *does not exist). We denote* $\mathbf{x} = \mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{x}})$*.*

Notation. By extension, we denote $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$, when $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{x}}) = \mathbf{x}$, and $\hat{\mathbf{x}}$ is uniformly distributed over $\{[\mathbf{x}, \mathbf{r}]\mathbf{A} \mid \mathbf{r} \in \mathbb{F}_q^m\}$.

As code based encoders / decoders rely on linear codes and their properties, we recall basic definitions and associated vocabulary.

Definition 11 (Linear code)**.** *A linear code of length* n *and rank* k *is a linear subspace* $\mathcal{C}$ *with dimension* k *of the vector space* $\mathbb{F}_q^n$ *where* $\mathbb{F}_q$ *is the finite field with* q *elements.*

A linear code is often associated to its parameters: $[n, k, d]$ *where:*

- n *is the length of the sharing,* k *the dimension of the code,*
- d *is the minimal distance of the code:* $d = \min_{w \in \mathcal{C} \setminus \{\mathbf{0}\}} \mathrm{HW}(w)$*.*

Definition 12 (Generator and parity-check matrices)**.** *The entire code* $\mathcal{C}$ *can be represented as the span of a* $k \times n$ *matrix* $\mathbf{G}$*, called generator matrix of* $\mathcal{C}$*. A code with a generating matrix* $\mathbf{G}$ *can be presented as* $\mathcal{C}_{\mathbf{G}}$*.*

The set of words $w \in \mathbb{F}_q^n$ *such that* $\mathbf{G}w = 0$ *(or equivalently* $c \cdot w = 0$ *for all* $c \in \mathcal{C}$*) forms a linear code called dual code of* $\mathcal{C}$ *and noted* $\mathcal{C}^\perp$*. A matrix* $\mathbf{H}$ *generating* $\mathcal{C}^\perp$ *is called a parity-check matrix for* $\mathcal{C}$*.*

Property 1 (Dual code parameters)**.** *Let* $\mathcal{C}$ *be an* $[n, k, d]$ *linear code, then* $\mathcal{C}^\perp$ *is a* $[n, n-k, d']$ *code, where* d' *is called the dual distance of* $\mathcal{C}$*.*

In the following, we illustrate the connections between some encoders from other works and code-based encoders. As described in the precedent sub-section, we can describe Boolean masking as a code-based encoder. More precisely, the matrix $\mathbf{A}$ for one secret binary variable and m random binary variables is depicted in Figure 2. This square matrix has size $1 + m$, the encoded variable is multiplied by the top part and the random part by the lower part. The matrix corresponding to the additive masking encoder over $\mathbb{F}_q$ is very similar (a negative sign appears for each element of the first column of the lower matrix if q is not a power of 2). [4]

For the inner product masking, the matrix $\mathbf{A}$ is obtained by modifying the first column of the lower part, which is given by the public vector used. We give an example in Figure 2 for the encoding of a secret variable in a characteristic 2 field, relatively to the public

[4] Additive masking is a generalization of Boolean masking to a finite field $\mathbb{F}_q$.

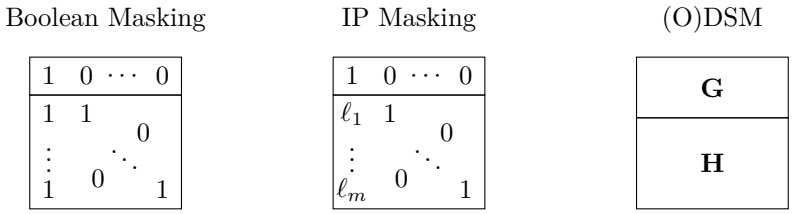


Figure 2: Matrix of $\mathsf{Enc}_{\mathbf{A}}$ for usual encoders. In each case the upper part has k rows, k being the number of encoded variables, and the lower part has m rows, the number of random variables.

vector $(\ell_1, \cdots, \ell_m)$. For the code-based encoders DSM [PGS+17], and ODSM [BCC+14] the square matrix $\mathbf{A}$ can be separated in two matrices $\mathbf{G}$ (upper part) and $\mathbf{H}$ (lower part) with few properties. The direct sum property means that any element of length n has a unique decomposition as a combination of rows of $\mathbf{G}$ plus a combination of rows of $\mathbf{H}$. For ODSM the extra condition of having $\mathbf{G}$ and $\mathbf{H}$ orthogonal is added. The general structure of these two encoders is depicted in Figure 2.

3 Code-based encoders for generalized masking

In this part, we introduce a new family of code-based encoders called generic encoders, which are more general than the previous constructions and we study their advantages. This generalization allows to overstep the upper bound on the probing security proven in [PGS+17] on DSM: we show that the probing security can be higher than the dual distance of the code generated by $\mathbf{H}$. For generic encoders, the structure of two matrices (one for the secret variables, the other for the random variables) is kept but the direct sum property is not mandatory anymore. It allows us to consider the resistance to fault injection attacks from the properties of the error correcting code used. The generalization encompasses various already used encoders such as Boolean masking (additive masking), inner product masking, ODSM and DSM.

We first define the new encoders, we explain the choices for this generalization of DSM and why further code generalization seems unlikely to benefit to encoders. Then, we focus on the d-privacy. More precisely, we exhibit a parameter of the code determining the d-privacy which is more precise than the dual distance d' of the code generated by $\mathbf{H}$, and is always equal or greater than d'. It oversteps the upper bound proven in [PGS+17] and we show why d and d' coincide in the special case of DSM. Finally, we study particularly this new parameter, giving tight lower and upper bounds.

Definition 13 (Generic encoder)**.** *Let k, m and n be three positive integers, a generic encoder is a code-based encoder with the following restriction on $\mathbf{A} \in \mathbb{F}_q^{(k+m)\times n}$:*

- $\mathbf{G}$ *is the $k \times n$ upper part of $\mathbf{A}$, which is the part multiplied by the encoded variables. $\mathbf{Rank}(\mathbf{G}) = k$ and we refer to $\mathcal{C}_{\mathbf{G}}$ for the code generated by $\mathbf{G}$.*
- $\mathbf{H}$ *is the $m \times n$ lower part of $\mathbf{A}$, which corresponds to the part multiplied by the randomness. $\mathbf{Rank}(\mathbf{H}) = m$ and we refer to $\mathcal{C}_{\mathbf{H}}$ for the code generated by $\mathbf{H}$.*
- $\mathcal{C}_{\mathbf{G}} \cap \mathcal{C}_{\mathbf{H}} = \{\mathbf{0}\}$.

Remark 1 (Generic encoder rationale). The restrictions on the matrices to encode are the result of the masking prerequisites. First, $\mathbf{Rank}(\mathbf{G}) = k$ enables to recover the k elements

of $\mathbf{x}$ from $\hat{\mathbf{x}}$. Then, $\mathbf{Rank}(\mathbf{H}) = m$, otherwise the same code $\mathcal{C}_\mathbf{A}$ could be obtained with less than m random variables, sparing some randomness. Finally, $\mathcal{C}_\mathbf{G} \cap \mathcal{C}_\mathbf{H} = \{\mathbf{0}\}$ is needed to be able to recover exactly $\mathbf{x}$ from $\hat{\mathbf{x}}$, it also corresponds to $\mathbf{Rank}(\mathbf{A}) = k + m$.

This encoding technique generalizes the approach of ODSM and DSM, by allowing n to be greater than $k + m$. When $n > k + m$, $\mathcal{C}_\mathbf{G}$ and $\mathcal{C}_\mathbf{H}$ are not supplementary anymore, invalidating the direct sum property, the dimension of $\mathcal{C}_\mathbf{A}$ is not equal to n anymore, but the decomposition of a codeword as a combination of rows of $\mathbf{G}$ plus a combination of rows of $\mathbf{H}$ is still unique. The merit of this technique is to allow some redundancy which can be used to thwart fault injection attacks. In this respect, we show in Appendix C that our scheme can provide the fault resistance with higher order (than DSM) when some error detection codes are adopted. The only remaining constraints on $\mathbf{G}$ and $\mathbf{H}$ directly come from the properties wanted for an encoder: the ability of recovering the whole encoded vector and the rule of minimizing the number of random variables which is why we claim that further code generalization does not seems relevant.

We next describe a lemma that relates to the decoding of codewords of a generic encoder $\mathsf{Enc}_\mathbf{A}$, and the generalized inverse of $\mathbf{A}$.

Lemma 5. *For a generic encoder* $\mathsf{Enc}_\mathbf{A}$ *with matrix* $\mathbf{A}$*, for any* $\mathbf{x} \in \mathbb{F}_q^k$ *and* $\hat{\mathbf{x}} \xleftarrow{\$} \mathsf{Enc}_\mathbf{A}$*,* $\mathbf{x} = \hat{\mathbf{x}}\mathbf{A}^{-1}[*, 1\!:\!k]$ *where where* $\mathbf{A}^{-1}$ *is the generalized inverse of* $\mathbf{A}$*.*

Proof. Let $\mathbf{r} \in \mathbb{F}_q^m$ be such that $\hat{\mathbf{x}} = [\mathbf{r}, \mathbf{x}]\mathbf{A}$, let also $\mathbf{x}' = \hat{\mathbf{x}}\mathbf{A}^{-1}[*, 1 : k]$ and $\mathbf{r}' = \hat{\mathbf{x}}\mathbf{A}^{-1}[*, k + 1\!:\!k + m]$. Using the property of the generalized inverse, we get

$$[\mathbf{x}, \mathbf{r}]\mathbf{A} = [\mathbf{x}, \mathbf{r}]\mathbf{A}\mathbf{A}^{-1}\mathbf{A} = \hat{\mathbf{x}}\mathbf{A}^{-1}\mathbf{A} = [\mathbf{x}', \mathbf{r}']\mathbf{A}.$$

Therefore, $[\mathbf{x}, \mathbf{r}] \times [\mathbf{G}; \mathbf{H}] = [\mathbf{x}', \mathbf{r}'] \times [\mathbf{G}; \mathbf{H}]$, and thus $\mathbf{x}\mathbf{G} = \mathbf{x}'\mathbf{G}$. Since the rank of $\mathbf{G}$ is k, its rows are linearly independent, giving $\mathbf{x}' = \mathbf{x}$. □

With this type of general encoders, and an additional definition, we can quantify the d-privacy of the encoder based on the properties of $\mathbf{G}$ and $\mathbf{H}$. In this case it will not be given by the dual distance of $\mathcal{C}_\mathbf{H}$ as for DSM but by a new parameter.

Definition 14 (Set of fixed weight codewords)**.** *Let* $\mathcal{C}$ *be a linear code of length* n*, we denote* $\{\mathcal{C}\}_{\leq t}$ *its set of codewords of Hamming weight less than or equal to* t*:*

$$\{\mathcal{C}\}_t = \{\mathbf{w} \in \mathcal{C} \,|\, \mathrm{HW}(\mathbf{w}) \leq t\}.$$

Proposition 4 (Generic encoders and d-privacy)**.** *Let* $\mathsf{Enc}_\mathbf{A}$ *be a generic encoder, let* d' *denote* $d(\mathcal{C}_{\mathbf{H}^\perp})$ *then* $\mathsf{Enc}_\mathbf{A}$ *is* d*-private with:*

$$d = \max\left\{t \in \{0, \ldots, n\} : \forall \mathbf{w} \in \{\mathcal{C}_{\mathbf{H}^\perp}\}_t,\, \mathbf{G}\mathbf{w}^\mathrm{T} = \mathbf{0}^\mathrm{T}\right\}, \tag{1}$$

$$= d' - 1 + \max\left\{t \in \{0, \ldots, n - d' + 1\} : \forall \mathbf{w} \in \{\mathcal{C}_{\mathbf{H}^\perp}\}_{d'-1+t},\, \mathbf{G}\mathbf{w}^\mathrm{T} = \mathbf{0}^\mathrm{T}\right\}. \tag{2}$$

Proof. Showing that the encoder is d-private consists in proving that any joint distribution given by at most d shares of a codeword is independent from the encoded vector. Using Lemma 1, such a joint distribution is fully characterized by the set of all linear combinations of these d shares. If all these linear combinations are independent from the encoded variable, so is the joint distribution.

Each codeword $\hat{\mathbf{x}} \xleftarrow{\$} \mathsf{Enc}_\mathbf{A}$ can be written as $\hat{\mathbf{x}} = \mathbf{x}\mathbf{G} + \mathbf{r}\mathbf{H}$, therefore a linear combination of elements of $\hat{\mathbf{x}}$ can be written as $\sum_{i=1}^{k} a_i\mathbf{x}[i] + \sum_{i=1}^{m} b_i\mathbf{r}[i]$, which is independent of $\mathbf{x}$ if all a_i are null or if any b_i is non-null (in which case it is uniformly distributed, independently of $\mathbf{x}$). Furthermore, probing ℓ positions of the codeword corresponds to multiplying $\hat{\mathbf{x}}$ by a vector $\mathbf{w} \in \mathbb{F}_q^n$ such that $\mathrm{HW}(\mathbf{w}) = \ell$ (more precisely by all vectors covered by $\mathbf{w}$). The encoder is then d-private if there exists no $\mathbf{w}$ such that $\mathrm{HW}(\mathbf{w}) \leq d$ and $\hat{\mathbf{x}}\mathbf{w}^\mathrm{T} = \sum_{i=1}^{k} a_i\mathbf{x}[i]$ where the $a_i \in \mathbb{F}_q$ are not all null.

Let's call $d = \max_{0 \leq t \leq n} \{\forall \mathbf{w} \in \{\mathcal{C}_{\mathbf{H}^\perp}\}_t, \mathbf{G}\mathbf{w}^\mathrm{T} = \mathbf{0}\}$, we first show that it is an upper bound of the d-privacy, and then that it is a lower bound. First, by definition of d, there exists $\mathbf{w}' \in \{\mathcal{C}_{\mathbf{H}^\perp}\}_{d+1}$, of Hamming weight $d+1$ such that $\mathbf{G}\mathbf{w}'^\mathrm{T} \neq \mathbf{0}^\mathrm{T}$. It means that for all codewords $\hat{\mathbf{x}}$ we have $\hat{\mathbf{x}}\mathbf{w}'^\mathrm{T} = \mathbf{x}\mathbf{G}\mathbf{w}'^\mathrm{T} + \mathbf{r}\mathbf{H}\mathbf{w}'^\mathrm{T} = \mathbf{x}\mathbf{G}\mathbf{w}'^\mathrm{T}$ since $\mathbf{w}' \in \mathcal{C}_{\mathbf{H}^\perp}$. Since $\mathbf{G}\mathbf{w}'^\mathrm{T} \neq \mathbf{0}^\mathrm{T}$, $\hat{\mathbf{x}}\mathbf{w}'^\mathrm{T}$ is a non trivial linear combination of $\mathbf{x}$ coefficients, consequently the encoding is not $(d+1)$-private.

Then, let us consider $e \leq d$. Any e-probing set corresponds to a vector $\mathbf{w}$ such that $\mathbf{w} \in \mathbb{F}_q^n, \mathrm{HW}(\mathbf{w}) = e$, and multiplying an encoding by $\mathbf{w}$ two cases arise. First, $\mathbf{w} \notin \mathcal{C}_{\mathbf{H}^\perp}$, in this case $\hat{\mathbf{x}}\mathbf{w}^\mathrm{T} = \mathbf{x}\mathbf{G}\mathbf{w}^\mathrm{T} + \mathbf{r}\mathbf{H}\mathbf{w}^\mathrm{T}$ where $\mathbf{H}\mathbf{w}^\mathrm{T} \neq \mathbf{0}^\mathrm{T}$ then $\mathbf{r}\mathbf{H}\mathbf{w}^\mathrm{T}$ removes the dependence on $\mathbf{x}$'s coefficients. In the other case, $\mathbf{w} \in \mathcal{C}_{\mathbf{H}^\perp}$, and since $\mathrm{HW}(\mathbf{w}) \leq e$ by definition of d we have that $\mathbf{G}\mathbf{w}^\mathrm{T} = \mathbf{0}^\mathrm{T}$, so $\hat{\mathbf{x}}\mathbf{w}^\mathrm{T} = 0$ which does not depend on the coefficients of $\mathbf{x}$. Finally the encoder is e-private for all $e \leq d$, and combining both bounds, the privacy is exactly d.

We proved Equation 1, Equation 2 is obtained by using that $\{\mathcal{C}_{\mathbf{H}^\perp}\}_{d'-1} = \{\mathbf{0}\}$ by definition of the distance d', and $\mathbf{G}\mathbf{0}^\mathrm{T} = \mathbf{0}^\mathrm{T}$. □

The d-privacy of the generic encoder comes with the following particular property (that is proven in the proof of Proposition 4).

Corollary 1. *Let* $\mathsf{Enc}_{\mathbf{A}}$ *be a d-private generic encoder, any linear combination of d shares of a codeword is either uniformly distributed or equal to* 0*, where the probability is over the random coins from the encoder.*

Remark 2. *Particular case of DSM.* For DSM encoders, the work in [PGS+17, Proposition 1] gives that the probing security is exactly $d(\mathcal{C}_{\mathbf{H}^\perp}) - 1$. Since $\mathcal{C}_{\mathbf{G}}$ and $\mathcal{C}_{\mathbf{H}}$ are supplementary for the code they consider, only $\mathbf{w} = \mathbf{0} \in \mathbb{F}_q^n$ is such that $\mathbf{H}\mathbf{w}^\mathrm{T} = \mathbf{0}^\mathrm{T}$ and $\mathbf{G}\mathbf{w}^\mathrm{T} = \mathbf{0}^\mathrm{T}$, justifying this result. As our generic encoder does not have this constraint, the dual distance of $\mathcal{C}_{\mathbf{H}}$ only gives a lower bound. The supplementary property also implies $d(\mathcal{C}_A) = 1$, therefore DSM encodings are 0 fault-resistant. (See Appendix C for more analysis on the fault resistance.)

The next proposition gives lower and upper bounds on the d-privacy. Both bounds can be tight, or even collapse, an example of such a situation is explained in Appendix B.

Proposition 5 (d-privacy bounds)**.** *Let* $\mathsf{Enc}_{\mathbf{A}}$ *be a generic encoder, let* d' *denote* $d(\mathcal{C}_{\mathbf{H}^\perp})$*, the d-privacy is such that:*

$$d' - 1 \leq d \leq m.$$

Proof. From Proposition 4, we know that $d' - 1 \leq d$. For the other bound, let us proceed by contradiction. If $d > m$, then $\forall \mathbf{w} \in \{\mathcal{C}_{\mathbf{H}^\perp}\}_{m+1}, \mathbf{G}\mathbf{w}^\mathrm{T} = \mathbf{0}^\mathrm{T}$. First, recall that $\mathcal{C}_{\mathbf{H}^\perp}$ has length n and dimension $n - m$, therefore the code is equivalent to one generated by a matrix $\mathbf{H}'^\perp$ made of the horizontal concatenation of the identity matrix of size $n - m$ and a matrix of size $(n - m) \times m$ (it can be obtained by performing the Gaussian elimination on $\mathbf{H}^\perp$).[5] All the rows of $\mathbf{H}'^\perp$ have Hamming weight at most $m + 1$, this means that the code $\mathcal{C}_{\mathbf{H}'^\perp} = \mathcal{C}_{\mathbf{H}^\perp}$ is generated by the set $\{\mathcal{C}_{\mathbf{H}^\perp}\}_{m+1}$ of its codewords of Hamming weight at most $m+1$. Therefore, the assumption implies that $\forall \mathbf{w} \in \mathcal{C}_{\mathbf{H}^\perp}, \mathbf{G}\mathbf{w}^\mathrm{T} = 0$, hence $\mathcal{C}_{\mathbf{H}^\perp} \subseteq \mathcal{C}_{\mathbf{G}^\perp}$, so $\mathcal{C}_{\mathbf{G}} \subseteq \mathcal{C}_{\mathbf{H}}$, which leads to a contradiction as per Definition 13, $k > 0$ and $\mathcal{C}_{\mathbf{G}} \cap \mathcal{C}_{\mathbf{H}} = \{\mathbf{0}\}$. □

For the ease of understanding, we provide an example of generic encoder in Example I.

Example I

We give a concrete non-trivial generic encoder over $\mathbb{F}_2$ as an example, which also will be further used to exemplify the new gadgets in the next sections. The matrices

[5] $\mathbf{H}'^\perp$ is also known as the systematic form of the generating matrix.

$\mathbf{G}$, $\mathbf{H}$ and $\mathbf{A}$ are:

$$\mathbf{G} = \begin{bmatrix} 11110000 \\ 00001100 \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} 11000000 \\ 00110001 \\ 00011100 \\ 00000111 \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} 11110000 \\ 00001100 \\ 11000000 \\ 00110001 \\ 00011100 \\ 00000111 \end{bmatrix}.$$

We can verify that $\mathcal{C}_{\mathbf{G}} \cap \mathcal{C}_{\mathbf{H}} = \{\mathbf{0}\}$, $\mathbf{Rank}(\mathbf{G}) = 2$ and $\mathbf{Rank}(\mathbf{H}) = 4$, and thus the corresponding encoder is a generic encoder. We consider the secret input $\mathbf{x} = [1, 1]$ and assume that the randomness drawn is $\mathbf{r} = [1, 1, 1, 1]$. The codeword is then:

$$\hat{\mathbf{x}} = [1, 1, 1, 1, 1, 1] \times \mathbf{A} = [0, 0, 0, 1, 0, 1, 1, 0].$$

By Equation 1 of Proposition 4, we can calculate the security order $d = 2$, whereas the dual distance of $\mathcal{C}_{\mathbf{H}}$ is $d' = 2$. This particular generic encoder has the property that $d > d' - 1$, and thus confirms that the dual distance of $\mathcal{C}_{\mathbf{H}}$ only gives a lower bound.

4 Multiplication gadget

In this section, we consider the element-wise multiplication (sometimes called the entrywise multiplication or the Hadamard multiplication) between two vectors of secret variables. That is, for $\mathbf{x} = \big(\mathbf{x}[1], \ldots, \mathbf{x}[k]\big)$ and $\mathbf{y} = \big(\mathbf{y}[1], \ldots, \mathbf{y}[k]\big)$, we consider the computation of $\mathbf{z} = \mathbf{x} \odot \mathbf{y} \stackrel{\text{def}}{=} \big(\mathbf{x}[1]\mathbf{y}[1], \ldots, \mathbf{x}[k]\mathbf{y}[k]\big)$ in the masked domain. The input of the multiplication gadget is the codewords $\hat{\mathbf{x}} \stackrel{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ and $\hat{\mathbf{y}} \stackrel{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y})$ of $\mathbf{x}$ and $\mathbf{y}$ respectively, and the output should be a codeword of $\mathbf{x} \odot \mathbf{y}$, *i.e.,* $\mathsf{G}_{mul}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \stackrel{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x} \odot \mathbf{y})$. We start this section with the construction, then give the proofs of the correctness and security.

4.1 Construction in a nutshell

As a beginning, for the input codewords $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ and output codeword $\hat{\mathbf{z}}$, we give an illustration of our multiplication gadget in Figure 3. The multiplication gadget can be divided into three steps as follows:

A This part first performs an ISW-like product-then-refresh procedure [ISW03], and then transforms the result $\mathbf{S}$ to additive sharings by multiplying each of its rows with a pre-computed matrix $\mathbf{M}_i$. That is, the columns of $\mathbf{T}[*, 1\!:\!k]$ are additive sharings of the secret result $\mathbf{z}$: $\sum_{i=1}^{n} \mathbf{T}[i, 1\!:\!k] = \mathbf{z} = \mathbf{x} \odot \mathbf{y}$.

B This part (linearly) transforms the additive sharings into codewords, by multiplying the matrix $\mathbf{T}$ with $\mathbf{A}$.

C This part performs a refresh operation followed by a vertical ISW-like compress procedure to get the final result.

Note that we make the intermediate matrix $\mathbf{T}$ explicit in our descriptions for convenience. On the one hand, is simplifies the understanding of the construction as well as the proofs of correctness and security in the next subsection. More importantly, it is also useful to make our multiplication gadget's results directly applicable to linear gadgets (as will be discussed in Section 5). In particular, the intuition that $\mathbf{T}[*, 1\!:\!k]$ are additive sharings

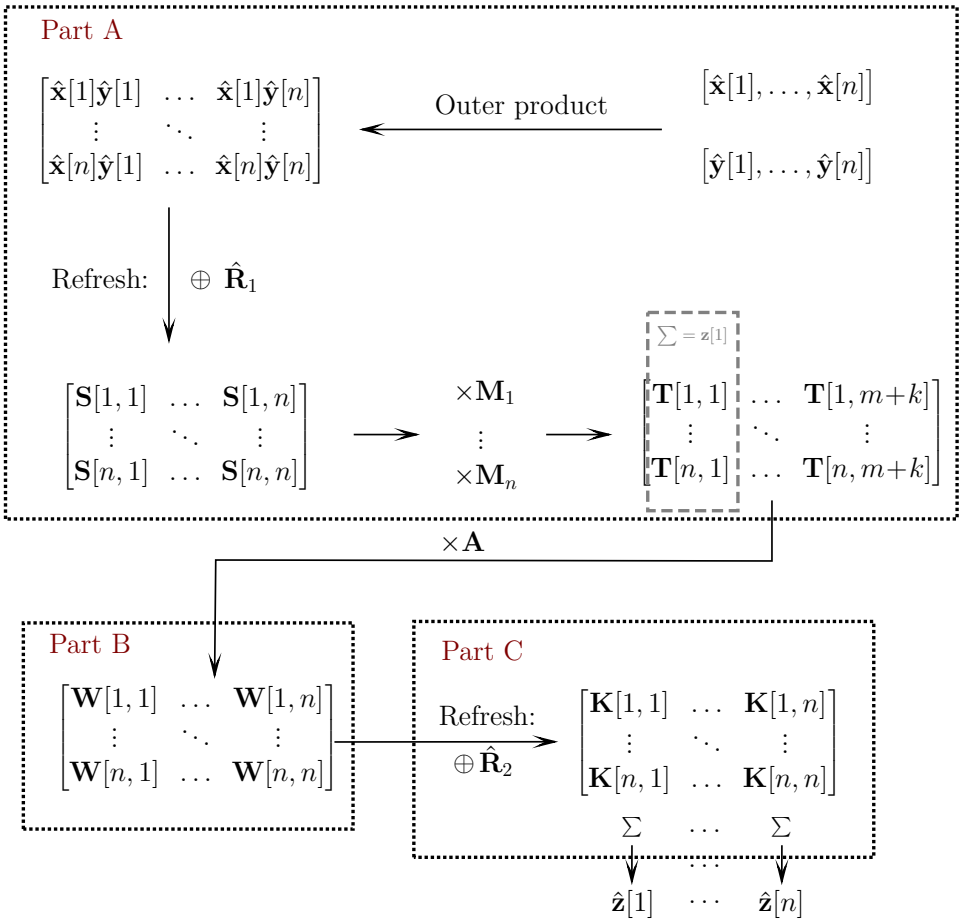


Figure 3: Illustration of the multiplication gadget.

of the secret result will be handy in this case. From an implementation viewpoint, the linear parts of this multiplication can be either precomputed (jointly) or take advantage of special structures that the matrices $\mathbf{M}$ and $\mathbf{A}$ may have.

We note that, because of the additive sharing of the columns of $\mathbf{T}$, we lose the order amplification as formalised by [WSY+16, PGS+17]. But since at this stage of the computation, we may still have more additive shares than d, there are cases where code-based masking could maintain security order amplification even for the multiplication (which we leave as a scope for further investigation).

4.2 Construction in details

We build the masked multiplication in Gadget 1 as described in Figure 3 of the last section. More details on the three-parts:

A The outer product $\hat{\mathbf{x}}^{\mathrm{T}} \otimes \hat{\mathbf{y}}$ is computed, and is refreshed by adding a matrix $\hat{\mathbf{R}}_1$ of which every column is a codeword of $\mathbf{0}$, resulting in a matrix denoted as $\mathbf{S}$. Then, for any $i \in [n]$, the i^{th} row of matrix $\mathbf{S}$ is multiplied by a pre-computed matrix $\mathbf{M}_i$ that is related to the generic encoder of the input and will be explained in the next sub-section, and the results compose a matrix $\mathbf{T}$.

B The product $\mathbf{W} = \mathbf{TA}$ is computed.

C The matrix $\mathbf{W}$ is refreshed by adding a matrix $\hat{\mathbf{R}}_2$ of which every row is a codeword of $\mathbf{0}$, resulting in a matrix denoted as $\mathbf{K}$. Then, the final result is computed by summing all the rows of $\mathbf{K}$.

Afterwards, we illustrate the construction in Example II.

Gadget 1 Multiplication gadget: CodeMul

Input: Codewords $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ and $\hat{\mathbf{y}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y})$ of $\mathbf{x} \in \mathbb{F}_q^k$ and $\mathbf{y} \in \mathbb{F}_q^k$ respectively
Output: $\hat{\mathbf{z}} \in \mathbb{F}_q^n$

The gadget ensures that $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathbf{x} \odot \mathbf{y}$
Encoder $\mathsf{Enc}_{\mathbf{A}}$ should be a d-private generic encoder
For $i \in [n]$, matrix $\mathbf{M}_i \in \mathbb{F}_q^{n\times(k+m)}$ is defined as $\mathbf{M}_i \overset{\text{def}}{=} (\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}})\mathbf{E}$, where $\tilde{\mathbf{A}} \overset{\text{def}}{=} \left[\mathbf{A}^{-1}[1\!:\!k,*], \mathbf{O}^{n\times m}\right]$ and $\mathbf{O}^{n\times m}$ is an $n \times m$ zero matrix

Gadget 1 part A

Input: $\hat{\mathbf{x}} \in \mathbb{F}_q^n$, $\hat{\mathbf{y}} \in \mathbb{F}_q^n$
Output: $\mathbf{T} \in \mathbb{F}_q^{n\times(k+m)}$

1: Generate a uniformly distributed matrix $\mathbf{R}_1 \in \mathbb{F}_q^{n\times m}$, and compute $\hat{\mathbf{R}}_1 = \left([\mathbf{O}^{n\times k}, \mathbf{R}_1]\mathbf{A}\right)^{\mathrm{T}}$
2: **for** $i = 1$; $i \le n$; i++ **do**
3: **for** $j = 1$; $j \le n$; j++ **do**
4: $\mathbf{S}[i,j] = \hat{\mathbf{x}}[i]\hat{\mathbf{y}}[j] \oplus \hat{\mathbf{R}}_1[i,j]$
5: **end for**
6: **end for**
7: **for** $i = 1$; $i \le n$; i++ **do**
8: $\mathbf{T}[i,*] = \mathbf{S}[i,*] \times \mathbf{M}_i$
9: **end for**

Gadget 1 part B

Input: $\mathbf{T} \in \mathbb{F}_q^{n\times(k+m)}$
Output: $\mathbf{W} \in \mathbb{F}_q^{n\times n}$

1: $\mathbf{W} = \mathbf{TA}$

Gadget 1 part C

Input: $\mathbf{W} \in \mathbb{F}_q^{n\times n}$
Output: $\hat{\mathbf{z}} \in \mathbb{F}_q^n$

1: Generate a uniformly distributed matrix $\mathbf{R}_2 \in \mathbb{F}_q^{n\times m}$, and compute $\hat{\mathbf{R}}_2 = [\mathbf{O}^{n\times k}, \mathbf{R}_2]\mathbf{A}$, where $\mathbf{O}^{n\times k}$ is an $n \times k$ zero matrix
2: $\mathbf{K} = \mathbf{W} \oplus \hat{\mathbf{R}}_2$
3: **for** $j = 1$; $j \le n$; j++ **do**
4: $\hat{\mathbf{z}}[j] = \sum_{i=1}^{n} \mathbf{K}[i,j]$
5: **end for**

Example II

We use the particular generic encoder shown in Example I, and thus $k = 2, m = 4, n = 8$ and all the variables are in $\mathbb{F}_2$. The corresponding pre-computed matrices

$\tilde{\mathbf{A}}$ and $\mathbf{M}_1, \ldots, \mathbf{M}_8$ are:

$$\tilde{\mathbf{A}} = \begin{bmatrix} 010000 \\ 010000 \\ 110000 \\ 010000 \\ 010000 \\ 000000 \\ 100000 \\ 100000 \end{bmatrix}, \qquad \mathbf{M}_1 = \mathbf{M}_2 = \mathbf{M}_4 = \mathbf{M}_5 = \begin{bmatrix} 010000 \\ 010000 \\ 010000 \\ 010000 \\ 010000 \\ 000000 \\ 000000 \\ 000000 \end{bmatrix},$$

$$\mathbf{M}_3 = \begin{bmatrix} 010000 \\ 010000 \\ 110000 \\ 010000 \\ 010000 \\ 000000 \\ 100000 \\ 100000 \end{bmatrix}, \qquad \mathbf{M}_6 = \begin{bmatrix} 000000 \\ 000000 \\ 000000 \\ 000000 \\ 000000 \\ 000000 \\ 000000 \\ 000000 \end{bmatrix}, \qquad \mathbf{M}_7 = \mathbf{M}_8 = \begin{bmatrix} 000000 \\ 000000 \\ 100000 \\ 000000 \\ 000000 \\ 000000 \\ 100000 \\ 100000 \end{bmatrix}.$$

Note that by construction, the entries of both $\tilde{\mathbf{A}}[*, 3:6]$ and $\mathbf{M}_i[*, 3:6]$ for $i \in \{1, \ldots, 8\}$ are all zeros. We consider the secret inputs $\mathbf{x} = [1, 1]$ and $\mathbf{y} = [1, 0]$. Without loss of generality, we assume that all the random bits are 1s. The codewords are:

$$\hat{\mathbf{x}} = [1, 1, 1, 1, 1, 1] \times \mathbf{A} = [0, 0, 0, 1, 0, 1, 1, 0], \text{ and}$$

$$\hat{\mathbf{y}} = [1, 0, 1, 1, 1, 1] \times \mathbf{A} = [0, 0, 0, 1, 1, 0, 1, 0].$$

Then, all the internal variables are:

$$\mathbf{R}_1 = \mathbf{A}^{\mathrm{T}} \times \begin{bmatrix} 00000000 \\ 00000000 \\ 11111111 \\ 11111111 \\ 11111111 \\ 11111111 \end{bmatrix} = \begin{bmatrix} 11111111 \\ 11111111 \\ 11111111 \\ 00000000 \\ 11111111 \\ 00000000 \\ 11111111 \\ 00000000 \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \end{bmatrix} \times \mathbf{A} = \begin{bmatrix} 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \end{bmatrix},$$

$$\mathbf{S} = \begin{bmatrix} 11111111 \\ 11111111 \\ 11111111 \\ 00011010 \\ 11111111 \\ 00011010 \\ 11100101 \\ 00000000 \end{bmatrix}, \ \mathbf{T} = \begin{bmatrix} 010000 \\ 010000 \\ 110000 \\ 000000 \\ 010000 \\ 000000 \\ 000000 \\ 000000 \end{bmatrix}, \ \mathbf{W} = \begin{bmatrix} 00001100 \\ 00001100 \\ 11111100 \\ 00000000 \\ 00001100 \\ 00000000 \\ 00000000 \\ 00000000 \end{bmatrix}, \ \mathbf{K} = \begin{bmatrix} 11100110 \\ 11100110 \\ 00010110 \\ 11101010 \\ 11100110 \\ 11101010 \\ 11101010 \\ 11101010 \end{bmatrix}.$$

Note that every column (resp., row) of $\hat{\mathbf{R}}_1$ (resp., $\hat{\mathbf{R}}_2$) is a codeword of $\mathbf{0}$. At last, Gadget 1 outputs $\hat{\mathbf{z}} = [1, 1, 1, 1, 0, 0, 0, 0]$, which is a codeword of $\mathbf{x} \odot \mathbf{y} = [1, 0]$. We can see that $\sum_{i=1}^{8} \mathbf{T}[i, 1:2] = [1, 0]$, which confirms that columns of $\mathbf{T}[*, 1:k]$ are additive sharings of the secret result (this argument will be formally proved in next sub-section).

4.3 Correctness of CodeMul gadget

First of all, we give the formulas and the proof about the relation between $\mathbf{M}_1, \ldots, \mathbf{M}_n$ and result codeword. Based on the formulas, we then give the correctness proof of Gadget 1.

Proposition 6. *Let* $\hat{\mathbf{x}} \in \mathbb{F}_q^n, \hat{\mathbf{y}} \in \mathbb{F}_q^n$. *Let* $\mathbf{E} \in \mathbb{F}_q^{(k+m)^2 \times (k+m)}$ *be the concatenation of the* $(k+m)^2$*-length* $\mathbf{e}_i^{\mathrm{T}}$ *for* $i \in \{j(k+m)+j+1 \mid 0 \le j < k+m\}$. *Let* $\tilde{\mathbf{A}} \in \mathbb{F}_q^{n \times (k+m)}$ *and* $\mathbf{M}_i \stackrel{def}{=} (\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}})\mathbf{E}$ *for* $i \in [n]$. *Then, the following holds:*

$$\sum_{i=1}^{n} \big(\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}}\big)\mathbf{M}_i = \hat{\mathbf{x}}\tilde{\mathbf{A}} \odot \hat{\mathbf{y}}\tilde{\mathbf{A}}.$$

Typically, if for $\mathbf{A} \in \mathbb{F}_q^{(k+m)\times n}$ *and* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^k$, $\hat{\mathbf{x}} \stackrel{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x}), \hat{\mathbf{y}} \stackrel{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y}) \in \mathbb{F}_q^n$, *and* $\tilde{\mathbf{A}} = \big[\mathbf{A}^{-1}[*, 1\!:\!k], \mathbf{O}^{n\times m}\big]$, *then the following holds:*

$$\sum_{i=1}^{n} \big(\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}}\big)\mathbf{M}_i = [\mathbf{x} \odot \mathbf{y}, \mathbf{0}], \text{ where } \mathbf{0} \in \mathbb{F}_q^m.$$

Proof. We start with proving the first equation. Let $i \in [n]$, first using the definition of $\mathbf{M}_i$, we have $(\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}})\mathbf{M}_i = (\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}})(\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}})\mathbf{E}$. Then, we can apply Proposition 1, giving: $(\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}})(\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}}) = (\hat{\mathbf{x}}[i]\tilde{\mathbf{A}}[i,*]) \otimes (\hat{\mathbf{y}}\tilde{\mathbf{A}})$. Thus, we have:

$$\begin{aligned}
\sum_{i=1}^{n} \big(\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}}\big)\mathbf{M}_i &= \sum_{i=1}^{n} \Big(\big(\hat{\mathbf{x}}[i]\tilde{\mathbf{A}}[i,*]\big) \otimes \big(\hat{\mathbf{y}}\tilde{\mathbf{A}}\big)\Big)\mathbf{E} \\
&= \Big(\big(\sum_{i=1}^{n} \hat{\mathbf{x}}[i]\tilde{\mathbf{A}}[i,*]\big) \otimes \big(\hat{\mathbf{y}}\tilde{\mathbf{A}}\big)\Big)\mathbf{E} \\
&= \big((\hat{\mathbf{x}}\tilde{\mathbf{A}}) \otimes (\hat{\mathbf{y}}\tilde{\mathbf{A}})\big)\mathbf{E}.
\end{aligned}$$

Multiplying a $(k+m)^2$-length vector by the matrix $\mathbf{E}$ corresponds to selecting its elements with indexes $j(k+m)+j+1$ for $j \in [k+m-1]$. More particularly, for a vector obtained by the tensor product of two $(k+m)$-length vectors, multiplying by $\mathbf{E}$ gives the vectors of cross-products with equal indexes. Then, we get:

$$\begin{aligned}
\sum_{i=1}^{n} (\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}})\mathbf{M}_i &= \big((\hat{\mathbf{x}}\tilde{\mathbf{A}}) \otimes (\hat{\mathbf{y}}\tilde{\mathbf{A}})\big)\mathbf{E} \\
&= \hat{\mathbf{x}}\tilde{\mathbf{A}} \odot \hat{\mathbf{y}}\tilde{\mathbf{A}}.
\end{aligned}$$

By Lemma 5, $\hat{\mathbf{x}}\mathbf{A}^{-1}[*, 1\!:\!k] = \mathbf{x}$, and thus $\hat{\mathbf{x}}\tilde{\mathbf{A}} = \big[\mathbf{A}^{-1}[*, 1\!:\!k], \mathbf{O}^{n\times m}\big] = [\mathbf{x}, \mathbf{0}]$ (similarly, $\hat{\mathbf{y}}\tilde{\mathbf{A}} = [\mathbf{y}, \mathbf{0}]$), giving that $\hat{\mathbf{x}}\tilde{\mathbf{A}} = [\mathbf{x}, \mathbf{0}]$ and $\hat{\mathbf{y}}\tilde{\mathbf{A}} = [\mathbf{y}, \mathbf{0}]$. □

In the following, we prove the correctness of Gadget 1.

Theorem 1. *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^k$, $\mathsf{Enc}_{\mathbf{A}}$ *be a generic encoder such that* $\mathbf{A} \in \mathbb{F}_q^{(k+m)\times n}$. *Then, Gadget 1 with inputs* $\hat{\mathbf{x}} \stackrel{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$, $\hat{\mathbf{y}} \stackrel{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y})$ *and output* $\hat{\mathbf{z}}$ *ensures that*

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathbf{x} \odot \mathbf{y}.$$

Moreover, Gadget 1 also ensures that

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathbf{x} \odot \mathbf{y} = \sum_{i=1}^{n} \mathbf{T}[i, 1\!:\!k],$$

where $\mathbf{T}$ *is the output of the part A.*

Proof. We aim at proving that for any $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ and $\hat{\mathbf{y}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y})$, the gadget produces $\hat{\mathbf{z}}$ such that $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathbf{x} \odot \mathbf{y}$. We begin by rewriting $\hat{\mathbf{z}}$ in terms depending on $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\mathbf{M}_i$ and the random values, going from its final expression and successively climbing the instructions of parts C, B, and A.

Following the instructions of part C:

$$\begin{aligned}\hat{\mathbf{z}} &= \left(\sum_{i=1}^{n} \mathbf{K}[i,1], \ldots, \sum_{i=1}^{n} \mathbf{K}[i,n]\right), \\ &= \left(\sum_{i=1}^{n} \left(\mathbf{W}[i,1] \oplus \hat{\mathbf{R}}_2[i,1]\right), \ldots, \sum_{i=1}^{n} \left(\mathbf{W}[i,n] \oplus \hat{\mathbf{R}}_2[i,n]\right)\right), \\ &= \sum_{i=1}^{n} \mathbf{W}[i,*] \oplus \sum_{i=1}^{n} \hat{\mathbf{R}}_2[i,*].\end{aligned}$$

The part B gives:

$$\hat{\mathbf{z}} = \sum_{i=1}^{n} \left(\mathbf{T}[i,*] \times \mathbf{A}\right) \oplus \sum_{i=1}^{n} \hat{\mathbf{R}}_2[i,*].$$

By the linearity of the code, we have:

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathsf{Dec}_{\mathbf{A}}\left(\sum_{i=1}^{n} \left(\mathbf{T}[i,*] \times \mathbf{A}\right)\right) \oplus \mathsf{Dec}_{\mathbf{A}}\left(\sum_{i=1}^{n} \hat{\mathbf{R}}_2[i,*]\right).$$

We first consider the second part of the right-hand side of the equation. By construction, $\hat{\mathbf{R}}_2[i,*] \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{0})$ for $i \in [n]$, then they decode to $\mathbf{0}$ with $\mathsf{Dec}_{\mathbf{A}}$, and the sum decodes to $\mathbf{0}$ due to the linearity of the code. Then, we consider the first part of the right-hand side. By the definition of $\mathsf{Dec}_{\mathbf{A}}$, for any $\mathbf{a} \in \mathbb{F}_q^{m+k}$, we have $\mathsf{Dec}_{\mathbf{A}}(\mathbf{a} \times \mathbf{A}) = \mathbf{a}[1\!:\!k]$, then:

$$\mathsf{Dec}_{\mathbf{A}}\left(\sum_{i=1}^{n} \left(\mathbf{T}[i,*] \times \mathbf{A}\right)\right) = \sum_{i=1}^{n} \left(\mathsf{Dec}_{\mathbf{A}}(\mathbf{T}[i,*] \times \mathbf{A})\right) = \sum_{i=1}^{n} \left(\mathbf{T}[i,1\!:\!k]\right).$$

Therefore, we obtain:

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \sum_{i=1}^{n} \mathbf{T}[i,1\!:\!k]. \tag{3}$$

Considering the part A of the gadget:

$$\begin{aligned}\sum_{i=1}^{n} \mathbf{T}[i,1\!:\!k] &= \sum_{i=1}^{n} \left(\mathbf{S}[i,*] \times \mathbf{M}_i\right)[1\!:\!k], \\ &= \sum_{i=1}^{n} \left(\left(\left(\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}}\right) \oplus \hat{\mathbf{R}}_1[i,*]\right) \mathbf{M}_i\right)[1\!:\!k], \\ &= \sum_{i=1}^{n} \left(\left(\hat{\mathbf{x}}[i] \otimes \hat{\mathbf{y}}\right) \times \mathbf{M}_i\right)[1\!:\!k] \oplus \sum_{i=1}^{n} \left(\hat{\mathbf{R}}_1[i,*] \times \mathbf{M}_i\right)[1\!:\!k]. \end{aligned} \tag{4}$$

We first show that the second part of the right-hand side of Equation 4 is equal to the zero vector. As $\hat{\mathbf{R}}_1$ is a matrix over $\mathbb{F}_q^{n \times n}$, applying Proposition 3, $\hat{\mathbf{R}}_1$ can be rewritten as:

$$\hat{\mathbf{R}}_1 = \sum_{j=1}^{n} (\hat{\mathbf{R}}_1[*,j] \otimes \mathbf{e}_j).$$

Then:

$$\hat{\mathbf{R}}_1[i,*] = \left(\sum_{j=1}^{n}(\hat{\mathbf{R}}_1[*,j]\otimes\mathbf{e}_j)\right)[i,*] = \sum_{j=1}^{n}(\hat{\mathbf{R}}_1[*,j]\otimes\mathbf{e}_j)[i,*] = \sum_{j=1}^{n}(\hat{\mathbf{R}}_1[i,j]\otimes\mathbf{e}_j).$$

Consequently, using the Proposition 2 the second part can be written as:

$$\begin{aligned}\sum_{i=1}^{n}\left(\hat{\mathbf{R}}_1[i,*]\times\mathbf{M}_i[*,1:k]\right) &= \sum_{i=1}^{n}\left(\left(\sum_{j=1}^{n}(\hat{\mathbf{R}}_1[i,j]\otimes\mathbf{e}_j)\right)\mathbf{M}_i[*,1:k]\right),\\ &= \sum_{j=1}^{n}\sum_{i=1}^{n}(\hat{\mathbf{R}}_1[i,j]\otimes\mathbf{e}_j)\mathbf{M}_i[*,1:k],\end{aligned}$$

Applying Proposition 6 (the first equation): $\sum_{i=1}^{n}(\hat{\mathbf{R}}_1[i,j]\otimes\mathbf{e}_j)\mathbf{M}_i[*,1:k] = \mathbf{0}\in\mathbb{F}_q^{k+m}$, Therefore,

$$\sum_{i=1}^{n}\left(\hat{\mathbf{R}}_1[i,*]\times\mathbf{M}_i[*,1:k]\right) = \mathbf{0}.$$

Now Equation 4 can be rewritten as:

$$\sum_{i=1}^{n}\mathbf{T}[i,1:k] = \left(\sum_{i=1}^{n}\left(\left(\hat{\mathbf{x}}^{\mathrm{T}}\otimes\hat{\mathbf{y}}\right)[i,*]\times\mathbf{M}_i\right)\right)[1:k].$$

By applying Proposition 6 (the second equation) we get $\sum_{i=1}^{n}\left((\hat{\mathbf{x}}[i]\otimes\hat{\mathbf{y}})\times\mathbf{M}_i\right) = (\mathbf{x}\odot\mathbf{y},\mathbf{0})$, where $\mathbf{0}\in\mathbb{F}_q^m$. Accordingly, the first term of Equation 4 is only $\mathbf{x}\odot\mathbf{y}$. It allows us to conclude:

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \sum_{i=1}^{n}\mathbf{T}[i,1:k] = \mathbf{x}\odot\mathbf{y}.$$

□

4.4 Security of CodeMul gadget

In the following part, we prove the security of Gadget 1. First, Proposition 7 shows some properties about linear combinations of a limited number of entries from the intermediate matrices $\mathbf{S}$ and $\mathbf{K}$ used in Gadget 1. Then, we give three lemmas, one for each part A, B and C of Gadget 1 to show how internal probes can be simulated with the input of each part. Finally, Theorem 2 combines the three lemmas to prove that Gadget 1 instantiated with a d-private generic encoder is t-SNI (with $0\le t\le d$).

Proposition 7. *Let $\mathsf{Enc}_{\mathbf{A}}$ be a d-private generic encoder. When it is fed with codewords $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ (resp. with any $\hat{\mathbf{x}},\hat{\mathbf{y}}\in\mathbb{F}_q^n$), Gadget 1 ensures that any linear combination of the entries from $\mathbf{S}[\mathcal{I},*]$ (resp., $\mathbf{K}[*,\mathcal{J}]$) is either uniformly distributed over $\mathbb{F}_q$ or equal to 0, for any $\mathcal{I}\subseteq[n], |\mathcal{I}|\le d$ (resp. $\mathcal{J}\subseteq[n]$, $|\mathcal{J}|\le d$).*

Proof. We first give the proof regarding rows of $\mathbf{S}$. By instructions of the Gadget 1, for $j\in[n]$, we have $\mathbf{S}[*,j] = \hat{\mathbf{x}}^{\mathrm{T}}\hat{\mathbf{y}}[j]\oplus\hat{\mathbf{R}}_1[*,j]$, where $\hat{\mathbf{x}}\overset{\$}{=}\mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$, and $\hat{\mathbf{R}}_1[*,j]\overset{\$}{=}\mathsf{Enc}_{\mathbf{A}}(\mathbf{0})^{\mathrm{T}}$. Since $\mathsf{Enc}_{\mathbf{A}}$ is a linear code, and a fresh codeword of $\mathbf{0}$ masks the secret data:

$$\mathbf{S}[*,j]\overset{\$}{=}\mathsf{Enc}_{\mathbf{A}}(\mathbf{x})^{\mathrm{T}}\hat{\mathbf{y}}[j]\oplus\mathsf{Enc}_{\mathbf{A}}(\mathbf{0})^{\mathrm{T}}\overset{\$}{=}\mathsf{Enc}_{\mathbf{A}}(\mathbf{x}\hat{\mathbf{y}}[j])^{\mathrm{T}}.$$

Using that $\mathsf{Enc}_{\mathbf{A}}$ is a d-private generic encoder, applying Corollary 1 gives that any linear combination of at most d entries from each column of $\mathbf{S}$ is either uniformly distributed over $\mathbb{F}_q$ or equal to 0. A linear combination of entries from at most d rows of $\mathbf{S}$ can be rewritten as: $h(\mathbf{S}) = \sum_{j\in[n]} a_j h_j(\mathbf{S}[\mathcal{I}, j])$, where $|\mathcal{I}| \leq d$, $a_j \in \mathbb{F}_q$ and $h_j(\mathbf{S}[\mathcal{I}, j])$ is a linear combination of entries in $\mathbf{S}[\mathcal{I}, j]$. We have following results:

- If $h_j(\mathbf{S}[\mathcal{I}, j]) = 0$ for every $j \in [n]$, then $h(\mathbf{S}) = 0$.
- If $h_j(\mathbf{S}[\mathcal{I}, j]) \neq 0$ for at least one $j \in [n]$, then $h_j(\mathbf{S}[\mathcal{I}, j])$ is uniformly distributed, and thus we have:
  - If $a_j = 0$ for every j such that $h_j(\mathbf{S}[\mathcal{I}, j]) \neq 0$, then $h(\mathbf{S}) = 0$.
  - If not, there exists $j \in [n]$ such that $a_j h_j(\mathbf{S}[\mathcal{I}, j])$ is uniformly distributed, whose random coins are from $\hat{\mathbf{R}}_1[\mathcal{I}, j]$. As different columns of $\hat{\mathbf{R}}_1$ are independent, $h(\mathbf{S}) = \sum_{j\in[n]} a_j h_j(\mathbf{S}[\mathcal{I}, j])$ is uniformly distributed.

We then give the proof regarding columns of $\mathbf{K}$. By instruction of the Gadget 1, for $i \in [n]$, we have $\mathbf{T}[i, *] = \mathbf{S}[i, *] \times \mathbf{M}_i$. By the definition of $\mathbf{M}_i$:

$$\mathbf{T}[i, *] = \mathbf{S}[i, *] \times \big((\tilde{\mathbf{A}} \otimes \tilde{\mathbf{A}})\mathbf{E}\big)[1+(i-1)n : in, *].$$

As $\tilde{\mathbf{A}} = [\mathbf{A}^{-1}[*, k], \mathbf{O}^{n\times m}]$, we have $\mathbf{T}[i, k+1 : m+k] = \mathbf{0}$. Finally, the i^{th} row of $\mathbf{K}$ can be rewritten as:

$$\mathbf{K}[i, *] = \mathbf{W}[i, *] \oplus \hat{\mathbf{R}}_2[i, *] = \mathbf{T}[i, *] \times \mathbf{A} \oplus \hat{\mathbf{R}}_2[i, *].$$

Since $\hat{\mathbf{R}}_2[i, *] \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{0})$ and $\mathsf{Enc}_{\mathbf{A}}$ is a d-private generic encoder, we have:

$$\mathsf{Enc}_{\mathbf{A}}(\mathbf{T}[i, 1 : k]) \oplus \hat{\mathbf{R}}_2[i, *] \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{T}[i, 1 : k]).$$

Thus, $\mathbf{K}[i, *] \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{T}[i, 1 : k])$. Then, by Corollary 1, any d entries from each row of $\mathbf{K}$ is either uniformly distributed over $\mathbb{F}_q$ or equal to 0. A linear combination of entries of at most d columns of $\mathbf{K}$ can be rewritten as $h(\mathbf{K}) = \sum_{i\in[n]} a_i h_i(\mathbf{K}[i, \mathcal{J}])$, where $|\mathcal{J}| \leq d$, $a_i \in \mathbb{F}_q$ and $h_j(\mathbf{K}[i, \mathcal{J}])$ is a linear combination of entries in $\mathbf{K}[i, \mathcal{J}]$. We have following results:

- If $h_i(\mathbf{K}[i, \mathcal{J}]) = 0$ for every $i \in [n]$, then $h(\mathbf{K}) = 0$.
- If $h_i(\mathbf{K}[i, \mathcal{J}]) \neq 0$ for at least one $i \in [n]$, then $h_i(\mathbf{K}[i, \mathcal{J}])$ is uniformly distributed for any $i \in [n]$, and thus we have:
  - If $a_i = 0$ for every i such that $h_i(\mathbf{K}[i, \mathcal{J}]) \neq 0$, then $h(\mathbf{K}) = 0$.
  - If not, then there exists $i \in [n]$ such that $a_i h_i(\mathbf{K}[i, \mathcal{J}])$ is uniformly distributed, whose random coins are from $\hat{\mathbf{R}}_2[i, \mathcal{J}]$. As different rows of $\hat{\mathbf{R}}_2$ are independent, $h(\mathbf{K}) = \sum_{i\in[n]} a_i h_i(\mathbf{K}[i, \mathcal{J}])$ is uniformly distributed.

□

Theorem 2 can be deducted by the security of three parts of Gadget 1, which we give in Lemmas 6, 7 and 8 respectively.

Lemma 6. *Let $\mathsf{Enc}_{\mathbf{A}}$ be a d-private generic encoder. For Gadget 1 part A, any t_{int} internal probes and all outputs in $\mathbf{T}[\mathcal{O}, *]$ such that $t_{int} + |\mathcal{O}| = t \leq d$, can be simulated under valid input sharings with inputs shares $\hat{\mathbf{x}}[\mathcal{I}]$ and $\hat{\mathbf{y}}[\mathcal{J}]$ for $|\mathcal{I}| \leq t, |\mathcal{J}| \leq t$.*

Proof. Let $\mathcal{P}$ denote the set of probes. We aim at proving that for all output probes $\mathbf{T}[\mathcal{O}, *]$ ($|\mathcal{O}|(m+k)$ output probes) and internal probes $\mathcal{P}_{int}$ such that $|\mathcal{P}_{int}| + |\mathcal{O}| \leq t$, there exists sets $\mathcal{I} \subseteq [n]$ and $\mathcal{J} \subseteq [n]$ such that $|\mathcal{I}| \leq t$ and $|\mathcal{J}| \leq t$, and a simulator S such that:

$$\Pr\left[\mathsf{D}\left(\mathsf{G}_{\mathcal{P}}\left(\hat{\mathbf{x}}, \hat{\mathbf{y}}\right), \hat{\mathbf{x}}, \hat{\mathbf{y}}\right) = 1\right] = \Pr\left[\mathsf{D}\left(\mathsf{S}\left(\hat{\mathbf{x}}[\mathcal{I}], \hat{\mathbf{y}}[\mathcal{J}]\right), \hat{\mathbf{x}}, \hat{\mathbf{y}}\right) = 1\right], \text{ for any distinguisher } \mathsf{D}.$$

In the following proof, we start with showing how to build the sets $\mathcal{I}$ and $\mathcal{J}$, and then investigate simulator.

Building the sets $\mathcal{I}$ and $\mathcal{J}$. First we divide all the possible probes on Gadget 1 part A into different subsets based on the types:

- The probes to the input shares: $\mathcal{P}_{input}$.
- The probes to the random variables: $\mathcal{P}_{rand}$, which include the entries in $\hat{\mathbf{R}}_1$, variables in the computation of $\left([\mathbf{O}^{n\times k}, \mathbf{R}_1]\mathbf{A}\right)^{\mathrm{T}}$, and entries in $\mathbf{R}_1$. In the second case, each row of $[\mathbf{O}^{n\times k}, \mathbf{R}_1]$ is linearly transformed to a distinct column of $\hat{\mathbf{R}}_1$. Thus, for each probe p in $\mathcal{P}_{rand}$, there exists a function of linear combination $f : \mathbb{F}_q^m \to \mathbb{F}_q$ and an index $j \in [n]$, such that $p = f(\mathbf{R}_1^{\mathrm{T}}[*, j])$.
- The probes of $\hat{\mathbf{x}}[i]\hat{\mathbf{y}}[j]$ for any $i, j \in [n]$: $\mathcal{P}_{xy}$.
- The probes to the computation from $\mathbf{S}$ to $\mathbf{T}$: $\mathcal{P}_{\mathbf{S}\to\mathbf{T}}$, where $\mathbf{S}[i, *]$ is multiplied by $\mathbf{M}_i$ to $\mathbf{T}[i, *]$. Thus, for each probe p in $\mathcal{P}_{\mathbf{S}\to\mathbf{T}}$, there exists a linear combination $g : \mathbb{F}_q^n \to \mathbb{F}_q$ and an index $i \in [n]$, such that $p = g(\mathbf{S}[i, *])$. Note that is also encompasses the probes on one $\mathbf{S}[i, j]$.
- The probes to the shares of output codewords $\mathbf{T}$: $\mathcal{P}_{out}$, whose corresponding row indices are in a set $\mathcal{O}$.

Note that the internal probes are $\mathcal{P}_{int} = \mathcal{P}_{input} \cup \mathcal{P}_{rand} \cup \mathcal{P}_{xy} \cup \mathcal{P}_{\mathbf{S}\to\mathbf{T}}$ and the output probes are $\mathcal{P}_O \subseteq \mathbf{T}[\mathcal{O}, *]$, and then we have $\mathcal{P}_{int} + |\mathcal{O}| \leq t$, $|\mathcal{P}_{int}| \leq t_{int}$ and $|\mathcal{P}_O| \leq |\mathcal{O}|(m+k)$.

To build the sets $\mathcal{I}$ and $\mathcal{J}$, note that except for the subset $\mathcal{P}_{rand}$, each other internal probes, or all entries of each row of $\mathbf{T}$ depend on at most one index i which relates to the input codeword $\hat{\mathbf{x}}$. Then $\mathcal{I}$ is built as the union of these indexes, and since $|\mathcal{P}_{int}| + |\mathcal{O}| \leq t$, it ensures $|\mathcal{I}| \leq t$. The set $\mathcal{J}$ is built as follow, for each probe in $\mathcal{P}_{input}$ on the codeword $\hat{\mathbf{y}}$: $\hat{\mathbf{y}}[j]$, j is added to $\mathcal{J}$. As seen above, any probe in $\mathcal{P}_{rand}$ can be written as $p = f(\mathbf{R}_1^{\mathrm{T}}[*, j])$, then this index is added to $\mathcal{J}$. For each probe in set $\mathcal{P}_{xy}$, *i.e.,* $\hat{\mathbf{x}}[i]\hat{\mathbf{y}}[j]$, we add j into into $\mathcal{J}$. Since $|\mathcal{P}_{input}| + |\mathcal{P}_{rand}| + |\mathcal{P}_{xy}| \leq t_{int}$, we get $|\mathcal{J}| \leq t$. Note also that, by construction, for each probe there is at least one index in $\mathcal{I}$ or in $\mathcal{J}$ (or in both for $\mathcal{P}_{xy}$).

Simulation. We define the simulator S which runs Gadget 1 part A by feeding the values of $\hat{\mathbf{x}}[\mathcal{I}]$ and $\hat{\mathbf{y}}[\mathcal{J}]$ and attributing random values to the other input shares, and outputs the probes. We show that the distribution of the probes provided by the simulator is identical to the distribution the adversary would obtain in a real attack. To achieve it, we use the equivalence of Lemma 1, and show that any linear combinations of the probes given by the simulator is identically distributed as the same linear combination of the probes obtained by the adversary. For each possible combination, we prove that it relates only to the input shares $\{\hat{\mathbf{x}}[\mathcal{I}], \hat{\mathbf{y}}[\mathcal{J}]\}$ and the random coins in the generation of $\mathbf{R}_1$.

First, notice that each linear combination of probes h is a combinations of the following terms only: $\hat{\mathbf{x}}[i]$, $\hat{\mathbf{y}}[j]$, $\hat{\mathbf{x}}[i]\hat{\mathbf{y}}[j]$, $\mathbf{R}_1^{\mathrm{T}}[*, j]$, $\mathbf{S}[i, j]$ and $\mathbf{S}[i, j']$, where $i \in \mathcal{I}$, $j \in \mathcal{J}$, and $j' \in \bar{\mathcal{J}}$. Let us call h_1 the part of the combination with the terms $\mathbf{S}[i, j']$ and h_2 the remaining part, any linear combination h can be written as $a_1 h_1 \oplus a_2 h_2$ with $a_1 \in \mathbb{F}_q$, $a_2 \in \mathbb{F}_q$. The distributions of h_1 and h_2 are independent as they are relative to different columns of $\mathbf{R}_1^{\mathrm{T}}$, which are generated from independent random coins. Then, since h_1 is a linear combination of terms $\mathbf{S}[i, j']$ for $i \in \mathcal{I}, j' \notin \mathcal{J}$ and $|\mathcal{I}| \leq t \leq d$, applying Proposition 7, we

get that h_1 is either uniformly distributed or equal to 0. Consequently, in both cases $a_1 h_1$ can be simulated without knowledge on input variables. Finally, the terms in $a_2 h_2$ only relate to $\hat{\mathbf{x}}[\mathcal{I}]$ and $\hat{\mathbf{y}}[\mathcal{J}]$ and random coins in the generation of $\mathbf{R}_1$, which can be simulated with $\hat{\mathbf{x}}[\mathcal{I}]$ and $\hat{\mathbf{y}}[\mathcal{J}]$ only. Thus, the whole linear combination h can be simulated with the input shares $\{\hat{\mathbf{x}}[\mathcal{I}], \hat{\mathbf{y}}[\mathcal{J}]\}$.

To summarize, for any output probes $\mathcal{P}_{out}$ whose indexes are in $\mathcal{O}$ and internal probes $\mathcal{P}_{int}$ such that $|\mathcal{P}_{int}| + |\mathcal{O}| \leq t$, we can build the sets $\mathcal{I}$ and $\mathcal{J}$ such that $|\mathcal{I}| \leq |\mathcal{P}|$ and $|\mathcal{J}| \leq |\mathcal{P}|$, and there exists a simulator S such that for any distinguisher D:

$$\Pr\left[\mathsf{D}\left(\mathsf{G}_{\mathcal{P}}\left(\hat{\mathbf{x}}, \hat{\mathbf{y}}\right), \hat{\mathbf{x}}, \hat{\mathbf{y}}\right) = 1\right] = \Pr\left[\mathsf{D}\left(\mathsf{S}\left(\hat{\mathbf{x}}[\mathcal{I}], \hat{\mathbf{y}}[\mathcal{J}]\right), \hat{\mathbf{x}}, \hat{\mathbf{y}}\right) = 1\right].$$

□

Lemma 7. *For Gadget 1 part B, any t_{int} internal probes and all outputs in $\mathbf{W}[\mathcal{O}, *]$ such that $t_{int} + |\mathcal{O}| = t$ can be simulated with all input variables of $\mathbf{T}[\mathcal{I}, *]$ for $|\mathcal{I}| \leq t$.*

Proof. We prove that for any output probes in $\mathbf{W}[\mathcal{O}, 1], \ldots, \mathbf{W}[\mathcal{O}, n]$ and internal probes $\mathcal{P}_{int}$ such that $|\mathcal{O}| + |\mathcal{P}_{int}| \leq t$, there exists a set $\mathcal{I}$ such that $|\mathcal{I}| \leq |\mathcal{P}_{int}| + |\mathcal{O}| \leq t$, and a simulator S such that for any distinguisher D:

$$\Pr\left[\mathsf{D}\left(\mathsf{G}_{\mathcal{P}}\left(\mathbf{T}[*, *]\right), \mathbf{T}[*, *]\right) = 1\right] = \Pr\left[\mathsf{D}\left(\mathsf{S}\left(\mathbf{T}[\mathcal{I}, *]\right), \mathbf{T}[*, *]\right) = 1\right],$$

where $\mathcal{P}$ is the set of all the probes.

Since Gadget 1-B performs a linear operation on shares with same index (each row of $\mathbf{T}$ is multiplied by a public matrix), each probe only relates to the input shares with one index. Thus, the set $\mathcal{I}$ can be built by taking the corresponding indexes of all the probes, and we have $|\mathcal{I}| \leq |\mathcal{P}| \leq t$. The simulator S runs the Gadget 1 part B by feeding the values of the input shares indexed by $\mathcal{I}$ and attributing random values to the other input shares, and outputs the probes. Therefore, the probes only relate to $\mathbf{T}[\mathcal{I}, *]$, finishing the proof. □

Lemma 8. *Let $\mathsf{Enc}_{\mathbf{A}}$ be a d-private generic encoder. Gadget 1 part C, any t_{int} internal variables and t_{out} output variables such that $t_{int} + t_{out} = t \leq d$, can be simulated with all input variables of $\mathbf{W}[\mathcal{I}, *]$ where $\mathcal{I} \subseteq [n]$, $|\mathcal{I}| \leq t_{int}$.*

Proof. The proof is similar to the one of Lemma 6. Let $\mathcal{P}$ denote the set of probes, we prove that for any t_{out} output probes and t_{int} internal probes such that $t_{int} + t_{out} \leq t$, there exists a set $\mathcal{I} \subseteq [n]$ such that $|\mathcal{I}| \leq t_{int}$, and a simulator S such that:

$$\begin{aligned} &\Pr\left[\mathsf{D}\left(\mathsf{G}_{\mathcal{P}}\left(\mathbf{W}[*, *]\right), \mathbf{W}[*, *]\right) = 1\right] \\ =&\Pr\left[\mathsf{D}\left(\mathsf{S}\left(\mathbf{W}[\mathcal{I}, *]\right), \mathbf{W}[*, *]\right) = 1\right], \text{ for any distinguisher } \mathsf{D}. \end{aligned}$$

In the following proof, we start with showing how to build the set $\mathcal{I}$, and then investigate simulator.

Building the set $\mathcal{I}$. First we divide all the possible probes on Gadget 1 part C into different subsets based on the types:

- The probes to the input shares: $\mathcal{P}_{input}$.
- The probes to the random variables: $\mathcal{P}_{rand}$. The probes include entries in $\mathbf{R}_2$, variables in the computation of $[\mathbf{O}^{n \times k}, \mathbf{R}_2]\mathbf{A}$, and entries in $\hat{\mathbf{R}}_2$. In the second case, each row of $[\mathbf{O}^{n \times k}, \mathbf{R}_2]$ is linearly transformed to a distinct row of $\hat{\mathbf{R}}_2$. Thus, for each probe p in $\mathcal{P}_{rand}$, there exists linear combination $f : \mathbb{F}_q^m \to \mathbb{F}_q$ and an index $i \in [n]$, such that $p = f(\mathbf{R}_2[i, *])$.

- The probes to the calculation of $\sum_{i=1}^{n} \mathbf{K}[i,*]$: $\mathcal{P}_{sum}$. For each probe p in $\mathcal{P}_{sum}$, there exists a linear combination $g : \mathbb{F}_q^n \to \mathbb{F}_q$ and an index $j \in [n]$, such that the probe $p = g(\mathbf{K}[*,j])$.
- The probes to the output: $\mathcal{P}_O$.

Note that the internal probes are $\mathcal{P}_{int} = \mathcal{P}_{input} \cup \mathcal{P}_{rand} \cup \mathcal{P}_{sum}$ and the output probes are $\mathcal{P}_O$, giving that $|\mathcal{P}_{int}| + |\mathcal{P}_O| \leq t$ and $|\mathcal{P}_{int}| \leq t_{int}$.

The set $\mathcal{I}$ is built as follow, for each probe in $\mathcal{P}_{input}$: $\mathbf{W}[i,j]$, i is added to $\mathcal{I}$. As seen above, any probe in $\mathcal{P}_{rand}$ can be written as $p = f(\mathbf{R}_2[i,*])$, then i is added to $\mathcal{I}$. Since $|\mathcal{P}_{input}| + |\mathcal{P}_{rand}| \leq t_{int}$, we get $|\mathcal{I}| \leq t_{int}$. The set $\mathcal{J}$ is built from $\mathcal{P}_{sum}$: for each probe $p = g(\mathbf{K}[*,j])$ the index j is added to $\mathcal{J}$. As each of these probes relates to at most one value of j, we have $|\mathcal{J}| \leq t$.

Simulation. We define the simulator S which runs Gadget 1 part C by feeding the values of $\mathbf{W}[\mathcal{I},*]$ and attributing random values to the other input shares, and outputs the probes. We show that the distribution of the probes provided by the simulator is identical to the distribution the adversary would obtain in a real attack. To achieve it, we use the equivalence of Lemma 1, and we show that any linear combination of the probes given by the simulator is identically distributed as the same linear combination of the probes obtained by the adversary. For each possible combination, we prove that it relates only to the input shares $\mathbf{W}[\mathcal{I},*]$ and the random coins in the generation of $\mathbf{R}_2$.

First, notice that (by construction of the sets) each linear combination of probes h is a combinations of the following terms only: $\mathbf{W}[i,j]$, $\mathbf{K}[i,j]$, $\mathbf{R}_2[i,j]$ and $\mathbf{K}[i',j']$, where $i \in \mathcal{I}$, $j \in [n]$, $i' \in \bar{\mathcal{I}}$, and $j' \in \mathcal{J}$. Let us call h_1 the part of the combination with the terms $\mathbf{K}[i',j']$ and h_2 the remaining part, any linear combination h can be written as $a_1 h_1 \oplus a_2 h_2$ with $a_1 \in \mathbb{F}_q$, $a_2 \in \mathbb{F}_q$. The distributions of h_1 and h_2 are independent as they are relative to different rows of $\mathbf{R}_2$, which are generated from independent random coins. Then, since h_1 is a linear combination of terms $\mathbf{K}[i',j']$ for $i' \notin \mathcal{I}, j' \in \mathcal{J}$ and $|\mathcal{J}| \leq |\mathcal{P}| \leq d$, applying Proposition 7, we get that h_1 is either uniformly distributed or equal to 0. Consequently, in both cases $a_1 h_1$ can be simulated without knowledge on input variables. Finally, the terms in $a_2 h_2$ only relate to $\mathbf{W}[\mathcal{I},*]$ and random coins in the generation of $\mathbf{R}_2$, which can be simulated with $\mathbf{W}[\mathcal{I},*]$ only. Thus, the whole linear combination h can be simulated with the input shares $\mathbf{W}[\mathcal{I},*]$.

To summarize, for any t_{out} output probes and t_{int} internal probes $\mathcal{P}_{int}$ such that $t_{int} + t_{out} \leq t$, we can build the sets $\mathcal{I}$ such that $|\mathcal{I}| \leq t_{int}$, and there exists a simulator S such that for any distinguisher D:

$$\Pr\left[\mathsf{D}\left(\mathsf{G}_{\mathcal{P}}\left(\mathbf{W}[*,*]\right), \mathbf{W}[*,*]\right) = 1\right] = \Pr\left[\mathsf{D}\left(\mathsf{S}\left(\mathbf{W}[\mathcal{I},*]\right), \mathbf{W}[*,*]\right) = 1\right].$$

□

Finally, Theorem 2 assesses the security of Gadget 1, using the results of Lemmas 6, 7 and 8:

Theorem 2. *Let $\mathsf{Enc}_{\mathbf{A}}$ be a d-private generic encoder. Then, Gadget 1 is t-SNI for valid input sharings for any positive integer t such that $t \leq d$.*

Proof. We prove that any t_{out} output probes and internal probes $\mathcal{P}_{int}$ such that $|\mathcal{P}_{int}| + t_{out} \leq t$, can be simulated with $\hat{x}[\mathcal{I}], \hat{y}[\mathcal{J}]$ for $|\mathcal{I}| \leq |\mathcal{P}_{int}|$ and $|\mathcal{J}| \leq |\mathcal{P}_{int}|$. To do so, we begin by dividing the probes into different subsets depending on which part of the gadget they are. Then, we successively use Lemmas 8, 7 and 6 to obtain the final result.

The probes are divided into different subsets:

- The probes to the variables in part A, denoted as $\mathcal{P}_A$.
- The probes to the variables in part B, denoted as $\mathcal{P}_B$.

- The probes to the internal variables in part C, denoted as $\mathcal{P}_{C,int}$.
- The probes to the output variables in part C, denoted as $\mathcal{P}_{C,out}$.

Since $\mathsf{Enc}_{\mathbf{A}}$ is a d-private generic encoder and $\mathcal{P}_{C,int} + \mathcal{P}_{C,out} \leq t \leq d$ we can apply Lemma 8: It gives that the $\mathcal{P}_{C,int}$ and $\mathcal{P}_{C,out}$ probes can be simulated by all the variables of $\mathbf{W}[\mathcal{I}_C, *]$ where $\mathcal{I}_C \subseteq [n]$ is such that $|\mathcal{I}_C| \leq |\mathcal{P}_{C,int}|$. Then, since $|\mathcal{P}_B| + |\mathcal{I}_C| \leq t$, we can apply Lemma 7: $\mathbf{W}[\mathcal{I}_C, *]$ can be simulated from the variables of $\mathbf{T}[\mathcal{I}_B, *]$ where $\mathcal{I}_B \subseteq [n]$ is such that $|\mathcal{I}_B| \leq |\mathcal{I}_C| + |\mathcal{P}_B|$, and therefore $|\mathcal{I}_B| \leq |\mathcal{P}_B| + |\mathcal{P}_{C,int}|$. Moreover, since $|\mathcal{P}_A| + |\mathcal{I}_B| \leq |\mathcal{P}_{int}| \leq t \leq d$, Lemma 6 applies: It implies that the probes in $\mathcal{P}_A$ and the variables of $\mathbf{T}[\mathcal{I}_B, *]$ can be simulated with input shares $\hat{\mathbf{x}}[\mathcal{I}]$ and $\hat{\mathbf{y}}[\mathcal{J}]$ for $|\mathcal{I}| \leq |\mathcal{P}_{int}|, |\mathcal{J}| \leq |\mathcal{P}_{int}|$. In summary, $\mathcal{P}_A$, $\mathcal{P}_B$, $\mathcal{P}_{C,int}$ and $\mathcal{P}_{C,out}$ can be simulated with $\hat{\mathbf{x}}[\mathcal{I}], \hat{\mathbf{y}}[\mathcal{J}]$ for $|\mathcal{I}| \leq |\mathcal{P}_{int}|$ and $|\mathcal{J}| \leq |\mathcal{P}_{int}|$, finalizing the proof.

□

5 Addition gadget and more

In this section, we begin by giving the gadget for the addition of two elements of $\mathbb{F}_q^k$: it can be simply implemented by independently adding the shares with the same index.

Next, we consider a more general class of functions: for some $l, l' \in \mathbb{N} \setminus \{0\}$, $L \in \mathrm{L}^{l,l'}$ if and only if $L : \mathbb{F}_q^l \to \mathbb{F}_q^{l'}$ and if there exists a constant $\mathbf{c} \in \mathbb{F}_q^{l'}$ such that for any $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^l$, $L(\mathbf{x} \oplus \mathbf{y}) = L(\mathbf{x}) \oplus L(\mathbf{y}) \oplus \mathbf{c}$, Examples of such functions are

- affine transformations: $L(\mathbf{x}) = \mathbf{x}\mathbf{L} \oplus \mathbf{c}$ where $\mathbf{L} \in \mathbb{F}_q^{l \times l'}$ and $\mathbf{c} \in \mathbb{F}_q^{l'}$.
- coordinate-wise p^h power: $L(\mathbf{x}) = (\mathbf{x}[1]^{p^h}, \ldots, \mathbf{x}[l]^{p^h})$ where p is the characteristic of $\mathbb{F}_q$ and h is an integer greater than 0. By Lucas' Theorem (all binomial coefficients $\binom{p^h}{i}$ for $i \in [p^h - 1]$ are multiples of p):

$$\begin{aligned} L(\mathbf{x} \oplus \mathbf{y}) &= \left((\mathbf{x}[1] \oplus \mathbf{y}[1])^{p^h}, \ldots, (\mathbf{x}[k] \oplus \mathbf{y}[k])^{p^h}\right), \\ &= (\mathbf{x}[1]^{p^h} \oplus \mathbf{y}[1]^{p^h}, \ldots, \mathbf{x}[1]^{p^h} \oplus \mathbf{y}[1]^{p^h}), \\ &= L(\mathbf{x}) \oplus L(\mathbf{y}). \end{aligned}$$

- selections: $L(\mathbf{x}) = (\mathbf{x}[s_1], \ldots, \mathbf{x}[s_l])$, where all s_i belong to $[l]$.

In the second part of this Section, we design the CodeL family of gadgets that implement $\mathrm{L}^{k,k}$ functions, re-using ideas from the CodeMul gadget.

Any $\mathrm{L}^{kl,kl'}$ function can be evaluated in code-based masking by using a composition of CodeL and addition gadgets. That is, any $L \in \mathrm{L}^{lk,l'k}$ can be represented as $L(\mathbf{x}_1, \ldots, \mathbf{x}_l) = L^{(1)}(\mathbf{x}_1) \oplus \ldots \oplus L^{(l)}(\mathbf{x}_l) \oplus (l-1)\mathbf{c}$,[6] where $L^{(i)} \in \mathrm{L}^{k,kl'}$ can be viewed as a vector of l' $\mathrm{L}^{k,k}$ functions, and the addition can be implemented by the addition gadget.

This composition is not quite efficient, hence we give a more dedicated design in the third part of the Section: the CodeLs (standing for "multiple CodeL") family of gadgets that directly implement any $\mathrm{L}^{kl,kl'}$ function. Compared to the previous construction, this more dedicated design can reduce the cost and ease the implementation (*e.g.,* the masked AES round in Figure 5b of Section 6.3).

5.1 Addition gadget

The addition of two codewords $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ and $\hat{\mathbf{y}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y})$ can be trivially implemented in the same way as in the case of additive masking: performing a coordinate-wise addition. Gadget 2 gives the masked addition.

[6] With $L^{(i)} \overset{\text{def}}{=} L(0, \ldots, 0, \mathbf{x}_i, 0, \ldots, 0)$.

Gadget 2 Addition gadget: CodeAdd

Input: Codewords $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ and $\hat{\mathbf{y}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y})$ of $\mathbf{x} \in \mathbb{F}_q^k$ and $\mathbf{y} \in \mathbb{F}_q^k$
Output: $\hat{\mathbf{z}}$

Encoder $\mathsf{Enc}_{\mathbf{A}}$ is a d-private generic encoder. The gadget ensures that $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathbf{x} \oplus \mathbf{y}$.

1: **for** $i := 1$; $i \leq n$; $i{+}{+}$ **do**
2: $\quad \hat{\mathbf{z}}[i] = \hat{\mathbf{x}}[i] \oplus \hat{\mathbf{y}}[i]$
3: **end for**

We give the correctness and security proof of Gadget 2 in Theorem 3.

Theorem 3. *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^k$, $\mathsf{Enc}_{\mathbf{A}}$ *be a d-private generic encoder such that* $\mathbf{A} \in \mathbb{F}_q^{(k+m)\times n}$. *Then, Gadget 2 with inputs* $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x}), \hat{\mathbf{y}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{y})$ *and output* $\hat{\mathbf{z}}$ *ensures that* $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathbf{x} \oplus \mathbf{y}$, *and Gadget 2 is t-NI for any positive integer t such that* $t \leq n$.

Proof. We address the correctness first. Following the instructions of Gadget 2, we have $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{x}} \oplus \hat{\mathbf{y}}) = \mathsf{Dec}_{\mathbf{A}}\big([\mathbf{x}, \mathbf{r}]\mathbf{A} \oplus [\mathbf{y}, \mathbf{s}]\mathbf{A}\big)$, where $\mathbf{r} \in \mathbb{F}_q^m$ and $\mathbf{s} \in \mathbb{F}_q^m$. Finally, $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \mathsf{Dec}_{\mathbf{A}}(\mathbf{x} \oplus \mathbf{y}, \mathbf{r} \oplus \mathbf{s})\mathbf{A} = \mathbf{x} \oplus \mathbf{y}$.

We follow by proving the NI property. Since the operations are performed share-wisely, any probe (internal and output) in Gadget 2 has one of the following three forms: $\hat{\mathbf{x}}[i]$, $\hat{\mathbf{y}}[i]$ or $\hat{\mathbf{x}}[i] \oplus \hat{\mathbf{y}}[i]$ where $i \in [n]$. Let $\mathcal{I}$ be the indexes appearing in the t probes, by construction $|\mathcal{I}| \leq t$, and $\hat{\mathbf{x}}[\mathcal{I}]$ and $\hat{\mathbf{y}}[\mathcal{I}]$ are sufficient to simulate all the probes. Therefore, for any set $\mathcal{P}$ of $t \leq n$ probes we can build a simulator S such that:

$$\Pr\left[\mathsf{D}\left(\mathsf{G}_{\mathcal{P}}\left(\hat{\mathbf{x}}, \hat{\mathbf{y}}\right), \hat{\mathbf{x}}, \hat{\mathbf{y}}\right) = 1\right] = \Pr\left[\mathsf{D}\left(\mathsf{S}\left(\hat{\mathbf{x}}[\mathcal{I}], \hat{\mathbf{y}}[\mathcal{I}]\right), \hat{\mathbf{x}}, \hat{\mathbf{y}}\right) = 1\right], \text{ for any distinguisher } \mathsf{D}.$$

□

Combining the multiplication Gadget 1 of Section 4 and the addition Gadget 2 allows us to compute any function of the form:

$$(\mathbf{x}, \mathbf{y}) \mapsto \big(f(\mathbf{x}[1], \mathbf{y}[1]), \ldots, f(\mathbf{x}[k], \mathbf{y}[k])\big),$$

where $\mathbf{x} \in \mathbb{F}_q^k$, $\mathbf{y} \in \mathbb{F}_q^k$, and f a function from $\mathbb{F}_q \times \mathbb{F}_q$ to $\mathbb{F}_q$. Note that a gadget implementing a selection function (as defined on page 151) combined with the multiplication and addition gadgets is sufficient to compute any functions from $\mathbb{F}_q^{kl}$ to $\mathbb{F}_q^{kl'}$. The selection function can be implemented by the CodeL gadget defined next.

5.2 CodeL gadget

In this sub-section, we give a gadget to evaluate any $\mathrm{L}^{k,k}$ function on a codeword $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$. First, let us observe that in additive masking, any $\mathrm{L}^{k,k}$ function can be evaluated by applying it independently on all the shares with the same index (to produce each share of the outputs), and adding the constant $\mathbf{c}$ as needed on the output share with index 1. This is an application of the reduction of L functions to $\mathrm{L}^{k,k}$ functions and additions, since in additive masking, essentially $k = 1$ and a $\mathrm{L}^{1,1}$ function can be implemented share-by-share. However, code-based masking cannot exploit such a simple implementation (since $k \neq 1$ in the general case). In order to obtain secure masked $\mathrm{L}^{k,k}$ functions, we modify the construction of the multiplication gadget (Gadget 1). Although our secure construction is more complex than the trivial implementation of additive masking schemes, it is SNI (rather than NI for the trivial implementation of additive masking), and thus can save refresh gadgets in the composition (see the example of masked AES in Section 6.3). We describe the gadget for L functions in Gadget 3, in three parts as Gadget 1, and explain the reused parts and the difference from the multiplication gadget. A running example of Gadget 3 can be found in Example III.

Gadget 3 CodeL gadget

Input: Codeword $\hat{\mathbf{x}} \xleftarrow{\$} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ of $\mathbf{x} \in \mathbb{F}_q^k$.
Output: $\hat{\mathbf{z}} \in \mathbb{F}_q^n$

The gadget ensures that $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = f(\mathbf{x})$ for $f \in \mathrm{L}^{k,k}$
Encoder $\mathsf{Enc}_{\mathbf{A}}$ should be a d-private generic encoder. For $i \in [n]$, matrix $\mathbf{M}_i \in \mathbb{F}_q^{n\times(k+m)}$ is defined as $\mathbf{M}_i \stackrel{\text{def}}{=} (\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}})\mathbf{E}$. Codeword $\hat{\mathbf{1}} \in \mathbb{F}_q^n$ is defined as $\hat{\mathbf{1}} \stackrel{\text{def}}{=} (\mathbf{1},\mathbf{0})\mathbf{A}$.

Gadget 3 part A

Input: $\hat{\mathbf{x}} \in \mathbb{F}_q^n$
Output: $\mathbf{T} \in \mathbb{F}_q^{n\times(k+m)}$

This part reuses the Gadget 1 part A but setting $\hat{\mathbf{y}} = \hat{\mathbf{1}}$ *and* $\hat{\mathbf{R}}_1$ *to be a zero matrix.*

1: **for** $i = 1; i \le n; i{+}{+}$ **do**
2: **for** $j = 1; j \le n; j{+}{+}$ **do**
3: $\mathbf{S}[i,j] = \hat{\mathbf{x}}[i]\hat{\mathbf{1}}[j]$
4: **end for**
5: **end for**
6: **for** $i = 1; i \le n; i{+}{+}$ **do**
7: $\mathbf{T}[i,*] = \mathbf{S}[i,*] \times \mathbf{M}_i$
8: **end for**

Gadget 3 part B

Input: $\mathbf{T} \in \mathbb{F}_q^{n\times(k+m)}$
Output: $\mathbf{W} \in \mathbb{F}_q^{n\times n}$.

This part reuses the Gadget 1 part B but adding lines 1-4 to evaluate the f.

1: **for** $i = 2; i \le n; i{+}{+}$ **do**
2: $\mathbf{V}[i,*] = \big[f(\mathbf{T}[i,1{:}k]), \mathbf{0}\big]$
3: **end for**
4: $\mathbf{V}[1,*] = \big[f(\mathbf{T}[1,1{:}k]), \mathbf{0}\big] \oplus \big[(n-1)\mathbf{c}, \mathbf{0}\big]$
▷ $\mathbf{c}$ is a constant in $\mathbb{F}_q^k$ associated to f, and $(n-1)\mathbf{c} = \sum_{i=1}^{n-1} \mathbf{c}$
5: $\mathbf{W} = \mathbf{V}\mathbf{A}$

Gadget 3 part C

Input: $\mathbf{W} \in \mathbb{F}_q^{n\times n}$.
Output: $\hat{\mathbf{z}} \in \mathbb{F}_q^n$.

This part fully reuses the Gadget 1 part C.

1: Call Gadget 1 part C with inputs $\mathbf{W}$.

Example III

We use the particular generic encoder shown in Example I, and thus $k = 2, m = 4, n = 8$ and all the variables are in $\mathbb{F}_2$. The corresponding pre-computed matrices $\tilde{\mathbf{A}}$ and $\mathbf{M}_1, \ldots, \mathbf{M}_8$ can be found in Example II. We consider the secret inputs $\mathbf{x} = [1, 1]$ and the function:

$$f(\mathbf{a}) = \mathbf{aL}, \text{ for any } \mathbf{a} \in \mathbb{F}_2^2, \text{ where } \mathbf{L} = \begin{bmatrix} 11 \\ 10 \end{bmatrix}.$$

Without loss of generality, we assume that all the random bits are 1s. The input codeword is $\hat{\mathbf{x}} = [1, 1, 1, 1, 1, 1] \times \mathbf{A} = [0, 0, 0, 1, 0, 1, 1, 0]$, and the constant input codeword is $\hat{\mathbf{1}} = [0, 0, 1, 1, 1, 1] \times \mathbf{A} = [0, 0, 0, 1, 0, 1, 1, 0]$. Then, all the internal variables are:

$$\mathbf{R}_2 = \begin{bmatrix} 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \\ 000111 \end{bmatrix} \times \mathbf{A} = \begin{bmatrix} 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \\ 11101010 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 00000000 \\ 00000000 \\ 00000000 \\ 11111100 \\ 00000000 \\ 11111100 \\ 11111100 \\ 00000000 \end{bmatrix},$$

$$\mathbf{T} = \begin{bmatrix} 000000 \\ 000000 \\ 000000 \\ 010000 \\ 000000 \\ 000000 \\ 100000 \\ 000000 \end{bmatrix}, \mathbf{V} = \begin{bmatrix} 000000 \\ 000000 \\ 000000 \\ 100000 \\ 000000 \\ 000000 \\ 110000 \\ 000000 \end{bmatrix}, \mathbf{W} = \begin{bmatrix} 00000000 \\ 00000000 \\ 00000000 \\ 11110000 \\ 00000000 \\ 00000000 \\ 11111100 \\ 00000000 \end{bmatrix}, \mathbf{K} = \begin{bmatrix} 11101010 \\ 11101010 \\ 11101010 \\ 00011010 \\ 11101010 \\ 11101010 \\ 00010110 \\ 11101010 \end{bmatrix}.$$

At last, Gadget 3 outputs $\hat{\mathbf{z}} = [0, 0, 0, 0, 1, 1, 0, 0]$, which is a codeword of $\mathbf{xL} = [0, 1]$.

We explain the principle of Gadget 3 in the following. From the construction of the multiplication gadget, at one stage the computation handles an additive sharing of the result which is $\sum_{i=1}^{n} \mathbf{T}[i, 1\!:\!k] = \mathbf{x} \odot \mathbf{y}$ (the formal proof is given with Theorem 1). First note that we can replace $\mathbf{y}$ by $\mathbf{1} \in \mathbb{F}_q^k$ to obtain n share of which the sum gives $\mathbf{x}$. Then, we can apply the $\mathrm{L}^{k,k}$ function f on each share independently, and add the constant term accordingly. We show that these modifications still guarantee the correctness based on the proof related to the correctness of Gadget 1. Note that in this case, part A does not require any randomness, because $\hat{\mathbf{y}} = \mathbf{1}$ is public. We also prove that adding extra computations for the $\mathrm{L}^{k,k}$ function in part B does not impact the security.

In the next sub-sections, we prove the correctness of Gadget 3, and finally its SNI property, using the results of Section 4.

5.2.1 Correctness of the CodeL gadget

In the following, we prove the correctness of Gadget 3.

Theorem 4. *Let* $\mathbf{x} \in \mathbb{F}_q^k$*, and* $\mathsf{Enc}_{\mathbf{A}}$ *be a generic encoder such that* $\mathbf{A} \in \mathbb{F}_q^{(k+m)\times n}$*. Then, Gadget 3 with input* $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ *and output* $\hat{\mathbf{z}}$ *ensures that* $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = f(\mathbf{x})$ *for* $f \in \mathrm{L}^{k,k}$*.*

Proof. The goal is to prove that for any $\hat{\mathbf{x}} \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x})$ the gadget produces $\hat{\mathbf{z}}$ such that $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = f(\mathbf{x})$. Since the part C of the gadget is identical to the one of Gadget 1, we can use the same arguments as in the beginning of the proof of Theorem 1, giving an equation similar to Equation 3:

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \sum_{i=1}^{n} \mathbf{V}[i, 1:k].$$

Following the instructions of the part B it gives:

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = \sum_{i=1}^{n} f(\mathbf{T}[i, 1:k]) \oplus (n-1)\mathbf{c} = f\left(\sum_{i=1}^{n} \mathbf{T}[i, 1:k]\right),$$

where the last equality comes from the $\mathrm{L}^{k,k}$ functions property.

Then, Gadget 3 part A corresponds to the particular case of Gadget 1 part A where $\hat{\mathbf{y}}$ is the codeword of $\mathbf{1}$ with randomness $\mathbf{0}$ and the matrix $\mathbf{R}_1$ is null. Therefore the arguments in the last part of Theorem 1 apply on this sub-case. Accordingly we can rewrite Equation 4 in this case:

$$\sum_{i=1}^{n} \mathbf{T}[i, 1:k] = \sum_{i=1}^{n} \left(\left(\hat{\mathbf{x}}^{\mathrm{T}} \otimes \hat{\mathbf{1}}\right)[i, *] \times \mathbf{M}_i\right)[1:k] \oplus \sum_{i=1}^{n} \left(\mathbf{0} \times \mathbf{M}_i\right)[1:k].$$

By Proposition 6, we have $\left(\left(\hat{\mathbf{x}}^{\mathrm{T}} \otimes \hat{\mathbf{1}}\right)[i, *] \times \mathbf{M}_i\right)[1:k] = \mathbf{x}$ and $\sum_{i=1}^{n} \left(\mathbf{0} \times \mathbf{M}_i\right)[1:k] = \mathbf{0}$. Thus, we have $\sum_{i=1}^{n} \mathbf{T}[i, 1:k] = \mathbf{x}$. At last, we can conclude:

$$\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}) = f\left(\sum_{i=1}^{n} \mathbf{T}[i, 1:k]\right) = f(\mathbf{x}).$$

□

5.2.2 Security of the CodeL gadget

In the following part, we prove the security of Gadget 3, showing that instantiated with a d-private generic encoder it is t-SNI for valid input sharings (with $0 \leq t \leq d$). To do so, we begin by giving a lemma about simulation properties of the part A and B of the gadget. Then, Theorem 5 addresses the SNI property of Gadget 3. The proofs largely follow the ones of Section 4.4.

Lemma 9. *For Gadget 3 parts A and B, any t_{int} internal probes and any output in $\mathbf{W}[\mathcal{O}, *]$ such that $t_{int} + |\mathcal{O}| = t$ can be simulated with input shares indexed by $\mathcal{I}$ for $|\mathcal{I}| \leq t$.*

Proof. We show that any (internal or output) probe depend on only on index i corresponding to one share of $\hat{\mathbf{x}}$. Therefore, the set $\mathcal{I} \subseteq [n]$ defined as the union of the indexes appearing in the internal probes and in $\mathbf{W}[\mathcal{O}, *]$ is such that $|\mathcal{I}| \leq t$, and $\hat{\mathbf{x}}[\mathcal{I}]$ is sufficient to simulate all the probes.

First, we list all the probes depending on their type:

- The probes to the input shares: $\mathcal{P}_{input}$.
- The probes of $\hat{\mathbf{x}}[i]\hat{\mathbf{1}}[j]$ for any $i, j \in [n]$: $\mathcal{P}_x$.
- The probes to the computation from $\mathbf{S}$ to $\mathbf{T}$: $\mathcal{P}_{\mathbf{S}\to\mathbf{T}}$, where $\mathbf{S}[i, *]$ is multiplied by $\mathbf{M}_i$ to $\mathbf{T}[i, *]$. Thus, for each probe p in $\mathcal{P}_{\mathbf{S}\to\mathbf{T}}$, there exists a linear combination $f : \mathbb{F}_q^n \to \mathbb{F}_q$ and an index $i \in [n]$, such that $p = f(\mathbf{S}[i, *])$.

- The probes to the computation from $\mathbf{T}$ to $\mathbf{V}$: $\mathcal{P}_{\mathbf{T}\to\mathbf{V}}$, where f is applied to $\mathbf{T}[i,*]$ to obtain $\mathbf{V}[i,*]$, or a constant vector is added to $\mathbf{V}[1,*]$. Each of these probes are related to a single index $i \in [n]$.
- The probes to the computation from $\mathbf{V}$ to $\mathbf{W}$: $\mathcal{P}_{\mathbf{V}\to\mathbf{W}}$, where $\mathbf{V}[i,*]$ is multiplied by $\mathbf{A}$ to $\mathbf{W}[i,*]$. Thus, for each probe p in $\mathcal{P}_{\mathbf{V}\to\mathbf{W}}$, there exists a linear combination $g : \mathbb{F}_q^n \to \mathbb{F}_q$ and an index $i \in [n]$, such that $p = g(\mathbf{V}[i,*])$.
- The probes to the shares of output codewords $\mathbf{W}$: $\mathcal{P}_{out}$, whose corresponding row indexes are in a set $\mathcal{O}$.

To build the set $\mathcal{I}$, note that each internal probes, or all entries of each row of $\mathbf{W}$ depend on at most one index i which relates to the input codeword $\hat{\mathbf{x}}$. Then $\mathcal{I}$ is built as the union of these indexes, by construction $\mathcal{P}_{int} = \mathcal{P}_{input} + \mathcal{P}_x + \mathcal{P}_{\mathbf{S}\to\mathbf{T}} + \mathcal{P}_{\mathbf{T}\to\mathbf{V}} + \mathcal{P}_{\mathbf{V}\to\mathbf{W}}$ and $\mathcal{P}_{out} \in \mathbf{W}[\mathcal{O},*]$ ensuring $|\mathcal{P}_{int}| + |\mathcal{O}| \leq t$, therefore $|\mathcal{I}| \leq t$. Consequently we can define a simulator S which runs Gadget 3 parts A and B by feeding the values of $\hat{\mathbf{x}}[\mathcal{I}]$ and attributing random values to the other input shares, and outputs the probes. The probes only relate to $\hat{\mathbf{x}}[\mathcal{I}]$, enabling to conclude:

$$\Pr\left[\mathsf{D}\left(\mathsf{G}_{\mathcal{P}}\left(\hat{\mathbf{x}}\right), \hat{\mathbf{x}}\right) = 1\right] = \Pr\left[\mathsf{D}\left(\mathsf{S}\left(\hat{\mathbf{x}}[\mathcal{I}]\right), \hat{\mathbf{x}}\right) = 1\right], \text{ for any distinguisher } \mathsf{D}.$$

□

Theorem 5. *Let $\mathsf{Enc}_{\mathbf{A}}$ be a d-private generic encoder. Then, Gadget 3 is t-SNI for valid input sharings for any positive integer t such that $t \leq d$.*

Proof. Similarly to Theorem 2 we prove that any t_{out} output probes and internal probes $\mathcal{P}_{int}$ such that $|\mathcal{P}_{int}| + t_{out} \leq t$, can be simulated with $\hat{x}[\mathcal{I}]$ for $\mathcal{I} \subseteq [n], |\mathcal{I}| \leq |\mathcal{P}_{int}|$. To do so, we begin by dividing the probes into different subsets depending on which part of the gadget they are. Then, we successively use Lemmas 8, and 9 to obtain the final result.

The probes are divided into different subsets:

- The probes to the variables in part A and B, denoted as $\mathcal{P}_{AB}$.
- The probes to the internal variables in part C, denoted as $\mathcal{P}_{C,int}$.
- The probes to the output variables in part C, denoted as $\mathcal{P}_{C,out}$.

Since $\mathsf{Enc}_{\mathbf{A}}$ is a d-private generic encoder and $\mathcal{P}_{C,int} + \mathcal{P}_{C,out} \leq t \leq d$ we can apply Lemma 8: It gives that the $\mathcal{P}_{C,int}$ and $\mathcal{P}_{C,out}$ probes can be simulated by all the variables of $\mathbf{W}[\mathcal{I}_C,*]$ where $\mathcal{I}_C \subseteq [n]$ is such that $|\mathcal{I}_C| \leq |\mathcal{P}_{C,int}|$. Then, since $|\mathcal{P}_{AB}| + |\mathcal{I}_C| \leq |\mathcal{P}_{int}| \leq t \leq d$, Lemma 9 applies: It implies that the probes in $\mathcal{P}_{AB}$ and the variables of $\mathbf{T}[\mathcal{I}_C,*]$ can be simulated with input shares $\hat{\mathbf{x}}[\mathcal{I}]$ for $|\mathcal{I}| \leq |\mathcal{P}_{int}|$. In summary, $\mathcal{P}_{AB}$, $\mathcal{P}_{C,int}$ and $\mathcal{P}_{C,out}$ can be simulated with $\hat{\mathbf{x}}[\mathcal{I}]$ for $|\mathcal{I}| \leq |\mathcal{P}_{int}|$, finalizing the proof.

□

5.3 A generalized version: CodeLs gadget

The CodeLs gadget implements any L function (provided that the input and output size are multiples of k). Therefore, it operates on sensitive variables of multiple codewords. That is, the input of the gadget is $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_l$ and output is $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_{l'}$ such that $(\mathbf{z}_1, \ldots, \mathbf{z}_{l'}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_l)$ for any $f \in \mathrm{L}^{kl,kl'}$, addition gadgets and CodeL gadgets. That is, $f : \mathbb{F}_q^{lk} \to \mathbb{F}_q^{l'k}$ can be represented as $f(\mathbf{x}_1, \ldots, \mathbf{x}_l) = f^{(1)}(\mathbf{x}_1) \oplus \ldots \oplus f^{(l)}(\mathbf{x}_l)$, where $f^{(i)} : \mathbb{F}_q^k \to \mathbb{F}_q^{kl'}$ is a composition of l' $\mathrm{L}^{k,k}$ functions, and the addition can be implemented by the addition gadget. But, obviously this composition is not quite efficient. In the following, we give a more efficient construction.

Recall that the CodeL gadget first transforms the codeword of the generic encoder into an additive sharing (in Gadget 3, part A), then apply the $\mathrm{L}^{k,k}$ function (in Gadget 3, part B), and transform the result back to the codeword of generic encoder (in Gadget 3, part C). Similarly, in the case of L^2 gadget with multiple input codewords, we can first call Gadget 3, part A l times to transform $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_l$ into the corresponding additive sharing, and then apply the $\mathrm{L}^{kl,kl'}$ function in the same way as Gadget 3, part B. Finally, we call Gadget 3, part C l' times to transform the result into codewords $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_{l'}$. This L^2 gadget is presented in Gadget 4. We can see that Gadget 4 requires $l'nm$ random variables for l' times calls of Gadget 3, part C.

Gadget 4 CodeLs gadget

Input: Codewords $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_l \overset{\$}{=} \mathsf{Enc}_{\mathbf{A}}(\mathbf{x}_1), \ldots, \mathsf{Enc}_{\mathbf{A}}(\mathbf{x}_l)$ of $\mathbf{x}_1, \ldots, \mathbf{x}_l \in \mathbb{F}_q^k$.
Output: $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_{l'} \in \mathbb{F}_q^n$

The gadget ensures that $\mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}_1), \ldots, \mathsf{Dec}_{\mathbf{A}}(\hat{\mathbf{z}}_{l'}) = f(\mathbf{x}_1, \ldots, \mathbf{x}_{l'})$ for $f \in \mathrm{L}^{kl,kl'}$
Encoder $\mathsf{Enc}_{\mathbf{A}}$ should be a d-private generic encoder.

1: Call Gadget 3 part A l times for each inputs $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_l$, resulting in $\mathbf{T}_1, \ldots, \mathbf{T}_l$
2: $\mathbf{T} = [\mathbf{T}_1, \ldots, \mathbf{T}_l]$
3: Call Gadget 3 part B with input $\mathbf{T}$, resulting in $\mathbf{W}$
4: $\mathbf{W}_1, \ldots, \mathbf{W}_{l'} \leftarrow \mathbf{W}$ ▷ $\mathbf{W} = [\mathbf{W}_1, \ldots, \mathbf{W}_{l'}]$
5: Call Gadget 3 part C l' times for each inputs $\mathbf{W}_1, \ldots, \mathbf{W}_{l'}$, resulting in $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_{l'}$

Gadget 4 is t-SNI for $t \leq d$. The proof is quite similar to the one of Gadget 3. We give a sketch in the following. By Lemma 8 (since Gadget 3, part C is also Gadget 1, part C), we can see that any t_{int} internal variables and t_{out} output variables such that $t_{int} + t_{out} = t \leq d$ can be simulated with all input variables of $\mathbf{W}[\mathcal{I}, *]$ where $\mathcal{I} \subseteq [n]$, $|\mathcal{I}| \leq t_{int}$. Then, by Lemma 9, any t_{int} internal probes and any output in $\mathbf{W}[\mathcal{O}, *]$ such that $t_{int} + |\mathcal{O}| = t$ can be simulated with inputs shares indexed by $\mathcal{I}$ for $|\mathcal{I}| \leq t$. Therefore, any t_{int} internal variables and t_{out} output variables of Gadget 4 can be simulated with t_{int} inputs shares of each of $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_l$, giving that Gadget 4 is t-SNI for $t \leq d$.

6 Cost amortization and application to AES

At a first glance, the masked operations introduced in the last two sections have a high randomness and computational complexity. It is the case if we consider that only one sensitive variable is encoded into a codeword, and typically, we show in Appendix B that with a single sensitive variable Boolean masking is optimal in term of randomness. Nevertheless, when we take into account that cryptographic algorithms usually apply the function that is computed over multiple variables in parallel (*e.g.,* the 16 S-boxes of the AES block cipher), our scheme can lead to decent performance by packing multiple (*e.g.,* 16 for the AES) secret variables in a single codeword, a technique we call *amortization*. Then, a single application of one of our gadget can perform the same operation in parallel for all the secret variables. In this section, we first give an example of code that can be used for amortization while being d-probing secure. Then, we compare the cost of this amortized masking scheme with the state-of-the-art for two relevant cases: the multiplication (which is typically the most expensive part in masking schemes) and a full AES.

6.1 A generic encoder for amortization

We choose the matrices $\mathbf{G} = [\mathbf{I}, \mathbf{O}]$ and $\mathbf{H}$ to be a generator matrix of the Reed-Solomon codes [RS60], where $\mathbf{I}$ is an identity matrix in $\mathbb{F}_{2^8}^{k\times k}$ and $\mathbf{O}$ is a zero matrix in $\mathbb{F}_{2^8}^{k\times m}$. A typical construction of $\mathbf{H}$ is the transpose of a Vandermonde matrix over $\mathbb{F}_{2^8}$:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ a_1 & a_2 & \dots & a_n \\ a_1^2 & a_2^2 & \dots & a_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{m-1} & a_2^{m-1} & \dots & a_n^{m-1} \end{bmatrix},$$

where $a_1 \dots a_n$ are distinct values in $\mathbb{F}_{2^8}$ (therefore our construction is limited to $n \leq 2^8$) and $n = k + m$. Since $\mathbf{H}$ is the generating matrix of a Reed-Solomon code of length $n = k + m$ and dimension m, and by the property of Reed-Solomon code [RS60], $\mathcal{C}_{\mathbf{H}}$ is an MDS (maximum distance separable) code, and thus its minimal distance $d(\mathcal{C}_{\mathbf{H}}) = n - m + 1 = m + k - m + 1 = k + 1$. Hence, any non null element of this code has a Hamming weight of at least $k + 1$ whereas all the non null elements of the code generated be $\mathbf{G}$ have a Hamming weight of at most k. It proves that $\mathcal{C}_{\mathbf{G}} \cap \mathcal{C}_{\mathbf{H}} = \{\mathbf{0}\}$, while both $\mathbf{G}$ and $\mathbf{H}$ are full-rank. Therefore, $\mathbf{A} = [\mathbf{G}; \mathbf{H}]$ corresponds to a generic encoder.

As $\mathcal{C}_{\mathbf{H}}$ is an MDS code, its dual code $\mathcal{C}_{\mathbf{H}^\perp}$ is an MDS code as well. Thus, the dual distance $d(\mathcal{C}_{\mathbf{H}^\perp}) = n - (n - m) + 1 = m + 1$. By Proposition 5, we have that this generic encoder is a d-probing secure encoder, where $m + 1 - 1 \leq d \leq m$, giving $d = m$.

6.2 Cost of amortized multiplication

Cost metrics. We evaluate the cost of masked algorithms using two metrics: the amount of randomness needed and the number of bilinear multiplication (i.e., multiplications of two non-constant values), since those are typically the most expensive part of masked computations [BBP+16]. We note that this provides only estimates of performance, and since the actual cost largely depends on the platform, the actual benchmarks should compare real-world optimized implementations, which we leave to future works.

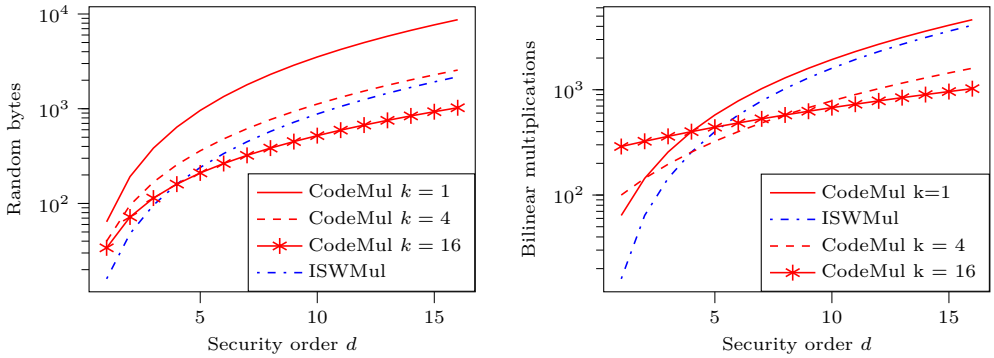
We compare the cost of one CodeMul execution, that computes k masked multiplications[7], to the cost of k (non-amortized) ISWMul [ISW03] masked multiplications.

The CodeMul gadget requires $2mn = 2d(d+k)$ random bytes and $n^2 = (d+k)^2$ bilinear multiplications. In constrast, k ISWMul gadgets require $kd(d+1)/2$ random bytes and $k(d+1)^2$ bilinear multiplications. Those costs are shown in Figure 4 for small values of d and k. With the increase of k, both the randomness and computational costs of CodeMul become smaller. More importantly, when $k \geq 8$ ($k \geq 2$, resp.), the randomness (computational, resp.) cost of CodeMul is smaller than that of ISWMul for sufficiently large d (and the gap increase with increasing d).

6.3 Our new construction of the masked AES

In the rest of this section, we assume $k|16$, and show an application of our masking to the AES block cipher. The AES-128 algorithm is performed on 16 variables in $\mathbb{F}_{2^8}$e. In every round, four types of transformations are performed: AddRoundKey, SubBytes, ShiftRows and MixColumns (see [DR02] for more details). ShiftRows and MixColumns are linear operations over $\mathbb{F}_{2^8}^{16}$, and a round key is XORed to the state in AddRoundKey. It becomes more complex for the SubBytes transformation, where a nonlinear function

[7] To avoid any confusion, we only consider the case that the masked multiplication is over $\mathbb{F}_{2^8}$ in this section.



(a) Randomness costs **(b)** Computational costs

Figure 4: Estimated costs of 16 multiplications in parallel.

$\mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ called S-box is computed over each of the 16 variables of the state. The S-box is a composition of an inverse in $\mathbb{F}_{2^8}$ and an affine transformation in $\mathbb{F}_2^8$.

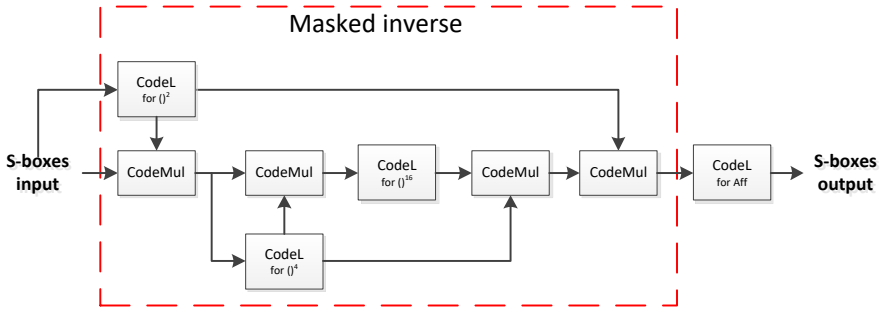
We firstly focus on the inverse. In [RP10], Rivain et al. propose to represent the inverse by a power function $x \to x^{254}$, which can be further decomposed into a quite efficient chain of several multiplications and squaring functions, and then can be performed by applying the multiplication gadget and L gadget. That is, the k variables are encoded into a codeword using the generic encoder, and then the k S-boxes are performed by applying Gadgets 1 and 3 (CodeMul and CodeL).

The squaring (as well as $(\cdot)^4$ and $(\cdot)^{16}$) is a typical instance of L function. The affine transformation Aff is also a L function: let us denote by $M : \mathbb{F}_{2^8} \to \mathbb{F}_2^8$ the AES change of representation from byte to bits. The affine transformation is then a map $\mathrm{Aff} : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$, $\mathrm{Aff}(x) = M^{-1}(M(x) \times \mathbf{A} \oplus \mathbf{b})$ for some constant $\mathbf{A} \in \mathbb{F}_2^{8\times 8}$ and $\mathbf{b} \in \mathbb{F}_2^8$. The map M is linear, therefore $\mathrm{Aff}(x \oplus y) = \mathrm{Aff}(x) \oplus \mathrm{Aff}(y) \oplus \mathrm{Aff}(0)$ and Aff is a L function and it can be implemented by using Gadget 3 (CodeL).

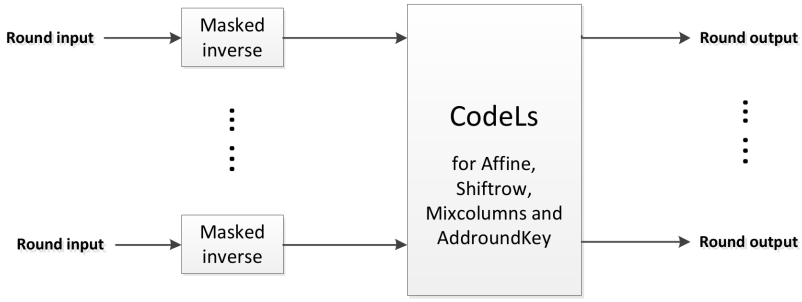
Figure 5a shows the masked S-boxes, where the input (resp., output) is a codeword of k input (resp., output) variables. Note that by Theorems 2 and 5, all the gadgets in Figure 5a are t-SNI for $t \leq d$. We can confirm from the figure that input codewords of each gadget comes from different gadgets, and thus by Lemma 4 the composed gadget is also t-SNI for $t \leq d$. Note also that the affine transformation, AddRoundKey, ShiftRows and MixColumns can be implemented together by instantiating Gadget 4 (CodeLs). Thus, in Figure 5b, we expand Figure 5a to a round function of AES consisting of SubBytes (16 S-boxes), ShiftRows MixColumns and AddRoundKey, and the full AES-128 can be built by considering 10 instances of the round function.[8]

It should be noted that our masked implementation of the AES does not require any refresh gadget. This is because the ingredients (i.e., CodeMul, CodeL and CodeLs) for the construction are all t-SNI. On the other hand, for the Boolean masking, building an SNI masked linear transformation usually requires refresh gadgets. In this respect, the relatively higher complexity of the "L function" of our scheme can somehow be mitigated by its stronger composability guarantees.

[8]The only concerns are the first round that start with an additional AddRoundKey, and the last round that does not contain MixColumns. For simplicity we estimate the cost of a full AES-128 by considering 10 instances of the round function.



(a) k masked AES S-boxes by applying the multiplication gadget and L gadget.



(b) Masked AES round function.

Figure 5: Masked AES S-boxes and round function by applying our masking scheme.

6.4 Estimated cost and comparison for masked AES

In the rest of this section, we estimate the cost of the masked AES and compare with benchmarks.

6.4.1 The benchmarks for comparison

For the benchmarks, we consider the constructions given in [BBP+16, Figure 6] and the bitsliced implementations in [BGR18]. The work in [BGR18] applies the ISW scheme to the bitslice technique and proposes a very efficient (software) implementation of masked AES that we call tight private circuits. The work in [BBP+16] reduces the random cost of masked multiplications from $O(d^2/2)$ to $O(d^2/4)$ (more precisely, from $d(d+1)/2$ to $\lfloor d^2/4 \rfloor + d$) and then proposed a masked AES where half the ISW multiplications can be replaced by randomness-reduced ones, while preserving the whole SNI security.

6.4.2 Randomness cost and comparison

Figure 5b contains $4(16/k)$ instances of CodeMul, $3(16/k)$ instances of CodeL and 1 instances of CodeLs. From Sections 4.2, 5.2 and 5.3, we know that:

- Each instance of CodeMul requires $2mn = 2d(d+k)$ random variables, consisting of $mn = d(d+k)$ random variables in the generation of $\mathbf{R}_1$ and another $m = d(d+k)n$ random variables over $\mathbb{F}_q$ in the generation of $\mathbf{R}_2$.
- Each instance of CodeL requires $nm = d(d+k)$ random variables for the generation of $\mathbf{R}_2$.
- Each instance of CodeLs requires $lnm = ld(d+k)$ random variables for the generation of l $\mathbf{R}_2$ matrices in l calls of CodeL. In our case, $l = 16/k$.

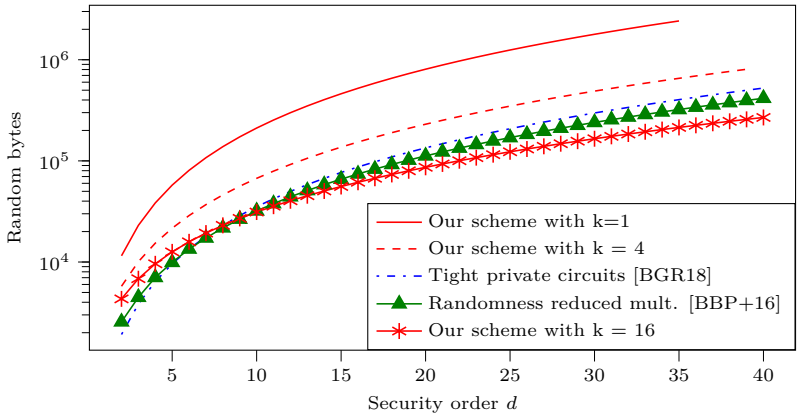


Figure 6: Randomness cost comparison with state-of-the-art based on masked AES-128.

In total, this implementation of a masked AES round function requires $4(16/k)2nm + 3(16/k)nm + 16nm/k = 192nm/k$ random bytes. Considering $d = m$, our masked AES requires $10(192nm/k) = 1920d(d+k)/k$ random bytes. In order to illustrate the cost amortization of our scheme, we consider $k = 1, 2, 4, 16$ for no cost amortization and encodings of 2, 4 and 16 variables (into one codeword) respectively in the comparison.

The tight private circuits require $32d(d+1)$ random bytes for 16 S-boxes in each round, giving a requirement of $320d(d+1)$ random bytes for a full AES-128. The reduced random cost of a masked multiplication in the work of [BBP+16] is $\lfloor d^2/4 \rfloor + d$, and the construction of one AES S-box they proposed in [BBP+16, Figure 6] contains the following parts:

1. Two randomness reduced multiplications with reduced randomness cost, and each of them requires at least $\lfloor d^2/4 \rfloor + d$ random bytes.
2. Two refresh gadgets. For a conservative comparison, we assume each refresh gadget requires at least d random bytes (it is even smaller than that in [BBD+18, Table 4]).
3. Two ISW multiplications, each of which requires at least $d(d+1)/2$ random bytes.

In Figure 6, we show the random costs of the above state-of-the-art implementations, and compare them with our new scheme for different values of k. We can see that, with the increase of k, the randomness cost becomes smaller. When $k = 16$, the randomness cost of our scheme is smaller than the state-of-the-art (when $d \geq 10$), and the gains increase with the security order. This gains can also be explained by comparing $1920d(d+k)/k$ (our scheme) and $320d(d+1)$ (tight private circuits): $1920d(d+k)/k < 320d(d+1)$ for large enough k and d.

6.4.3 Computational cost and comparison

We finally compare our scheme with the one in [BBP+16] which shares a similar construction with ours (both of them are based on the decomposition of the inverse in $\mathbb{F}_{2^8}$). As CodeL and CodeLs contain no bilinear multiplications, we only consider the masked multiplications. Both our implementation and the one in [BBP+16] contain 4 masked multiplications. However, the former one encodes the inputs of k AES S-boxes all together into a codeword, and performs k AES S-boxes only with 4 masked multiplications (*i.e.,* CodeMul). CodeMul contains n^2 bilinear multiplications for the calculation of $\mathbf{S}$, which gives that there are in total $64n^2/k$ bilinear multiplications for one masked round function. Meanwhile, the S-boxes are implemented separately in [BBP+16], and thus each S-box contains 4 masked multiplications, giving in total $64(d+1)^2$ bilinear multiplications (the use of either ISW

Table 1: Estimated computational cost[1] of AES implementations

		k multiplications in parallel	Full AES-128
Our scheme	k=1	$(d+1)^2$	$640(d+1)^2$
	k=2	$(d+2)^2$	$320(d+2)^2$
	k=4	$(d+4)^2$	$160(d+4)^2$
	k=16	$(d+16)^2$	$40(d+16)^2$
[BBP+16]		$k(d+1)^2$	$640(d+1)^2$

[1] Metric: the number of bilinear multiplications.

scheme or the NI multiplication does not impact the number of bilinear multiplications) for one round function. Thus, by $n = 16 + m = 16 + d$, the number of bilinear multiplication of our AES implementation (*i.e.,* $640(d+k)^2/k$) is smaller than that of [BBP+16] (*i.e.,* $640(d+1)^2$) if $k > 1$. We summarize the computational comparisons in Table 1.

We emphasize that, from the comparison above, we only claim that the computational cost of our scheme is asymptotically comparable to state-of-the-art, and leave the estimation based on real-world implementations as a future work.

7 Conclusion and future works

In this paper, we tackle the computational issue of code-based masking. We propose a more generalized masking encoder that covers all the well-known code-based masking encoders, based on which efficient masked operations are given. We show that our masked operations can lead to decent performance thanks to amortization. We provide an efficient masked AES implementation based on the new scheme with a limited complexity: in comparison with the state-of-the-art [BBP+16, BGR18, CS19], our implementation uses less randomness, and the computational complexity is comparable (for high security levels). Our scheme not only gives the first solution for the open challenge pointed in [PGS+17] (i.e., finding an efficient multiplication algorithms for DSM), but it also provides new directions for future works as listed next.

Analyzing the fault resistance. The encoders of code-based masking can offer the property of fault resistance. We have shown in Appendix C that the fault resistance of DSM codewords can be increased with the generic encoder. But Gadgets 1 and 3 cannot prevent the fault propagation issue [IPSW06, AMR+18]. It is an important problem to study how the computations/multiplications can resist to faults. In this respect, we foresee the value of the recent composable secure notions on both side-channel and fault attacks proposed in [DN19], and refer to improvement of the schemes for stronger fault resistance as a prospective further work.

Performance improvements with more specific codes. A dedicated choice of matrix $\mathbf{A}$ of the generic encoder can improve the computational complexity of Gadget 1, but also offers the possibility to decrease the randomness cost. For example, by Proposition 7, Gadget 1 ensures that any linear combination of the entries from $\mathbf{S}[i,*]$ gives no information about the secret, for any $i \in [n]$, and from the proof of it, this property comes from the similar property of matrix $\hat{\mathbf{R}}_1$: any linear combination of the entries from $\hat{\mathbf{R}}_1[i,*]$ is either uniformly distributed or null. This is usually too conservative, since one variable in computation of $\mathbf{S}[i,*] \times \mathbf{M}_i$ may only contain limited entries from $\mathbf{S}[i,*]$, which is determined by the matrix $\mathbf{M}_i \stackrel{\text{def}}{=} (\tilde{\mathbf{A}}[i,*] \otimes \tilde{\mathbf{A}})\mathbf{E}$. We can see that a sparse choice of $\tilde{\mathbf{A}}$ gives a sparse matrix $\mathbf{M}_i$, which leads to the fact that a variable in the computation of $\mathbf{S}[i,*] \times \mathbf{M}_i$ only contains limited entries from $\hat{\mathbf{R}}_1[i,*]$. In this case, the requirement of $\hat{\mathbf{R}}_1$ can be relaxed, saving a certain amount of randomness.

Acknowledgments

The authors thank Philippe Delsarte for his decisive help in proving the result of Appendix A. Gaëtan Cassiers and François-Xavier Standaert are respectively PhD Student and Senior Research Associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.). Pierrick Méaux is a beneficiary of a FSR Incoming Post-doctoral Fellowship. This work has been funded in parts by the European Union (EU) through the ERC project 724725 (acronym SWORD), the CHIST-ERA project SECODE and the H2020 project 731591 (acronym REASSURE).

References

[AMR+18] Anita Aghaie, Amir Moradi, Shahram Rasoolzadeh, Falk Schellenberg, and Tobias Schneider. Impeccable circuits. *IACR Cryptology ePrint Archive*, 2018:203, 2018.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 116–129, 2016.

[BBD+18] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Improved parallel mask refreshing algorithms: Generic solutions with parametrized non-interference & automated optimizations. *IACR Cryptology ePrint Archive*, 2018:505, 2018.

[BBP+16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien 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*, pages 616–648, 2016.

[BCC+14] Julien Bringer, Claude Carlet, Hervé Chabanne, Sylvain Guilley, and Houssem Maghrebi. Orthogonal direct sum masking - A smartcard friendly computation paradigm in a code, with builtin protection against side-channel and fault attacks. In *Information Security Theory and Practice. Securing the Internet of Things - 8th IFIP WG 11.2 International Workshop, WISTP 2014, Heraklion, Crete, Greece, June 30 - July 2, 2014. Proceedings*, pages 40–56, 2014.

[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 *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*, pages 535–566, 2017.

[BFG15] Josep Balasch, Sebastian Faust, and Benedikt 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*, pages 486–510, 2015.

[BFG+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier 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*, pages 724–754, 2017.

[BFGV12] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. Theory and practice of a leakage resilient masking scheme. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 758–775, 2012.

[BGR18] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. Tight private circuits: Achieving probing security with the least refreshing. 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*, pages 343–372, 2018.

[CCG+19] Wei Cheng, Claude Carlet, Kouassi Goli, Sylvain Guilley, and Jean-Luc Danger. Detecting faults in inner product masking scheme - IPM-FD: IPM with fault detection. *IACR Cryptology ePrint Archive*, 2019:919, 2019.

[CDG+14] Claude Carlet, Jean-Luc Danger, Sylvain Guilley, Houssem Maghrebi, and Emmanuel Prouff. Achieving side-channel high-order correlation immunity with leakage squeezing. *J. Cryptographic Engineering*, 4(2):107–121, 2014.

[CDGM14] Claude Carlet, Jean-Luc Danger, Sylvain Guilley, and Houssem Maghrebi. Leakage squeezing: Optimal implementation and security evaluation. *J. Mathematical Cryptology*, 8(3):249–295, 2014.

[CG16] Claude Carlet and Sylvain Guilley. Complementary dual codes for countermeasures to side-channel attacks. *Adv. in Math. of Comm.*, 10(1):131–150, 2016.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. pages 398–412, 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.

[CS19] Gaetan Cassiers and François-Xavier Standaert. Towards globally optimized masking: From low randomness to low noise rate or probe isolating multiplications with reduced randomness and security against horizontal attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):162–198, 2019.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 423–440, 2014.

[DN19] Siemen Dhooghe and Svetla Nikova. My gadget just cares for me - how NINA can prove security against combined attacks. *IACR Cryptology ePrint Archive*, 2019:615, 2019.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[FMPR10] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu 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*, pages 262–280, 2010.

[FPS17] Sebastian Faust, Clara Paglialonga, and Tobias Schneider. Amortizing randomness complexity in private circuits. 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*, pages 781–810, 2017.

[GM11] Louis Goubin and Ange 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*, pages 79–94, 2011.

[Gol95] Oded Goldreich. Three xor-lemmas - an exposition. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(56), 1995.

[GSF13] Vincent Grosso, François-Xavier Standaert, and Sebastian Faust. Masking vs. multiparty computation: How large is the gap for aes? In *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, pages 400–416, 2013.

[GSP13] Vincent Grosso, François-Xavier Standaert, and Emmanuel 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*, pages 33–43, 2013.

[IKL+13] Yuval Ishai, Eyal Kushilevitz, Xin Li, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and David Zuckerman. Robust pseudorandom generators. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 576–588, 2013.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David A. Wagner. Private circuits II: keeping secrets in tamperable circuits. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 308–327, 2006.

[IR90] Kenneth F. Ireland and Michael I. Rosen. *A classical introduction to modern number theory /*. Springer-Verlag„ New York :, 2nd ed. edition, c1990. "A revised and expanded version of Elements of number theory, published in 1972 by Bogden & Quigley."–Dorso port.

[ISW03] Yuval Ishai, Amit Sahai, and David 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*, pages 463–481, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 104–113, 1996.

[MGD11] Houssem Maghrebi, Sylvain Guilley, and Jean-Luc 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*, pages 208–223, 2011.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[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 *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, pages 123–141, 2017.

[PR11] Emmanuel Prouff and Thomas 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*, pages 63–78, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 142–159, 2013.

[RP10] Matthieu Rivain and Emmanuel 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*, pages 413–427, 2010.

[RS60] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[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 *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, pages 174–191, 2016.

A Equivalent distributions

In this part we give a proof of Lemma 1, dealing with joint distributions of probabilities and linear combinations of marginal distributions. The main motivation comes from the

use of different representation of the same distributions. On one side, probing security is usually considered in terms of joint distribution given by probes. On the other side, for linear codes considering linear combinations of codewords' coefficients is more appropriate, and it enables then to study linear combinations of marginal distributions. The joint distribution enables to derive any combination of the marginal distributions. However, the marginal distributions only are not sufficient to fully characterize the joint distribution. A well-chosen set of combinations of these distributions (rather than all) can be sufficient, but we focus on a simpler result implying all combinations since it fits better to the code notions.

The high-level idea of the proof is similar to the one given in [Gol95] Section 1 for example. The distance between probability distributions is studied by considering probability distributions as functions from $\mathbb{F}_2^n$ to $\mathbb{R}$, and more particularly using their expression in different orthogonal bases of this vector space. In our case, we show that from the linear combinations of the marginal distributions we can build a basis of the function from $\mathbb{F}_q^n$ to $\mathbb{R}$, which is therefore sufficient to determine all the equations given by the joint distribution. For sake of clarity, we articulate the proof in the following way. First, we recall how the joint distribution is connected to the canonical, or Kronecker basis of this vector space. Second, we recall results from Fourier analysis to give a basis of the function from K to the complex $\mathbb{C}$ for any finite Abelian group K. Then, we show how to give a basis of $\mathbb{F}_q^n$ to $\mathbb{R}$ using character's theory. Finally, we prove that the set of all linear combinations is connected to the set of all characters and is sufficient to give a basis.

First, let consider the functions $\delta_{\boldsymbol{\beta}} \mid \boldsymbol{\beta} \in \mathbb{F}_q^n$ from $\mathbb{F}_q^n$ to $\mathbb{R}$ defined as:

$$\delta_{\boldsymbol{\beta}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \boldsymbol{\beta}, \\ 0 & \text{otherwise.} \end{cases}$$

These q^n functions form a basis (often called canonical, or Kronecker basis) of the vector space of the functions from $\mathbb{F}_q^n$ to $\mathbb{R}$. The function $\delta_{\boldsymbol{\beta}}$ corresponds to the probability:

$$Pr\left[X_1 = \boldsymbol{\beta}_1, \text{ and } \cdots, \text{ and } X_n = \boldsymbol{\beta}_n\right],$$

defining the joint distribution of the random variables X_i. Another basis of the same vector space can be used to study the joint probability distribution, for example the Fourier basis.

In the following we recall some results presented in works on Boolean Fourier analysis *e.g.,* [O'D14], and characters of finite Abelian groups, to give bases of the functions from K to $\mathbb{C}$ (The field structure of $\mathbb{F}_q$ is not required for these results, the Abelian group one is sufficient). We begin by giving definitions and basic properties needed for the result, for more details and proofs of the properties we refer to textbooks of number theory such as [IR90].

Definition 15 (Group characters)**.** *Let K be an Abelian group, a function $f : K \to \mathbb{C} \setminus 0$ is called a character of K if it is a group homomorphism to $(\mathbb{C}, \times)$.*

Property 2 (Properties of characters of finite Abelian groups)**.** *Let K be a finite Abelian group, K has $|K|$ different characters, and the characters are either real-valued, or complex-valued conjugates 2 by 2.*

Proposition 8 (Characters and basis, *e.g.,* [O'D14] Proposition 8.55)**.** *Let K be a finite Abelian group, the set of all characters constitutes a basis of the functions from K to $\mathbb{C}$.*

We show that we can deduce a similar basis only $\mathbb{R}-$valued.

Proposition 9 (Characters and real basis)**.** *Let K be a finite Abelian group, let $\chi^{\mathbb{R}}$ be the set of the real-valued characters of K, and $\chi^{\mathbb{C}}$ the set of the other characters of K. The set excluding repetitions $S_\chi = \chi^{\mathbb{R}} \cup \bigcup_{\chi \in \chi^{\mathbb{C}}} \{\chi + \bar{\chi}, -i\chi + i\bar{\chi}\}$ is a basis of the functions from K to $\mathbb{R}$.*

Proof. Using Proposition 8 the characters of K constitute a basis of the function from K to $\mathbb{C}$. For any pair of conjugates characters in $\chi^{\mathbb{C}}$, χ and $\bar{\chi}$ can be written as $\chi(\mathbf{x}) = \phi(\mathbf{x}) + i\psi(\mathbf{x})$, and $\chi(\mathbf{x}) = \phi(\mathbf{x}) - i\psi(\mathbf{x})$, where i denotes the complex root of -1, ϕ and ψ are real-valued functions. The matrix $[1, 1; -i, i] \in \mathbb{C}^{2\times 2}$ is invertible, so the vector spaces generated by $\{\chi, \bar{\chi}\}$ and $\{\chi+\bar{\chi}, -i\chi+i\bar{\chi}\}$ are the same. The latter set is equal to $\{2\phi, 2\psi\}$ which are real-valued functions. Then, the vector space generated by S_χ is the same as the one generated by the set of the characters. As the $|K|$ vectors are $\mathbb{C}$-linearly independent, they are $\mathbb{R}$ linearly independent, therefore giving a basis of the functions from K to $\mathbb{R}$. □

Finally, we show that the linear combinations in $\mathbb{F}_q^n$ fully characterize the joint distribution by exhibiting the characters from these combinations. We first recall the definition of the trace of a field extension and some properties. Then we show how to build a basis relatively to $\mathbb{F}_q$ and finally $\mathbb{F}_q^n$.

Definition 16 (Field Trace)**.** *Let L be a field and M a finite extension, M can be viewed as a vector space over L. The multiplication by β an element of M, $m_\beta : M \to M$ given by $m_\beta(x) = \beta x$ is a $L-$linear transformation of this vector space into itself. The trace, $\mathsf{Tr}_{M/L}(\beta)$ (or $\mathsf{Tr}(\beta)$ when M and L are already identified) is the sum of the elements on the main diagonal of a matrix representing this linear transformation.*

Property 3 (Properties of finite field traces)**.** *Let L be a field and M a finite extension, the trace $\mathsf{Tr}_{M/L}$ is linear, for any $\beta \in M, \mathsf{Tr}(\beta) \in L$, and the $|M|$ functions defined as $m_\beta(x) = \mathsf{Tr}(\beta x)$ are all different.*

The trace of $\mathbb{F}_q$ seen as $\mathbb{F}_{p^k}$ where p is prime allows to exhibit the characters of $\mathbb{F}_q$ (seen as additive group):

Proposition 10 (Trace and characters of $\mathbb{F}_q$)**.** *Let $\mathbb{F}_q$ be the finite field of $q = p^k$ elements where p is prime, the set $\{f_\beta \mid f_\beta(x) = \omega^{\mathsf{Tr}(\beta x)}$, for $\beta \in \mathbb{F}_q\}$, where ω is a q-th primitive root of the unity and the result of the trace is considered in $\mathbb{Z}$ [9], is the set of the characters of $\mathbb{F}_q$.*

Proof. First, we prove that this functions are characters. Based on Definition 15 it consists in showing that each f_β is a group homomorphism to $(\mathbb{C}, \times)$. Using Property 3, the trace always lies in L which is $\mathbb{F}_p$ in this case, which can be considered in $\mathbb{Z}$ (as the set $[p]$). For any $\ell \in \mathbb{Z}$, ω^ℓ is a root of the unity so for all β and x in $\mathbb{F}_q$, $f_\beta(x) \in \mathbb{C} \setminus 0$. Now we show the group homomorphism, let $x, y \in \mathbb{F}_q$, $f_\beta(x+y) = \omega^{\mathsf{Tr}(\beta(x+y))}$. By Property 3 the linearity of the trace gives:

$$f_\beta(x+y) = \omega^{\mathsf{Tr}(\beta x)+\mathsf{Tr}(\beta y)} = \omega^{\mathsf{Tr}(\beta x)}\omega^{\mathsf{Tr}(\beta y)} = f_\beta(x) \cdot f_\beta(y).$$

The q functions f_β are the characters, and using Property 3 as the m_β functions are all different, the f_β are all different, giving exactly the set of characters. □

The basis for $\mathbb{F}_q^n$ is easy to obtain from the one of $\mathbb{F}_q$, for example using the fact that the $n-$fold product of a Fourier basis will give a Fourier basis (*e.g.,* [O'D14], Section 8).

Proposition 11 (Trace and characters of $\mathbb{F}_q^n$)**.** *Let n be a positive non-zero integer, $\mathbb{F}_q$ be the finite field of $q = p^k$ elements where p is prime, the set $\{f_{\boldsymbol{\beta}} \mid f_{\boldsymbol{\beta}}(\mathbf{x}) = \omega^{\sum_{i=1}^n \mathsf{Tr}(\boldsymbol{\beta}_i \mathbf{x}_i)}$, for $\boldsymbol{\beta} \in \mathbb{F}_q^n\}$, where ω is a q-th primitive root of the unity and the result of the trace is considered in $\mathbb{Z}$, is the set of the characters of $\mathbb{F}_q^n$.*

[9] the elements of $\mathbb{F}_p$ are identified to the elements between 0 and $p-1$.

Proof. For each $\boldsymbol{\beta} \in \mathbb{F}_q^n$ we get $f_{\boldsymbol{\beta}}(\mathbf{x}) = \prod_{i=1}^n f_{\boldsymbol{\beta}_i}(\mathbf{x}_i)$ using the notations of Proposition 10. Using this expression and the result of Proposition 10 is sufficient to conclude that the $f_{\boldsymbol{\beta}}$ are characters of $\mathbb{F}_q^n$. Then, we show that all these functions are different. For any $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ there is at least one index i such that $\boldsymbol{\beta}_i \neq \boldsymbol{\beta}'_i$. Let consider the set S_i of the inputs where all coefficients with index different from i are zero, (*i.e.,* of the shape $(0, \cdots, 0, x, 0, \cdots, 0)$ with $x \in \mathbb{F}_q$). For $\mathbf{x} \in S_i$, $f_{\boldsymbol{\beta}}(\mathbf{x}) = f_\beta(x)$, and $f_{\boldsymbol{\beta}'}(\mathbf{x}) = f'_\beta(x)$. As this two functions are different from Proposition 10's proof, $f_{\boldsymbol{\beta}} \neq f_{\boldsymbol{\beta}'}$, then we can conclude that this set of q^n functions is the set of characters.

□

To conclude, we rewrite here Lemma 1 and put the results of this section together to give the proof.

Lemma 1 (Distributions equivalence)**.** *Let X_i, $i \in [n]$, be random variables defined on a probability space where Ω consists in the elements of $\mathbb{F}_q$, the joint distribution is fully characterized by the set of distributions:*

$$\left\{ \sum_{i=1}^n \boldsymbol{\beta}_i X_i \mid \boldsymbol{\beta} \in \mathbb{F}_q^n \right\}.$$

Proof. The set of distributions are the linear combinations of $\mathbb{F}_q^n$, from which, using the field trace linearity, we obtain the characters of $\mathbb{F}_q^n$ as shown in Proposition 11. Using Properties 2, the characters of $\mathbb{F}_q^n$ constitute a basis of the functions from $\mathbb{F}_q^n$ to $\mathbb{C}$, that we can turn in a basis of the functions from $\mathbb{F}_q^n$ to $\mathbb{R}$ using Proposition 9. This basis is equivalent to the Kronecker basis, which corresponds to the expression of the joint distribution.

□

B Randomness optimality of Boolean masking and additive masking with single sensitive variable

Proposition 12. *Let* $\mathbf{A}$ *(resp.* $\mathbf{B}$*) be the matrix corresponding to the Boolean masking (resp. additive masking) with $m+1$ shares, then the d-privacy of* $\mathsf{Enc}_{\mathbf{A}}$ *(resp.* $\mathsf{Enc}_{\mathbf{B}}$ *) is optimal, equal to m.*

Proof. The Boolean masking encodes $\mathbf{x} \in \mathbb{F}_2^1$, and $\mathbf{r} \in \mathbb{F}_2^m$ uniformly random to:

$$\left(\mathbf{x} \oplus \sum_{i=1}^m \mathbf{r}[i], \mathbf{r}[1], \ldots, \mathbf{r}[m] \right).$$

With the definition of encoding described below, $\mathbf{A}$ corresponds to the first matrix of Figure 2. The matrix $\mathbf{G}$ is the first row $(1, \mathbf{0}^m)$ and the matrix $\mathbf{H}$ is the matrix $(\mathbf{1}^{m\mathrm{T}} | \mathbf{I}_m)$. $\mathcal{C}_{\mathbf{H}^\perp}$ is generated by $(1, \mathbf{1}^m)$, it is a code of parameters $[m+1, 1, m+1]$ which corresponds to the trivial code $\mathbf{1}^{m+1}$ the unique maximum distance separable code of length $m+1$ in this characteristic. Using Proposition 5 it gives $d(\mathcal{C}_{\mathbf{H}^\perp}) - 1 = m + 1 - 1 \leq d' \leq m$, so $d' = m$, which is optimal.

The additive masking encodes $\mathbf{x} \in \mathbb{F}_q^1$, and $\mathbf{r} \in \mathbb{F}_q^m$ uniformly random to:

$$\left(\mathbf{x} - \sum_{i=1}^m \mathbf{r}[i], \mathbf{r}[1], \ldots, \mathbf{r}[m] \right).$$

With the definition of encoding described below, $\mathbf{B}$ corresponds to the following matrix:

$$\mathbf{B} = \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline -1 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \cdots & 1 \end{array}\right).$$

The matrix $\mathbf{G}$ is the first row $(1, \mathbf{0}^m)$ and the matrix $\mathbf{H}$ is the matrix $(-\mathbf{1}^{m\mathrm{T}}|\mathbf{I}_m)$. $\mathcal{C}_{\mathbf{H}^\perp}$ is generated by $(1, \mathbf{1}^m)$, it is a code of parameters $[m+1, 1, m+1]$ as for Boolean masking. Using Proposition 5 it gives $d(\mathcal{C}_{\mathbf{H}^\perp}) - 1 = m + 1 - 1 \leq d' \leq m$, so $d' = m$, which is optimal. □

C Fault attacks: discussions and codeword resistance

Note that the set of possible values of a codeword can be a proper subset of $\mathbb{F}_q^n$. This property of the generic encoder provides redundancy as well as error detection for the codeword. To ease the analysis, the following two assumptions are required.

- The fault can only be injected to the input and output of each gadget (intermediates inside gadgets are tamper resistant). We require this assumption because a fault inside a gadget may impact several shares in the output codewords, masking the error undetectable. This issue is called the fault propagation [IPSW06, AMR+18], and tackling it needs other techniques (e.g., the "forced independence" in [AMR+18]) that are independent of the contributions of this paper.

- The circuit is always able to detect whether the current codeword is in the space of the code of the generic encoder. This can be realized by inserting checkpoints [IPSW06, AMR+18] into the private circuit.

With those assumptions, we can focus on showing fault resistance property of the generic encoder by quantifying the number of detectable faulted shares in a codeword. This property comes from the error detection/correction of the linear code and is given by the minimum distance of the code.

Proposition 13 (Encoding resistance to fault injection attacks)**.** *Let* $\mathsf{Enc}_{\mathbf{A}}$ *be a generic encoder, and let* d *denote* $d(\mathcal{C}_A)$ *then the associated encoding has fault resistance* $d - 1$*.*

Proof. Each encoding $\hat{\mathbf{x}}$ is an element of $\mathbb{F}_q^n$ such that there exists $\mathbf{x} \in \mathbb{F}_q^k$ and $\mathbf{r} \in \mathbb{F}_q^m$ such that $\hat{\mathbf{x}} = (\mathbf{x}, \mathbf{r})\mathbf{A}$, then $\hat{\mathbf{x}} \in \mathcal{C}_A$. As $\mathcal{C}_A$ has distance d, it enables to detect up to $d - 1$ faults and correct $\lfloor \frac{d-1}{2} \rfloor$ errors.

□

In comparison with the DSM masking [BCC+14, CG16, PGS+17], we can show that generic encoder permits to increase the fault resistance, keeping the same number of random elements m and probing security. It comes from the following proposition.

Proposition 14 (Increasing fault resistance)**.** *Let* $\mathsf{Enc}_{\mathbf{A}}$ *be a generic encoder, with* d*-privacy* $d_{\mathbf{A}}$ *and fault resistance* $f_{\mathbf{A}}$*. Let* $j \in [n]$*, the encoding relatively to the matrix* $\mathbf{A}' = [\mathbf{A}, \mathbf{A}[*, j]]$ *where* $\mathbf{A}[*, j]$ *is the* j^{th} *column of* $\mathbf{A}$*,* $\mathsf{Enc}_{\mathbf{A}'}$ *has* d*-privacy* $d_{\mathbf{A}'} = d_{\mathbf{A}}$ *and fault resistance* $f_{\mathbf{A}'} \geq f_{\mathbf{A}}$*.*

Proof. First, we argue that $\mathbf{A}'$ defines a generic encoder. Denoting $\mathbf{G}'$ the upper part of k rows and $\mathbf{H}'$ the lower part of m rows, as $\mathbf{A}' = [\mathbf{A}, \mathbf{A}[*, j]]$ we get $\mathbf{Rank}(\mathbf{A}') = \mathbf{Rank}(\mathbf{A})$. It gives $\mathbf{Rank}(\mathbf{G}') = k$, $\mathbf{Rank}(\mathbf{H}') = m$ and $\mathcal{C}_{\mathbf{G}'} \cap \mathcal{C}_{\mathbf{H}'} = \{\mathbf{0}\}$, defining a well-formed encoder.

Then, the fault resistance increase comes from Proposition 13 and the property that adding a column to a generator matrix can only increase the distance, $f_{\mathbf{A}'} = d(\mathcal{C}_{\mathbf{A}'}) \geq d(\mathcal{C}_{\mathbf{A}}) = f_{\mathbf{A}}$.

Finally, we have to prove that the d-privacy is the same relatively to $\mathbf{A}$ and $\mathbf{A}'$. $d_{\mathbf{A}'} \leq d_{\mathbf{A}}$ as for any element $\mathbf{w} \in \mathcal{C}_{\mathbf{H}^\perp}$ such that $\mathbf{G}\mathbf{w}^{\mathrm{T}} \neq \mathbf{0}^{\mathrm{T}}$ we have $(\mathbf{w}, 0) \in \mathcal{C}_{\mathbf{H}'^\perp}$ and $\mathbf{G}'(\mathbf{w}, 0)^{\mathrm{T}} = \mathbf{G}\mathbf{w}^{\mathrm{T}} \neq 0\mathbf{0}^{\mathrm{T}}$, with $\mathrm{HW}(\mathbf{w}, 0) = \mathrm{HW}(\mathbf{w})$. We show that $d_{\mathbf{A}'} \geq d_{\mathbf{A}}$ by contradiction. Assume that $d'_{\mathbf{A}} < d_{\mathbf{A}}$, then using Proposition 4 it implies that there exists $\mathbf{w}' \in \{\mathcal{C}_{\mathbf{H}'^\perp}\}_{d_{\mathbf{A}}}$ such that $\mathbf{G}'\mathbf{w}'^{\mathrm{T}} \neq \mathbf{0}^{\mathrm{T}}$. As $\mathbf{A}' = [\mathbf{A}, \mathbf{A}[*, j]]$ we have $\mathbf{A}'\mathbf{w}'^{\mathrm{T}} = \mathbf{A}\tilde{\mathbf{w}}^{\mathrm{T}}$ where $\tilde{\mathbf{w}} \in \mathbb{F}_q^n$ is defined as:

$$\tilde{\mathbf{w}}[i] = \begin{cases} \mathbf{w}'[i] & \text{if } i \in [n] \backslash \{j\}, \\ \mathbf{w}'[i] + 1 & \text{if } i = j. \end{cases}$$

As $\mathbf{w}' \in \{\mathcal{C}_{\mathbf{H}'^\perp}\}_{d_{\mathbf{A}}}$, $\mathbf{H}'\mathbf{w}'^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}} = \mathbf{H}\tilde{\mathbf{w}}^{\mathrm{T}}$, so $\tilde{\mathbf{w}} \in \mathcal{C}_{\mathbf{H}^\perp}$, and by construction $\mathrm{HW}(\tilde{\mathbf{w}}) \leq d_{\mathbf{A}}$, it gives $\tilde{\mathbf{w}} \in \{\mathcal{C}_{\mathbf{H}^\perp}\}_{d_{\mathbf{A}}}$. Therefore, $\mathbf{G}\tilde{\mathbf{w}}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}} = \mathbf{G}'\mathbf{w}'^{\mathrm{T}}$ which leads to a contradiction. It enables to conclude that $d_{\mathbf{A}'} \geq d_{\mathbf{A}}$, and finally $d_{\mathbf{A}'} = d_{\mathbf{A}}$, finishing the proof. □