Private Multiplication over Finite Fields

Sonia Belaïd[1], Fabrice Benhamouda[2], Alain Passelègue[3], Emmanuel Prouff[4,5], Adrian Thillard[6], and Damien Vergnaud[7,8]

[1] Thales Communications & Security, Gennevilliers, France
[2] IBM Research, Yorktown Heights, USA
[3] UCLA, Los Angeles, USA
[4] Safran Identity and Security, France
[5] Sorbonne Universités, UPMC Univ Paris 06, CNRS, INRIA, Laboratoire d'Informatique de Paris 6 (LIP6), Équipe PolSys, 4 place Jussieu, 75252 Paris, France
[6] ANSSI, Paris, France
[7] Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France
[8] INRIA

Abstract. The notion of privacy in the probing model, introduced by Ishai, Sahai, and Wagner in 2003, is nowadays frequently involved to assess the security of circuits manipulating sensitive information. However, provable security in this model still comes at the cost of a significant overhead both in terms of arithmetic complexity and randomness complexity. In this paper, we deal with this issue for circuits processing multiplication over finite fields. Our contributions are manifold. Extending the work of Belaïd, Benhamouda, Passelègue, Prouff, Thillard, and Vergnaud at Eurocrypt 2016, we introduce an algebraic characterization of the privacy for multiplication in any finite field and we propose a novel algebraic characterization for *non-interference* (a stronger security notion in this setting). Then, we present two generic constructions of multiplication circuits in finite fields that achieve non-interference in the probing model. Denoting by d the number of probes used by the adversary, the first proposal reduces the number of *bilinear* multiplications (i.e., of general multiplications of two non-constant values in the finite field) to only $2d + 1$ whereas the state-of-the-art was $O(d^2)$. The second proposal reduces the randomness complexity to d random elements in the underlying finite field, hence improving the $O(d \log d)$ randomness complexity achieved by Belaïd et al. in their paper. This construction is almost optimal since we also prove that $d/2$ is a lower bound. Eventually, we show that both algebraic constructions can always be instantiated in large enough finite fields. Furthermore, for the important cases $d \in \{2, 3\}$, we illustrate that they perform well in practice by presenting explicit realizations for finite fields of practical interest.

Keywords. Side-Channel Analysis, Probing Model, Bilinear Complexity, Randomness Complexity, Constructions, Lower Bounds, Probabilistic Method.

1 Introduction

While most symmetric cryptographic algorithms are now assumed to be secure against classical black-box attacks (e.g., when the attacker gets the knowledge of some inputs and/or outputs), their implementation can still be vulnerable to *side-channel attacks*. These attacks, revealed by Kocher in the 1990s [19], make additional use of the physical leakage of the underlying device (e.g., temperature, power consumption, execution time, ...) during the algorithm execution to recover the secret key.

These side-channel attacks are actually very powerful both against hardware and software implementations. In practice, keys from a classical block cipher can be recovered in a few minutes on many devices. Therefore, there is a huge need in efficient and secure countermeasures. Among the many ones proposed by the community, *masking* (a.k.a. *splitting* or *sharing*) [9, 16] is probably the most widely deployed. The main idea is to split each sensitive data, which depends both on the secret key and on known variables (e.g., inputs or outputs) into $d+1$ shares. The first d shares are generated uniformly at random and the last one is computed so that the combination of the $d+1$ shares with some group law $*$ is equal to the initial value. With this technique, the attacker actually needs the whole set of $d+1$ shares to learn any information on the initial value. Since each share's observation comes with noise, the higher the order d is, the more complex the attack is [9, 21].

In order to evaluate the security of masking schemes, the cryptographic community has made important efforts to define leakage models which properly reflect the reality of embedded devices. In 2003 [18], Ishai, Sahai, and Wagner introduced the *d-probing model* in which the attacker can get access to the exact values of at most d intermediate variables of its choice in the targeted implementation. While in practice, the attacker has access to the noisy values of all the manipulated variables, this model may still make sense, since recovering the exact value of d variables from their noisy observations is exponentially hard in the order d. Furthermore, it is widely used for its convenience to realize security proofs. Ten years later [21], Prouff, and Rivain extended a model initially introduced by Chari et al. [9], referred to as the *noisy leakage model*. This time, the model fits the reality of embedded devices since the attacker is assumed to get the noisy observations of all the intermediate variables of the implementation. However, because it requires the manipulation of noisy data (i.e., real values), this model is not convenient to make security proofs. Fortunately, Duc, Dziembowski, and Faust [13] exhibited a reduction from the noisy leakage model to the d-probing model, later improved in practice by Duc, Faust, and Standaert [14]. In other words, they proved that if an implementation is secure in the d-probing model, then it is also secure in the realistic noisy leakage model for specific number of shares, level of noise and circuit sizes. This sequence of works makes the d-probing model both realistic and convenient to make security proofs of masking schemes. An implementation secure in the d-probing model is said to satisfy the *d-privacy property* or equivalently to be *d-private* [18].

1.1 Our Problem

For the large majority of symmetric cryptographic algorithms which manipulate Boolean values, we naturally protect their implementation using Boolean masking for which $* = \oplus$. Each sensitive data is thus split into $d+1$ shares whose Boolean addition returns the initial value.[9]

In this context, the protection of linear functions is trivial since they just need to be applied independently to each share. However, the protection of non-linear functions is more complicated since the shares cannot be manipulated independently from each other. Concretely, additional randomness is required to randomize the computations which manipulate several shares of the same data. In particular, it is not trivial to evaluate the best way to build such countermeasures while minimizing the quantity of additional randomness as well as the number of operations.

The first proposal to perform a d-private multiplication over the finite field $\mathbb{F}_2$ was made by Ishai, Sahai, and Wagner in their seminal paper [18] (further referred to as ISW multiplication). They achieved d-privacy with $d(d+1)/2$ additional random bits and $(d+1)^2$ products over $\mathbb{F}_2$. Their multiplication then became the cornerstone of a sequence of works to build more complex d-private implementations [3, 10, 13, 14, 24]. Their proposal was described to securely compute a d-private multiplication over $\mathbb{F}_2$, but it can actually be transposed to secure a multiplication over any finite field $\mathbb{F}_q$ (e.g. [15, 24]) (in which case it requires $d(d+1)/2$ random field elements and $(d+1)^2$ products over $\mathbb{F}_q$). Secure implementation of multiplications over larger finite fields $\mathbb{F}_q$ (in particular for finite fields of characteristic 2), is of utmost practical interest to evaluate an S-box expressed as a polynomial over a such a finite field. For instance, it has been shown in [24] and [12] respectively that the implementation of the AES S-box (resp. the DES S-boxes) may be done with 4 (resp. 3) multiplications over $\mathbb{F}_{2^8}$ (resp. $\mathbb{F}_{2^6}$), instead of several dozens of multiplications over $\mathbb{F}_2$. However, with the order d growing up in practice for security reasons, this multiplication remains quite expensive. In particular, it consumes a large amount of randomness, which is generated by a physical source followed by a deterministic random bit generator, and it also requires a large number of multiplications, which are more expensive than linear operations.

That is why the community started to investigate more efficient d-private multiplications. Belaïd et al. [4] proposed a new d-private multiplication over the finite field $\mathbb{F}_2$ with twice as less randomness while preserving the number of multiplications. They also proved that any d-private multiplication over $\mathbb{F}_2$ requires at least d random bits and they proved a $O(d \log d)$ quasi-linear (non-constructive) upper bound for this randomness complexity. Most of their results can be readily generalized to d-private multiplication over any finite field $\mathbb{F}_{2^n}$

[9] An alternative is to apply so-called *threshold implementations* [20]. In [23], Reparaz et al. have shown that the latter implementations can be built from circuits that are made secure in the probing model. Thus, any improvement of the complexity of arithmetic circuits secure in the probing model may lead to complexity improvement for higher-order threshold implementations.

of characteristic 2 (except for the lower bound which holds only in $\mathbb{F}_2$). While their multiplication is d-private, it offers less security than the ISW one since it does not compose necessarily securely with other private circuits (see below for formal security definitions). It still can be used in symmetric algorithms to improve their performances: for instance, in the S-box of the block cipher AES defined over $\mathbb{F}_{2^8}$, three of the four multiplications can be replaced by theirs. Nevertheless, the proposal remains expensive and there is still a huge need in more efficient d-private multiplications.

1.2 Related Work

Other methods of encoding have been proposed in the literature. The *inner product masking*, proposed by Balasch et al. [2] encodes, over any finite field $\mathbb{F}_q$, the secret as a pair of vectors (L, R) such that the secret equals the inner product of L and R. In [1], this construction was enhanced by fixing a public value for L, hence allowing to achieve d-privacy using $d+1$ shares. The subsequent randomness and computation complexities for the multiplication are however still quadratic in d. Another approach, proposed by Prouff, and Roche [22] uses *polynomial masking*. Based on Shamir's secret sharing scheme, the secret is viewed as the constant coefficient of a certain polynomial, whose values when evaluated at some public points $(\alpha_i)_{i\leq d}$ constitute the shares.[10]. Though the complexity for the multiplication of the original proposal is cubic in d, Coron, Prouff, and Roche [11] achieved a complexity in $O(d^2 \log^4 d)$ for fields of characteristic 2. The recent work [17], which aims at achieving higher-order security in the presence of so-called *glitches*, is based on ISW multiplication and therefore requires $O(d^2)$ random values and field multiplications. It may moreover be noticed that this work directly benefits from the improvement proposed in [4] and in this paper.

1.3 Our Contributions

In this work, we aim to go further in the research of efficient d-private multiplications over finite fields $\mathbb{F}_q$ (where q is some prime power). Given two sharings $\boldsymbol{a} = (a_0, \ldots, a_d) \in \mathbb{F}_q^{d+1}$ and $\boldsymbol{b} = (b_0, \ldots, b_d) \in \mathbb{F}_q^{d+1}$, we aim to exhibit an output sharing $\boldsymbol{c} = (c_0, \ldots, c_d) \in \mathbb{F}_q^{d+1}$ such that

$$\sum_{i=0}^{d} c_i = \left(\sum_{i=0}^{d} a_i\right) \cdot \left(\sum_{i=0}^{d} b_i\right)$$

where the sum and product denote $\mathbb{F}_q$ operations. The computation of this sharing $\boldsymbol{c}$ should achieve the d-privacy (and actually will achieve a stronger security

[10] It may be remarked that the inner product masking with fixed public values for L is very close to polynomial masking, where R plays a similar role as the tuple of polynomial evaluations and where L plays a similar role as the reconstruction vector (deduced from the public values $(\alpha_i)_{i\leq d}$).

notion) with the use of a minimal number of random $\mathbb{F}_q$ elements and a minimal number of products in $\mathbb{F}_q$.

Extending the work of Belaïd et al. [4], we first present an algebraic characterization for privacy in the d-probing model for multiplication in any finite field. Contrary to the work done in [4] in which the authors limited themselves to multiplications based on the sum of shares' products, in this paper, we extend the possibilities by authorizing products of sums of shares.

As mentioned above, the scheme proposed by Belaïd et al. offers less security than the original ISW proposal since it does not compose necessarily securely with other private circuits. It is thus necessary to consider new security properties which strengthen the d-privacy. The introduction of such properties was made by Barthe, Belaïd, Dupressoir, Fouque, Grégoire, Strub, and Zucchini in [3], under the name of *non-interference*, *tight non-interference*, and *strong non-interference* (see Section 2 for formal definitions and for a comparison of these notions).

We then propose a novel algebraic characterization for *non-interference* in the d-probing model for multiplication in any finite field (and actually for any bivariate function over a finite field, as long as intermediate values are linear in the randomness and linear or bilinear in the inputs).

Theorem 3.5 ***(informal).*** A multiplication algorithm is non-interfering in the d-probing model if and only if there does not exist a set of $\ell \leq d$ intermediate results $\{p_1, \dots, p_\ell\}$ and a $\mathbb{F}_q$-linear combination of $\{p_1, \dots, p_\ell\}$ that can be written as

$$\boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} \; + \; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu} \; + \; \boldsymbol{\nu}^\intercal \cdot \boldsymbol{b} \; + \; \tau \;\; ,$$

where $\boldsymbol{M} \in \mathbb{F}_q^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{F}_q^{d+1}$, and $\tau \in \mathbb{F}_q$, and all the rows of the matrix $\big(\boldsymbol{M}|\boldsymbol{\mu}\big) \in \mathbb{F}_q^{(d+1)\times(d+2)}$ or the matrix $\big(\boldsymbol{M}^\intercal|\boldsymbol{\nu}\big) \in \mathbb{F}_q^{(d+1)\times(d+2)}$ are non-zero.

We then present two generic algebraic constructions of multiplication circuits in finite fields (based on this characterization) that achieve non-interference in the d-probing model. Both constructions are explicit and improve the complexity of previous proposals and their security is ensured as soon as some matrices satisfy some precise linear algebraic condition.

The first proposal (Algorithm 4) aims at reducing the number of *bilinear* multiplications (i.e., of general multiplications of two non-constant values in the finite field). The scheme requires only $2d + 1$ bilinear multiplications whereas all previous proposals need $O(d^2)$ such multiplications (at the cost of increasing the number of linear multiplications, i.e. multiplications by some constant). This leads to an important efficiency improvement in practice since bilinear multiplications over $\mathbb{F}_q$ cannot be tabulated for $q \geqslant 2^6$ (such a tabulation indeed requires $\log_2(q)q^2$ bits of ROM memory which is quickly too high for constrained devices), while multiplications by a constant can often be tabulated as long as $q \leqslant 2^{10}$ (such a tabulation indeed requires $\log_2(q)q$ bits of ROM memory). When the processing cannot be tabulated, it must be computed on-the-fly, which implies a non-negligible timing penalty: for instance a multiplication over $\mathbb{F}_{2^8}$ based on *log-alog* tables[11] would take around 40 CPU cycles on a classical AVR 8-bit architecture, while a direct lookup table access only takes 2 cycles (see [6] for more details about the different time/memory trade-offs for the multiplication processing). Additionally, our new scheme (Algorithm 4) achieves the strong non-interference security notion (Theorem 4.3) and composes therefore securely with other private circuits.

The goal of the second construction (Algorithm 5) is to reduce the randomness complexity; it needs only d random elements in the underlying finite field (improving the non-constructive upper bound $O(d \log d)$ proven in [4]). This constitutes an important improvement both from a theoretical and practical point of views since the generation of random values on a constrained device may be very time-consuming. Our second proposal achieves the non-interference security notion (which is stronger than the privacy notion achieved in [4]).

We show (using the probabilistic method) that both algebraic constructions can always be instantiated in large enough finite fields (Theorem 4.5 and Theorem 5.4). The second construction is almost optimal (for randomness complexity) since from our algebraic characterization, we can deduce the following lower bound on the randomness complexity:

Proposition 5.6 ***(informal).*** A non-interfering multiplication algorithm in the d-probing model uses more than $\lfloor (d-1)/2 \rfloor$ random elements in $\mathbb{F}_q$.

With our upper-bound, this proposition shows that the randomness complexity is therefore in $\Theta(d)$. These asymptotic results provide strong theoretical insights on the complexity of private multiplication. However, we also show that our constructions perform well in practice. In particular, for the important cases $d \in \{2, 3\}$, that are used in real-world implementations, we present explicit realizations of our constructions for finite fields of practical interest (and in particular for $\mathbb{F}_{2^8}$ used by the AES).

In terms of performance, we also compared the efficiency of our proposed constructions with the state of the art [4], for the practical masking orders $d \in \{2, 3\}$ and the finite field $\mathbb{F}_{2^8}$. The simulations have been done on a classical AVR 8-bit architecture; for different timing complexities of randomness generation[12] and of field multiplication, we measured the number of CPU cycles necessary to run the algorithms.

For $d = 2$ and a field multiplication taking 45 CPU cycles,[13] the proposal of [4] is more efficient, as soon as the generation of a random byte takes more than 7 cycles. In the event where this generation is shorter, our Algorithm 4

[11] More precisely, the non-zero field elements to multiplied are first represented as powers of a primitive element α such that $z = x \times y$ becomes $\alpha^c = \alpha^{a+b}$ with $(x, y, z) = (\alpha^a, \alpha^b, \alpha^c)$. The mappings $x \to \alpha^a$, $y \to \alpha^b$ and $\alpha^c \to z$ have been tabulated for efficiency reasons. The particular case $x = 0$ or $y = 0$ has been treated with care to not introduce timing dependency.

[12] For comparison/testing purpose, we did not call the device random generator but, instead, simulated the generation by a software code.

[13] This timing corresponds to a code written in assembly and involving log-alog look-up tables.

(Section 4.1) is better. Algorithm 5 (Section 5.1) is, in this case, always worse than the state of the art proposal, but it still outperforms Algorithm 4 as soon as the generation of random takes more than 12 cycles.

When the masking order is $d = 3$, Algorithm 4 is better when the random generation takes less than 16 cycles. Then, the algorithm of [4] is better when this number is lower than 60. Finally, Algorithm 5 outperforms both other constructions when the generation takes more than 60 cycles.

Similarly, we ran several simulations studying the impact of the complexity of the multiplication on our constructions. By fixing at 20 the number of cycles for the random generation, we observed that Algorithm 4 outperforms state of the art algorithms when the multiplication takes more than 6 cycles (resp. 93 cycles) for $d = 2$ (resp. $d = 3$). A comparison of the complexities of state of the art algorithms and our new proposals can be found in Table 1.

Table 1: Complexities of ISW, EC16, our new d-private compression gadget for multiplication and our specific gadgets at several orders

Complexities	ISW	EC16 [4]/small cases	Alg. 4	Alg. 5
Second-Order Masking ($d = 2$)				
sums	12	12 / 10	38	12
linear products	0	0 / 0	8	6
products	9	9 / 9	5	9
random scalars	3	3 / 2	9	2
Third-Order Masking ($d = 3$)				
sums	24	22 / 20	84	24
linear products	0	0 / 0	18	12
products	16	16 / 16	7	16
random scalars	6	5 / 4	21	3
Fourth-Order Masking ($d = 4$)				
sums	40	38 / 30	148	40
linear products	0	0 / 0	32	20
products	25	25 / 25	9	25
random scalars	10	8 / 5	38	4
d^{th}-Order Masking				
sums	$2d(d+1)$	$\begin{cases} d(7d+10)/4 & (d \text{ even}) \\ (7d+1)(d+1)/4 & (d \text{ odd}) \end{cases}$	$9d^2 + d$	$2d(d+1)$
linear products	0	0	$2d^2$	$d(d+1)$
products	$(d+1)^2$	$(d+1)^2$	$2d+1$	$(d+1)^2$
random scalars	$d(d+1)/2$	$\begin{cases} d^2/4 + d & (d \text{ even}) \\ (d^2-1)/4 + d & (d \text{ odd}) \end{cases}$	$2d^2 + \frac{d(d-1)}{2}$	d

2 Preliminaries

This section defines notation and basic notions that we use in this paper.

2.1 Notation

For a finite set S, we denote by $|S|$ its cardinality, and by $s \overset{\$}{\leftarrow} S$ the operation of picking up an element s of S uniformly at random. We denote by $\mathbb{F}_q$ the finite field with q elements. Vectors are denoted by lower case bold font letters, and matrices are denoted by bold font letters. All vectors are column vectors unless otherwise specified. The *image* of the linear map associated to a matrix $\boldsymbol{M}$ is denoted by $\mathrm{im}(\boldsymbol{M})$. For a vector $\boldsymbol{x}$, we denote by x_i its i-th coordinate and by $\mathsf{hw}(\boldsymbol{x})$ its Hamming weight (i.e., the number of its coordinates that are different from 0). When double indexing will be needed, we shall denote by $x_{i,j}$ the j-th coordinate of the vector $\boldsymbol{x}_i$. For vectors $\boldsymbol{x}_1, \dots, \boldsymbol{x}_t$ in $\mathbb{F}_q^n$, we denote $\langle \boldsymbol{x}_1, \dots, \boldsymbol{x}_t \rangle$ the vector space generated by the set $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_t\}$.

The *probability density function* associated to a discrete random variable X defined over S (e.g., $\mathbb{F}_q$) is the function which maps $x \in S$ to $\Pr[X = x]$. It is denoted by $\{X\}$ or by $\{X\}_r$ if there is a need to specify the randomness source r over which the *distribution* is considered.

Throughout the rest of this paper, when not specified, we consider the elements to belong to the finite field $\mathbb{F}_q$ for some prime power q. Some of our results require q to be larger than some lower bound that is then specified in the corresponding statements. We denote by $r \leftarrow \$$ the fact of sampling a fresh uniform element from $\mathbb{F}_q$ and assigning it to r.

2.2 Arithmetic Circuits and Privacy

An arithmetic circuit C is a directed acyclic graph whose vertices are input gates, output gates, addition gates, multiplication gates, or constant-scalar gates (over $\mathbb{F}_q$) and whose edges are wires carrying the inputs/outputs of the operations performed by the vertices. A constant-scalar gate is parameterized by a scalar $\gamma \in \mathbb{F}_q$, has fan-in 0, and outputs γ. A *randomized circuit* is a circuit augmented with random-scalar gates. A random-scalar gate is a gate with fan-in 0 that produces a random scalar in $\mathbb{F}_q$ and sends it along its output wire; the scalar is selected uniformly and independently of everything else afresh for each invocation of the circuit.

For a circuit C, we denote by $(y_1, y_2, \dots) \leftarrow C(x_1, x_2, \dots)$ the operation of running C on inputs $(x_1, x_2, \dots)$ and letting $(y_1, y_2, \dots)$ denote the outputs. Moreover, if C is *randomized*, we denote by $(y_1, y_2, \dots) \overset{\$}{\leftarrow} C(x_1, x_2, \dots)$ the operation of running C on inputs $(x_1, x_2, \dots)$ and with uniform fresh randomness. When we will need to specify this randomness we shall use the notation $(y_1, y_2, \dots) \leftarrow C(x_1, x_2, \dots; r)$. Eventually, for any subset P of wires in C, we denote by $C_P(x_1, x_2, \dots; r)$ (or $C_P(x_1, x_2, \dots)$ if the randomness is not specified) the list of values on the wires in P.

We hereafter give a formal definition of the notion of *gadget* used in prior works (e.g., [15]).

Definition 2.1 (gadget). *Let n, m be two positive integers and f be a function from $\mathbb{F}_q^n$ to $\mathbb{F}_q^m$. Let u, v be two positive integers. A (u,v)-*gadget for f *is an arithmetic (randomized) circuit C such that for every tuple $(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n)^\intercal \in (\mathbb{F}_q^u)^n$ and every randomness r, $(\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_m)^\intercal \leftarrow C(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n; r)$ satisfies*

$$\left(\sum_{j=1}^{v} y_{1,j}, \sum_{j=1}^{v} y_{2,j}, \dots, \sum_{j=1}^{v} y_{m,j}\right)^\intercal = f\left(\sum_{j=1}^{u} x_{1,j}, \sum_{j=1}^{u} x_{2,j}, \dots, \sum_{j=1}^{u} x_{n,j}\right) .$$

We usually define $x_i = \sum_{j=1}^{u} x_{i,j}$ and $y_i = \sum_{j=i}^{v} y_{i,j}$. The element $x_{i,j}$ (resp. $y_{i,j}$) is called the j-th *share* of x_i (resp. y_i).

Let us now define the notion of privacy for a gadget.

Definition 2.2 (d-private gadget). *Let n be a positive integer and let f be a function defined over $\mathbb{F}_q^n$. Let u and v be two positive integers. A (u,v)-gadget C for f is* d-private *if and only if for any set P of d wires in C, the distribution $\{C_P(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n; r) \mid \forall i \in \{1, \dots, n\},\ \sum_{j=1}^{u} x_{i,j} = x_i\}_{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n, r}$ is the same for every $(x_1, x_2, \dots, x_n)^\intercal \in \mathbb{F}_q^n$.*

Remark 2.3. In Definition 2.2, we recall that x_i denotes the i-th input of f, while $\boldsymbol{x}_i$ represents a sharing of x_i.

Remark 2.4. When there is no ambiguity, and for simplicity, the mention of the privacy order d will sometimes be omitted.

From now on, and to clarify the link with the *probing attack model* introduced in [18], the wires in a set P used to attack an implementation are referred as the *probes* and the corresponding values in $C_P(\dots; r)$ as the *intermediate results.* To simplify the descriptions, a probe p is sometimes used to directly denote the corresponding intermediate result. When the inputs w and the circuit C are clear from the context, the distribution $\{C_P(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n; r)\}_r$ is simplified to $\{(p)_{p \in P}\}$.

2.3 Compositional Security Notions

A (u,w)-gadget for the function $f \circ f'$ can be obviously built by composing a (v,w)-gadget of f and a (u,v)-gadget of f'. However, the composition $C \circ C'$ of two d-private gadgets C and C' is not necessarily itself d-private. For the latter to hold, gadget C' must satisfy a property which strengthens the privacy. The introduction of such a property has been made by Barthe et al. in [3]. Before recalling their definitions, we first need to introduce the notion of *t-simulatability.*

Definition 2.5 (t-simulatability). *Let u and v be two positive integers. Let C be a (u,v)-gadget for a function defined over $\mathbb{F}_q^n$. For some positive integers ℓ and t, a set $P = \{p_1, \dots, p_\ell\}$ of ℓ probes on C is* t-simulatable*, if there exist n sets*

I_1, I_2, ..., I_n of at most t indices in $\{1, \dots, u\}$ and a randomized function sim *defined from $(\mathbb{F}_q^t)^n$ to $\mathbb{F}_q^\ell$ such that for any fixed tuple $(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n) \in (\mathbb{F}_q^u)^n$, the distributions $\{p_1, \dots, p_\ell\}$ (which implicitly depends on $(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n)$, and the random values used by the gadget) and $\{\mathsf{sim}((x_{1,i})_{i\in I_1}, (x_{2,i})_{i\in I_2}, \dots, (x_{n,i})_{i\in I_n})\}$ are identical.*

Remark 2.6. The notation $\mathsf{sim}((x_{1,i})_{i\in I_1}, (x_{2,i})_{i\in I_2}, \dots, (x_{n,i})_{i\in I_n})$ will be simplified to $\mathsf{sim}(\boldsymbol{x}_{I_1}, \boldsymbol{x}_{I_2}, \dots, \boldsymbol{x}_{I_n})$. Moreover, depending on the context, we will sometimes call a t-simulatable set of probes, a set of probes which can be simulated with at most t shares of each of the n inputs of the gadget (which is an equivalent definition).

We now provide the notions of security that we will be using throughout the rest of the paper.

Definition 2.7 (d-non-interference). *A (u, v)-gadget C for a function f defined over $\mathbb{F}_q^n$ is* d-non-interfering (or d-NI) *if and only if every set of at most d probes can be simulated with at most d shares of each of its n inputs.*

Definition 2.8 (d-tight non-interference). *[3] A gadget C is* d-tight non-interfering (or d-TNI) *if and only if every set of $t \leq d$ probes can be simulated with at most t shares of each input.*

Definition 2.9 (d-strong non-interference). *A (u, v)-gadget C for a function f defined over $\mathbb{F}_q^n$ is* d-strong non-interfering (or d-SNI) *if and only if for every set P_1 of d_1 probes on internal wires (i.e., no output wires nor output shares) and every set P_2 of d_2 probes on output shares such that $d_1 + d_2 \leq d$, the set $P_1 \cup P_2$ of probes can be simulated by only d_1 shares of each of its n inputs.*

The d-SNI property is stronger than the d-NI property, which is itself stronger than the d-privacy property. The relations between all these notions are discussed in more details below.

2.4 Relations Between Compositional Security Notions

We recall that, from [3], if C is d-SNI (see Definition 2.9), then it is d-NI (see Definition 2.7); and if it is d-NI, then it is d-private. But a d-private gadget is not necessarily d-NI (see the counterexample given in [4, Appendix B]), and a d-NI gadget is not necessarily d-SNI (see for instance gadgets implementing SecMult in [24] or Algorithm 3 in [4]). Furthermore, in [4, Proposition 7.4], it is proven that d-NI and d-TNI are equivalent. These relations are depicted in Fig. 1.

From [3], the composition of a d-TNI (or d-NI) gadget with a d-SNI[14] is d-SNI, while the composition of d-TNI gadgets is not necessarily d-NI. This implies that d-SNI gadgets can be directly composed while maintaining the d-privacy property, whereas a d-SNI *refreshing* gadget (which randomizes the shares of its inputs using fresh random values) must sometimes be involved before the composition of d-NI gadgets.

[14] The inputs of the final gadget correspond to the inputs of the d-TNI one, while the outputs of the final gadget correspond to the outputs of the d-SNI one.

$$\begin{tikzcd}
\text{SNI} \arrow[r, bend left] & \text{TNI} \arrow[l, bend left, "/" marking] \arrow[r, bend left] & \text{NI} \arrow[l, bend left] \arrow[r, bend left] & \text{privacy} \arrow[l, bend left, "/" marking]
\end{tikzcd}$$

Fig. 1: Relations between privacy, NI, TNI, and SNI
(normal arrows are implications, strike out arrows are separations)

2.5 Case of Study

In this paper, we focus on the construction of efficient d-NI or d-SNI multiplication gadgets over $\mathbb{F}_q$ for any order d.

Definition 2.10 (multiplication gadget). *A* multiplication (u, v)-gadget *is a (u, v)-gadget C for the function $f : (a, b) \in \mathbb{F}_q^2 \mapsto a \cdot b \in \mathbb{F}_q$.*

Remark 2.11. When the sharing orders u and v will be clear from the context, the term (u, v) will be omitted.

In the sequel, the two inputs of a multiplication (u, v)-gadget C are denoted by a and b. Their respective sharings are thus denoted by $\boldsymbol{a} = (a_0, \ldots, a_{u-1})^\intercal \in \mathbb{F}_q^u$ and $\boldsymbol{b} = (b_0, \ldots, b_{u-1})^\intercal \in \mathbb{F}_q^u$. The output is denoted by c and its sharing is denoted by $\boldsymbol{c} = (c_0, \ldots, c_{v-1})^\intercal \in \mathbb{F}_q^v$. We also denote by $\boldsymbol{r} = (r_1, \ldots, r_R)^\intercal \in \mathbb{F}_q^R$ the vector of the random scalars that are involved in the gadget C. Thus, any intermediate result, a.k.a. probe, in the evaluation of C is a function of $a_0, \ldots, a_{u-1}, b_0, \ldots, b_{u-1}, r_1, \ldots, r_R$.

3 Algebraic Characterizations

This section aims at introducing algebraic characterizations for the privacy and the non-interference properties of a multiplication $(d + 1, v)$-gadget (for some positive integers d and v) over $\mathbb{F}_q$.

3.1 Bilinear Probes and Matrix Notation

For our algebraic characterizations, we focus on specific probes we call *bilinear probes.*

Definition 3.1. *Let C be a $(d + 1, v)$-gadget for a function f: $\mathbb{F}_q^2 \to \mathbb{F}_q$. A bilinear probe p is a probe on C (and thus an expression of $a_0, \ldots, a_d, b_0, \ldots, b_d$, $r_1, \ldots, r_R$), which is an affine functions of $a_i b_j$, a_i, b_j and r_k (for $0 \le i, j \le d$ and $1 \le k \le R$). In other words, a bilinear probe p can be written as:*

$$\boldsymbol{a}^\intercal \cdot \boldsymbol{M_p} \cdot \boldsymbol{b} \;+\; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu}_p \;+\; \boldsymbol{\nu}_p^\intercal \cdot \boldsymbol{b} \;+\; \boldsymbol{\sigma}_p^\intercal \cdot \boldsymbol{r} \;+\; \tau_p \;\; ,$$

where $M_p \in \mathbb{F}_q^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}_p \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\nu}_p \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\sigma}_p \in \mathbb{F}_q^R$, and $\tau_p \in \mathbb{F}_q$.

In the following sections we shall say that an expression $f(x_1, \ldots, x_n, r)$ functionally depends on the variable r if there exists $a_1, \ldots, a_n$ such that the function $r \mapsto f(a_1, \ldots, a_n, r)$ is not constant.

3.2 Algebraic Characterization for Privacy

We start by a simple extension of the algebraic characterization in [4] to any field $\mathbb{F}_q$ and to any function $f\colon \mathbb{F}_q^2 \to \mathbb{F}_q$ instead of just the multiplication function $f(a,b) = a \cdot b$ (however, please note that our characterization consider only bilinear probes). We consider the following condition:

Condition 3.1. *Let C be a $(d+1,v)$-gadget for a two-input function $f\colon \mathbb{F}_q^2 \to \mathbb{F}_q$. A set of bilinear probes $P = \{p_1,\dots,p_\ell\}$ on C satisfies Condition 3.1 if and only if there exists a vector $\boldsymbol{\lambda} \in \mathbb{F}_q^\ell$ such that the expression $\sum_{i=1}^{\ell} \lambda_i p_i$ can be written as*

$$\sum_{i=1}^{\ell} \lambda_i p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} \;+\; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu} \;+\; \boldsymbol{\nu}^\intercal \cdot \boldsymbol{b} \;+\; \tau\,,$$

where $M \in \mathbb{F}_q^{(d+1)\times(d+1)}$, $\boldsymbol{\mu} \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\nu} \in \mathbb{F}_q^{d+1}$, and $\tau \in \mathbb{F}_q$, and such that the all-one vector $\boldsymbol{u}_{d+1} = (1,\dots,1)^\intercal \in \mathbb{F}_q^{d+1}$ is in the affine space $\boldsymbol{\mu} + \mathrm{im}(\boldsymbol{M})$ or $\boldsymbol{\nu} + \mathrm{im}(\boldsymbol{M}^\intercal)$, where $\mathrm{im}(\boldsymbol{M})$ is the column space of $\boldsymbol{M}$.

We point out that, using notation of the above condition, for any set of bilinear probes $P = \{p_1,\dots,p_\ell\}$ on C and any $\boldsymbol{\lambda} \in \mathbb{F}_q^\ell$, the expression $\sum_{i=1}^{\ell} \lambda_i p_i$ can be written as

$$\sum_{i=1}^{\ell} \lambda_i p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M}_{\boldsymbol{\lambda}} \cdot \boldsymbol{b} \;+\; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu}_{\boldsymbol{\lambda}} \;+\; \boldsymbol{\nu}_{\boldsymbol{\lambda}}^\intercal \cdot \boldsymbol{b} \;+\; \boldsymbol{\sigma}_{\boldsymbol{\lambda}}^\intercal \cdot \boldsymbol{r} \;+\; \tau_{\boldsymbol{\lambda}}\,, \qquad (1)$$

where $M_{\boldsymbol{\lambda}} \in \mathbb{F}_q^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}_{\boldsymbol{\lambda}} \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\nu}_{\boldsymbol{\lambda}} \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\sigma}_{\boldsymbol{\lambda}} \in \mathbb{F}_q^{R}$, and $\tau_{\boldsymbol{\lambda}} \in \mathbb{F}_q$. Condition 3.1 is therefore equivalent to asking that there exists $\boldsymbol{\lambda} \in \mathbb{F}_q^\ell$ such that:

$$\boldsymbol{\sigma}_{\boldsymbol{\lambda}} = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{u}_{d+1} \in (\boldsymbol{\mu}_{\boldsymbol{\lambda}} + \mathrm{im}(\boldsymbol{M}_{\boldsymbol{\lambda}})) \cup (\boldsymbol{\nu}_{\boldsymbol{\lambda}} + \mathrm{im}(\boldsymbol{M}_{\boldsymbol{\lambda}}^\intercal))\ .$$

Theorem 3.2. *Let C be a $(d+1,v)$-gadget for a two-input function $f\colon \mathbb{F}_q^2 \to \mathbb{F}_q$. Let P be a set of bilinear probes on C. Then P satisfies Condition 3.1 if and only if there exist $a^{(0)}, b^{(0)}, a^{(1)}, b^{(1)} \in \mathbb{F}_q$, such that:*

$$\{(p)_{p\in P} \mid (a,b) = (a^{(0)}, b^{(0)})\} \neq \{(p)_{p\in P} \mid (a,b) = (a^{(1)}, b^{(1)})\}\ .$$

That is, the distribution $\{(p)_{p\in P}\}$ does depend on the value of (a,b).

The proof essentially uses the same ideas as the proof of Theorem A.1 of [4] and is detailed in the full version.

Remark 3.3. We do not restrict the size of the set P. Furthermore, the proof does not rely on the correctness property of C.

Corollary 3.4. *Let C be a $(d+1,v)$-gadget for a two-input function $f\colon \mathbb{F}_q^2 \to \mathbb{F}_q$. We suppose that any possible probe on C is bilinear. Then, C is d-private if and only if there does not exist any set P of d probes on C satisfying Condition 3.1.*

Proof. The proof is straightforward from Theorem 3.2. □

When $q = 2$ and when $f(a,b) = a \cdot b$, this corollary is actually equivalent to Theorem A.1 of [5]. Contrary to this former theorem, we only need to consider set of exactly d probes, as Condition 3.1 allows for discarding some probes (by choosing $\lambda_i = 0$). Furthermore, the gadget C has at least $2d + 2 \geq d$ possible probes: $a_0, \dots, a_d, b_0, \dots, b_d$. Thus, any set $\ell < d$ probes can be completed into a set of d probes.

3.3 Algebraic Characterization for Non-Interference

In this subsection, we introduce a novel algebraic characterization for Non-Interference (NI). We consider the following condition:

Condition 3.2. *Let C be a $(d+1, v)$-gadget for a two-input function $f\colon \mathbb{F}_q^2 \to \mathbb{F}_q$. A set of bilinear probes $P = \{p_1, \dots, p_\ell\}$ on C satisfies Condition 3.2 if and only if there exists $\boldsymbol{\lambda} \in \mathbb{F}_q^\ell$ such that the expression $\sum_{i=1}^{\ell} \lambda_i p_i$ can be written as*

$$\sum_{i=1}^{\ell} \lambda_i p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} \;+\; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu} \;+\; \boldsymbol{\nu}^\intercal \cdot \boldsymbol{b} \;+\; \tau \;\;,$$

where $\boldsymbol{M} \in \mathbb{F}_q^{(d+1)\times(d+1)}$, $\boldsymbol{\mu} \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\nu} \in \mathbb{F}_q^{d+1}$, and $\tau \in \mathbb{F}_q$, and such that all the rows of the matrix $\begin{pmatrix} \boldsymbol{M} & \boldsymbol{\mu} \end{pmatrix} \in \mathbb{F}_q^{(d+1)\times(d+2)}$ (which is the concatenation of the matrix $\boldsymbol{M}$ and the column vector μ) are non-zero or all the columns of the matrix $\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{\nu}^\intercal \end{pmatrix} \in \mathbb{F}_q^{(d+2)\times(d+1)}$ are non-zero.

We recall that, using notation of the above condition, for any set of bilinear probes $P = \{p_1, \dots, p_\ell\}$ on C and any $\boldsymbol{\lambda} \in \mathbb{F}_q^\ell$, the expression $\sum_{i=1}^{\ell} \lambda_i p_i$ can be written as in Equation (1). Therefore, Condition 3.2 is equivalent to asking that there exists $\boldsymbol{\lambda} \in \mathbb{F}_q^\ell$ such that $\sum_{i=1}^{\ell} \lambda_i p_i$ is functionally independent from any r_k $(0 \leq k \leq R)$ and functionally depends on every a_i $(0 \leq i \leq d)$ or on every b_j $(0 \leq j \leq d)$. This condition is therefore quite natural.

Theorem 3.5. *Let C be a $(d+1, v)$-gadget for a two-input function $f\colon \mathbb{F}_q^2 \to \mathbb{F}_q$. Let P be a set of bilinear probes on C. Then if P satisfies Condition 3.2, P is not d-simulatable. Furthermore, if P is not d-simulatable and $q > d+1$, then P satisfies Condition 3.2.*

We point out that the first part of the theorem does not require $q > d + 1$. As the second part is used for constructions while the first part is used for lower bounds, the restriction $q > d + 1$ is never an issue in our paper.

Proof. Let us start by proving the first direction, the second being more complex.
Direction 1: Left to right. By contrapositive, let us assume that there exists

a set $P = \{p_1, \dots, p_\ell\}$ of probes that satisfies Condition 3.2: that is, there exists $\lambda \in \mathbb{F}_q^\ell$ such that the sum $\sum_{i=1}^{\ell} \lambda_i p_i$ can be written as:

$$s = \sum_{i=1}^{\ell} \lambda_i p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu} + \boldsymbol{\nu}^\intercal \cdot \boldsymbol{b} \enspace ,$$

and, without loss of generality, such that all the rows of the matrix $M' = \begin{pmatrix}\boldsymbol{M} & \boldsymbol{\mu}\end{pmatrix} \in \mathbb{F}_q^{(d+1)\times(d+2)}$ are non-zero, meaning that s *does* functionally depend on every a_i but *does not* functionally depend on any r_i.

Then, assume that the set P can be simulated with at most d values of the a_i's, e.g., using only $a_1, \dots, a_d$, and let us further assume that the simulator has access to all the b_i's. That is, there exists a randomized function sim that takes as inputs $(a_1, \dots, a_d)$ and $(b_0, \dots, b_d)$ such that the distribution $\mathsf{sim}(a_1, \dots, a_d, b_0, \dots, b_d)$ is exactly the same as the distribution P.

Since s functionally depends on a_0, there exist specific values $a_1, \dots, a_d$, $b_0, \dots, b_d$ such that the function:

$$f_{(a_1,\dots,a_d,b_1,\dots,b_d)} \colon a_0 \mapsto \boldsymbol{a}^\intercal \cdot \boldsymbol{M} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu} + \boldsymbol{\nu}^\intercal \cdot \boldsymbol{b} \enspace ,$$

is not constant, by definition of s functionally depending on a_0.

Therefore, since $\mathsf{sim}(a_1, \dots, a_d, b_0, \dots, b_d)$ does not depend on a_0, it is impossible that it perfectly simulates the distribution P. This implies that one cannot simulate such a set of probes with at most d shares of each input and concludes the proof of this first direction.

Direction 2: Right to left. Let us now consider a set $P = \{p_1, \dots, p_\ell\}$ of bilinear probes that cannot be simulated with at most d shares of each input. Probes in P being bilinear, any linear combination of these probes can be written as

$$s_{\boldsymbol{\lambda}} = \sum_{i=1}^{\ell} \lambda_i p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M_\lambda} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu_\lambda} + \boldsymbol{\nu}_{\boldsymbol{\lambda}}^\intercal \cdot \boldsymbol{b} + \boldsymbol{\sigma}_{\boldsymbol{\lambda}}^\intercal \cdot \boldsymbol{r} \enspace ,$$

by definition. We want to show that, since P cannot be simulated with at most d shares of each input, there exists a particular λ such that $\boldsymbol{\sigma_\lambda} = \boldsymbol{0}$ and all the rows of $\begin{pmatrix}\boldsymbol{M_\lambda} & \boldsymbol{\mu_\lambda}\end{pmatrix}$ are non-zero or all the columns of $\begin{pmatrix}\boldsymbol{M_\lambda} \\ \boldsymbol{\nu}_{\boldsymbol{\lambda}}^\intercal\end{pmatrix}$ are non-zero.

Let us once again consider the matrix $\boldsymbol{S} \in \mathbb{F}_q^{\ell \times R}$ whose coefficients $s_{i,j}$ are defined as $s_{i,j} = \alpha$ if and only if p_i can be written as $\alpha r_j + z_i$ where z_i does not functionally depend on r_j. That is, if we write $p_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M_{p_i}} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu_i} + \boldsymbol{\nu_i}^\intercal \cdot \boldsymbol{b} + \boldsymbol{s_{p_i}}^\intercal \cdot \boldsymbol{r}$, the i-th row of $\boldsymbol{S}$ is $\boldsymbol{s_{p_i}}^\intercal$. We can permute the columns of $\boldsymbol{S}$ and the rows of $\boldsymbol{r}$ such that a row reduction on the matrix $\boldsymbol{S}$ yields a matrix of the form:

$$\boldsymbol{S'} = \begin{pmatrix}\boldsymbol{0}_{t,t} & \boldsymbol{0}_{t,\ell-t} \\ \boldsymbol{I}_t & \boldsymbol{S''}\end{pmatrix} \enspace .$$

Again, it is clear that since the distribution $\{p_1, \dots, p_\ell\}$ cannot be simulated with at most d shares of each input, we have $t > 0$. Indeed, otherwise we can simply simulate all probes by uniformly random values (and thus do not even need shares of the input). Let $\boldsymbol{N}$ be the invertible matrix in $\mathbb{F}_q^{\ell \times \ell}$ such that $\boldsymbol{N} \cdot \boldsymbol{S} = \boldsymbol{S'}$. We write $(p'_1, \dots, p'_\ell)^\intercal = \boldsymbol{N} \cdot \boldsymbol{p}$. Then, the distribution $\{p'_1, \dots, p'_\ell\}$ also cannot be simulated with at most d shares of each input. In addition, for $t < i \leq \ell$, p'_i does functionally depend on r_i and no other p'_j does functionally depend on r_j (due to the shape of $\boldsymbol{S'}$). Therefore, it is immediate that these probes can be simulated by setting them to uniformly random values, and thus the distribution $\{p'_1, \dots, p'_t\}$ also cannot be simulated with at most d shares of each input.

We remark that $(p'_1, \dots, p'_t)$ does not functionally depend on any random bit, due to the shape of $\boldsymbol{S'}$. Therefore, for each $1 \leq i \leq t$, we can write:

$$p'_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M'_i} \cdot \boldsymbol{b} + \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu'_i} + \boldsymbol{\nu'_i}^\intercal \cdot \boldsymbol{b} \ ,$$

for some matrices $\boldsymbol{M'_i} \in \mathbb{F}_q^{(d+1)\times(d+1)}$ and vectors $\boldsymbol{\mu'_i}, \boldsymbol{\nu'_i} \in \mathbb{F}_q^{d+1}$. Clearly, up to switching to roles of a and b, this implies that for any a_i, $i \in \{0, \dots, d\}$, there exists $j \in \{1, \dots, t\}$ such that p'_j functionally depends on a_i, otherwise one can simulate all the p'_i's with at most d shares of a, and then one can simulate $P = \{p_1, \dots, p_\ell\}$ as well.

We then just need to show that there exist $\boldsymbol{\lambda} \in \mathbb{F}_q^t$ such that $\sum_{i=1}^t \lambda_i \cdot p'_i$ satisfies Condition 3.2. This is actually immediate as soon as $q > d+1$: for $i = 0, \dots, d$ the set $\mathcal{H}_i = \{\boldsymbol{\lambda} \in \mathbb{F}_q^t \mid \sum_{i=1}^t \lambda_i p'_i \textit{ does not functionally depends on } a_i\}$ is a hyperplane, and thus we just need to prove that there exists $\boldsymbol{\lambda} \in \mathbb{F}_q^t \setminus \cup_{i=0}^d \mathcal{H}_i$, which is true as soon as $q > d+1$. This concludes the proof of Theorem 3.5. □

Remark 3.6. As for Theorem 3.2, we do not restrict the size of the set P in Theorem 3.5. Furthermore, the proof does not rely on the correctness property of C.

Corollary 3.7. *Let C be a $(d+1, v)$-gadget for a two-input function $f\colon \mathbb{F}_q^2 \to \mathbb{F}_q$. We suppose that any possible probe on C is bilinear. If $q > d+1$ and there does not exist any set P of d probes on C satisfying Condition 3.2, then C is d-NI. Furthermore, if C is d-NI, then there does not exist any set P of d probes on C satisfying Condition 3.2.*

Proof. The proof is straightforward from Theorem 3.5. □

4 Construction with a Linear Number of Bilinear Multiplications

Let us now show our generic d-SNI construction with a linear number of bilinear multiplications (i.e., multiplications by a value which is not constant), in the order d. The construction is in two steps. We first construct a d-NI multiplication $(d+1, 2d+1)$-gadget. In other words, our first construction outputs $2d+1$ shares instead of $d+1$. We then show how to compress these $2d+1$ shares into $d+1$ shares to get a d-SNI multiplication $(d+1, d+1)$-gadget, using the gadget

SharingCompress from the Appendix C.1 of [8], that we recall and prove to be d-SNI (while it was only implicitly proved d-NI in [8]).

We start by presenting the generic construction and its security proof. The first part of our construction uses a matrix $\boldsymbol{\gamma} \in \mathbb{F}_q^{d \times d}$ satisfying some conditions. That is why we then show that such a matrix exists for any d when q is large enough (but we only prove that q being exponential in $d \log d$ is sufficient) using the probabilistic method. We conclude by explicitly constructing matrices $\boldsymbol{\gamma}$ for small values of d.

4.1 Construction

Construction with $2d+1$ output shares. Let $\boldsymbol{\gamma} = (\gamma_{i,j})_{1 \le i,j \le d} \in \mathbb{F}_q^{d \times d}$ be a constant matrix and let $\boldsymbol{\delta} \in \mathbb{F}_q^{d \times d}$ be the matrix defined by $\delta_{i,j} = 1 - \gamma_{j,i}$.

The main idea of our construction with $2d+1$ output shares is to remark that:

$$a \cdot b = \left(a_0 + \sum_{i=1}^{d} (r_i + a_i)\right) \cdot \left(b_0 + \sum_{i=1}^{d} (s_i + b_i)\right) \\ - \sum_{i=1}^{d} r_i \cdot \left(b_0 + \sum_{j=1}^{d} (\delta_{i,j} s_j + b_j)\right) - \sum_{i=1}^{d} s_i \cdot \left(a_0 + \sum_{j=1}^{d} (\gamma_{i,j} r_j + a_j)\right)$$

if $a = \sum_{i=0}^{d} a_i$ and $b = \sum_{j=0}^{d} b_j$. On the right-hand side of the above equation there are only $2d+1$ bilinear multiplications.

We can then construct a multiplication $(d+1, 2d+1)$-gadget which outputs the following $2d+1$ shares (the computation is performed with the usual priorities: parenthesis first, then products, then from left to right):

- $c_0 = \left(a_0 + \sum_{i=1}^{d}(r_i + a_i)\right) \cdot \left(b_0 + \sum_{i=1}^{d}(s_i + b_i)\right)$;
- $c_i = -r_i \cdot \left(b_0 + \sum_{j=1}^{d}(\delta_{i,j} s_j + b_j)\right)$, for $i = 1, \dots, d$;
- $c_{i+d} = -s_i \cdot \left(a_0 + \sum_{j=1}^{d}(\gamma_{i,j} r_j + a_j)\right)$, for $i = 1, \dots, d$.

The corresponding gadget is given in Algorithm 1 and is clearly correct.

However, the latter gadget has two issues. First, it outputs $2d+1$ shares instead of $d+1$. Second, it is obviously not secure for every matrix $\boldsymbol{\gamma}$. For example, if $\boldsymbol{\gamma}$ is a matrix of zeros or the identity matrix, the gadget is clearly not d-private, let alone d-NI or d-SNI. Actually, it is not even clear that there exists a matrix $\boldsymbol{\gamma}$ for which the gadget is private. Let us now deal with these two issues.

From $2d+1$ output shares to $d+1$. For the first issue, we use the gadget SharingCompress from the Appendix C.1 of [8] to compress the shares $c_0, \dots, c_{2d}$ into $d+1$ shares. We recall this gadget in Algorithm 2.

Algorithm 1 ExtendedMult

```
Require: $\boldsymbol{a} = (a_0, \dots, a_d), \boldsymbol{b} = (b_0, \dots, b_d)$
Ensure: $\boldsymbol{c} = (c_0, \dots, c_{2d})$ such that $\sum_{i=0}^{2d} c_i = (\sum_{i=0}^{d} a_i) \cdot (\sum_{i=0}^{d} b_i)$
  $x \leftarrow a_0$;  $y \leftarrow b_0$
  for $i = 1$ to $d$ do
      $c_i \leftarrow b_0$
      $c_{i+d} \leftarrow a_0$
      for $j = 1$ to $d$ do
          $s_j \leftarrow \$$
          $r_j \leftarrow \$$
          $t \leftarrow \delta_{i,j} s_j + b_j$
          $c_i \leftarrow c_i + t$
          $y \leftarrow y + (s_j + b_j)$
          $t \leftarrow \gamma_{i,j} r_j + a_j$
          $c_{i+d} \leftarrow c_{i+d} + t$
          $x \leftarrow x + (r_j + a_j)$
      $c_i \leftarrow -r_i \cdot c_i$
      $c_{i+d} \leftarrow -s_i \cdot c_{i+d}$
  $c_0 \leftarrow x \cdot y$
  return $(c_0, c_1, \dots c_{2d})$
```

Proposition 4.1. *The gadget SharingCompress*$[k : \ell]$ *depicted in Algorithm 2 is* $(\ell - 1)$*-SNI.*

This proof is given in the full version. From this proposition, we deduce that the instance SharingCompress$[2d + 1 : d + 1]$ that we need is d-SNI.

Finally, the full gadget with a linear number of bilinear multiplications is depicted in Algorithm 4. It essentially calls Algorithm 3 which handles the special case where the number of input shares is twice the number of output shares.

As we are composing the gadget SharingCompress with our multiplication gadget above, we need to prove that the former gadget satisfies a security property which behaves well with composition. In [8], only privacy is proven which does not behave well with composition. That is why we prove instead the following proposition in the full version.

Conditions on $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$. As mentioned before, the construction is completely insecure for some matrices $\boldsymbol{\gamma}$, such as the matrix of zeros. Let us now exhibit necessary conditions for the scheme to be d-NI.

The probes involving only the a_i's and the r_i's[15] are of the following forms:

- a_i, r_i, $r_i + a_i$, $\gamma_{j,i} r_i$, $\gamma_{j,i} r_i + a_i$, (for $0 \leq i \leq d$ and $1 \leq j \leq d$)
- $a_0 + \sum_{i=1}^{k} (r_i + a_i)$ (for $1 \leq k \leq d$),
- $a_0 + \sum_{i=1}^{k} (\gamma_{j,i} r_i + a_i)$ (for $1 \leq j \leq d$ and $1 \leq k \leq d$).

[15] By probes involving only the a_i's and the r_i's, we mean probes that do not functionally depend on any b_i nor any s_i.

Algorithm 2 SharingCompress[$k : \ell$] from [8, Appendix C.1]

```
Require: $k$-sharing $(x_i)_{1 \le i \le k}$
Ensure: $\ell$-sharing $(y_i)_{1 \le i \le \ell}$ such that $\sum_{i=1}^{\ell} y_i = \sum_{i=1}^{k} x_i$
  $K \leftarrow \ell \lceil k/\ell \rceil$
  for $j = k+1$ to $K$ do
      $x_j \leftarrow 0$
  for $j = 1$ to $\ell$ do
      $y_j \leftarrow x_j$
  for $j = 1$ to $\frac{K-\ell}{\ell}$ do
      $(y_1, \ldots, y_\ell) \leftarrow$ SharingCompress$[2\ell : \ell](y_1, \ldots y_\ell, x_{j\ell+1}, \ldots, x_{j\ell+\ell})$
  return $(y_1, \ldots, y_\ell)$
```

Algorithm 3 SharingCompress[$2d : d$] from [8, Appendix C.1]

```
Require: $2d$-sharing $(x_i)_{1 \le i \le 2d}$
Ensure: $d$-sharing $(y_i)_{1 \le i \le d}$ such that $\sum_{i=1}^{d} y_i = \sum_{i=1}^{2d} x_i$
  for $i = 1$ to $d$ do
      for $j = i+1$ to $d$ do
          $r_{i,j} \leftarrow \$$
  for $i = 1$ to $d$ do
      $v_i \leftarrow 0$
  for $i = 1$ to $d$ do
      for $j = 1$ to $i-1$ do
          $v_i \leftarrow v_i - r_{j,i}$
      for $j = i+1$ to $d$ do
          $v_i \leftarrow v_i + r_{i,j}$
  for $i = 1$ to $d$ do
      $y_i \leftarrow x_i + v_i$
      $y_i \leftarrow y_i + x_{i+d}$
  return $(y_1, \ldots y_d)$
```

Algorithm 4 Construction with a Linear Number of Bilinear Multiplications

```
Require: $\boldsymbol{a} = (a_0, \ldots, a_d), \boldsymbol{b} = (b_0, \ldots, b_d)$
Ensure: $\boldsymbol{c'} = (c'_0, \ldots, c'_d)$ such that $\sum_{i=0}^{d} c'_i = (\sum_{i=0}^{d} a_i) \cdot (\sum_{i=0}^{d} b_i)$
  $(c_0, \ldots c_{2d}) \leftarrow$ ExtendedMult$(\boldsymbol{a}, \boldsymbol{b})$
  $(c'_0, \ldots c'_d) \leftarrow$ SharingCompress$[2d+1 : d+1](c_0, \ldots c_{2d})$
  return $(c'_0, c'_1, \ldots c'_d)$
```

Thanks to Theorem 3.5, a necessary condition for d-NI is that there is no linear combination of at most d of these expressions, which do not functionally depend on any r_i but which does depend on all the a_i's.

The probes involving only the b_i's and the s_i's are similar except that a_i, r_i, and $\gamma_{j,i}$ are replaced by b_i, s_i, $\delta_{j,i}$ respectively. A similar necessary condition can be deduced from Theorem 3.5.

Formally, let us introduce a first necessary condition on the matrix $\boldsymbol{\gamma}$.

Condition 4.1. *Let $\ell = (2d+4)\cdot d+1$. Let $\boldsymbol{I}_d \in \mathbb{F}_q^{d\times d}$ be the identity matrix, $\boldsymbol{0}_{m\times n} \in \mathbb{F}_q^{m\times n}$ be a matrix of zeros (when $n=1$, $\boldsymbol{0}_{m\times n}$ is also written $\boldsymbol{0}_m$), $\boldsymbol{1}_{m\times n} \in \mathbb{F}_q^{m\times n}$ be a matrix of ones, $\boldsymbol{D}_{\boldsymbol{\gamma},j} \in \mathbb{F}_q^{d\times d}$ be the diagonal matrix such that $D_{\boldsymbol{\gamma},j,i,i} = \gamma_{j,i}$, $\boldsymbol{T}_d \in \mathbb{F}_q^{d\times d}$ be the upper-triangular matrix with just ones, and $\boldsymbol{T}_{\boldsymbol{\gamma},j} \in \mathbb{F}_q^{d\times d}$ be the upper-triangular matrix for which $T_{\boldsymbol{\gamma},j,i,k} = \gamma_{j,i}$ for $i \le k$. In other words, we have:*

$$\boldsymbol{I}_d = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix} \qquad \boldsymbol{D}_{\boldsymbol{\gamma},j} = \begin{pmatrix} \gamma_{j,1} & 0 & \dots & 0 \\ 0 & \gamma_{j,2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & \gamma_{j,d} \end{pmatrix}$$

$$\boldsymbol{T}_d = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 1 & & 1 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix} \qquad \boldsymbol{T}_{\boldsymbol{\gamma},j} = \begin{pmatrix} \gamma_{j,1} & \gamma_{j,1} & \cdots & \gamma_{j,1} \\ 0 & \gamma_{j,2} & & \gamma_{j,2} \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & \gamma_{j,d} \end{pmatrix}$$

We define the following matrices:

$$\boldsymbol{L} = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c} 1 & \boldsymbol{0}_{1\times d} & \boldsymbol{0}_{1\times d} & \boldsymbol{0}_{1\times d} & \boldsymbol{0}_{1\times d} & \dots & \boldsymbol{0}_{1\times d} & \boldsymbol{1}_{1\times d} & \boldsymbol{1}_{1\times d} & \dots & \boldsymbol{1}_{1\times d} \\ \boldsymbol{0}_d & \boldsymbol{I}_d & \boldsymbol{0}_{d\times d} & \boldsymbol{I}_d & \boldsymbol{I}_d & \dots & \boldsymbol{I}_d & \boldsymbol{T}_d & \boldsymbol{T}_d & \dots & \boldsymbol{T}_d \end{array}\right)$$

$$\boldsymbol{M} = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c} \boldsymbol{0}_d & \boldsymbol{0}_{d\times d} & \boldsymbol{I}_d & \boldsymbol{I}_d & \boldsymbol{D}_{\boldsymbol{\gamma},1} & \dots & \boldsymbol{D}_{\boldsymbol{\gamma},d} & \boldsymbol{T}_d & \boldsymbol{T}_{\boldsymbol{\gamma},1} & \dots & \boldsymbol{T}_{\boldsymbol{\gamma},d} \end{array}\right)$$

Condition 4.1 is satisfied for a matrix $\boldsymbol{\gamma}$ *if for any vector $\boldsymbol{v} \in \mathbb{F}_q^{\ell}$ of Hamming weight $\mathsf{hw}(\boldsymbol{v}) \le d$ such that $\boldsymbol{L}\cdot\boldsymbol{v}$ contains no coefficient equal to* 0 *then $\boldsymbol{M}\cdot\boldsymbol{v} \neq \boldsymbol{0}_d$.*

Let us explain how this condition was constructed. The rows of $\boldsymbol{L}$ correspond to $a_0, \dots, a_d$. The rows of $\boldsymbol{M}$ correspond to $r_1, \dots, r_d$. Any linear combination of the probes involving only the a_i's and the r_i's can be written as

$$(a_0, \dots, a_d) \cdot \boldsymbol{L} \cdot \boldsymbol{v} + (r_1, \dots, r_d) \cdot \boldsymbol{M} \cdot \boldsymbol{v} \ .$$

Hence the above condition.

Remark 4.2. If all the vectors $\boldsymbol{v} \in \mathbb{F}_q^{\ell}$ of Hamming weight $\mathsf{hw}(\boldsymbol{v}) \le d$ were considered, this condition would be equivalent to saying that the linear code of parity-check matrix $\boldsymbol{M}$ has minimum distance at least d. However, as we only consider vectors $\boldsymbol{v}$ such that additionally $\boldsymbol{L}\cdot\boldsymbol{v}$ contains no coefficient equal to 0, this simple relation to codes is not true. We remark however that if the linear code of parity-check matrix $\boldsymbol{M}$ has minimum distance at least d, then the condition would be satisfied. Unfortunately for us, this code clearly has minimum distance 1, as it contains the vector $(1, 0, \dots, 0)^\intercal \in \mathbb{F}_q^\ell$. That is why we cannot naively use classical coding theory results to prove the existence of a matrix $\boldsymbol{\gamma}$ satisfying Condition 4.1.

We remark that the same necessary condition should hold for the matrix $\boldsymbol{\delta}$ by symmetry between $a_i, r_i, \boldsymbol{\gamma}$ and $b_i, s_i, \boldsymbol{\delta}$. Therefore, the formal condition we are considering is the following.

Condition 4.2. Condition 4.2 holds (for a matrix $\boldsymbol{\gamma} \in \mathbb{F}_q^{d \times d}$) *if Condition 4.1 is satisfied for both $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$, where $\boldsymbol{\delta} \in \mathbb{F}_q^{d \times d}$ is the matrix defined by $\delta_{i,j} = 1 - \gamma_{j,i}$.*

4.2 Security Analysis

We have shown that Condition 4.2 is necessary for our gadget (Algorithm 4) to be d-NI. The next theorem shows it is also sufficient for it to be d-SNI.

Theorem 4.3. *If $\boldsymbol{\gamma} \in \mathbb{F}_q^{d \times d}$ satisfies Condition 4.2 and if $q > d + 1$, then Algorithm 4 is d-SNI.*

To prove this theorem, we use the following lemma.

Lemma 4.4. *Let P be a set of t probes in Algorithm 1 such that $t \leq d$. Then, there exists a set Q_1 of at most t probes involving only the a_i's and the r_i's and a set Q_2 of at most t probes involving only the b_i's and the s_i's, such that the set P can be simulated by the probes in $Q_1 \cup Q_2$.*

Proof (Lemma 4.4). We list hereafter all the possible probes in Algorithm 1. We gather them by sets for the needs of the proof.

Set 1: a_i, r_i, $r_i + a_i$, $\gamma_{j,i} r_i$, $\gamma_{j,i} r_i + a_i$, (for $0 \leq i \leq d$ and $1 \leq j \leq d$);
Set 2: $a_0 + \sum_{i=1}^{k} (r_i + a_i)$ (for $1 \leq k \leq d$);
Set 3: $a_0 + \sum_{i=1}^{k} (\gamma_{j,i} r_i + a_i)$ (for $1 \leq j \leq d$ and $1 \leq k \leq d$);
Set 4: b_i, s_i, $s_i + b_i$, $\delta_{j,i} s_i$, $\delta_{j,i} s_i + b_i$, (for $0 \leq i \leq d$ and $1 \leq j \leq d$);
Set 5: $b_0 + \sum_{i=1}^{k} (s_i + b_i)$ (for $1 \leq k \leq d$);
Set 6: $b_0 + \sum_{i=1}^{k} (\delta_{j,i} s_i + b_i)$ (for $1 \leq j \leq d$ and $1 \leq k \leq d$);
Set 7: $-r_i \cdot \left(b_0 + \sum_{j=1}^{d} (\delta_{i,j} s_j + b_j) \right)$ (for $1 \leq i \leq d$);
Set 8: $-s_i \cdot \left(a_0 + \sum_{j=1}^{d} (\gamma_{i,j} r_j + a_j) \right)$ (for $1 \leq i \leq d$);
Set 9: $(a_0 + \sum_{i=1}^{d} (r_i + a_i)) \cdot (b_0 + \sum_{i=1}^{d} (s_i + b_i))$.

Let us now consider a set P of t probes among the listed ones. We initialize two sets Q_1 and Q_2 to the empty set and show how to fill them with at most t probes involving only the a_i's and the r_i's for Q_1 and at most t probes involving only the b_i's and the s_i's for Q_2 in such a way that P can be perfectly simulated by probes of $Q_1 \cup Q_2$.

For all the probes of P which belong to Sets 1 to 3, then we add them directly to Q_1 since they only depend on a_i's, r_i's and constants. Similarly, for all the probes of P which belong to Sets 4 to 6, then we add them directly to Q_2 since they only depend on b_i's, s_i's and constants. For P's probes belonging to Set 7, we add probe $-r_i$ to Q_1 and $b_0 + \sum_{j=1}^{d}(\delta_{i,j}s_i + b_j)$ to Q_2. For P's probes belonging to Set 8, we add probe $-s_i$ to Q_2 and $a_0 + \sum_{j=1}^{d}(\gamma_{i,j}r_i + a_j)$ to Q_1. Finally, for probes of P from Set 9, we add $a_0 + \sum_{i=1}^{d}(r_i + a_i)$ to Q_1 and $b_0 + \sum_{i=1}^{d}(s_i + b_i)$ to Q_2. Since for each probe of P, at most one probe was added to Q_1 and at most one probe was added to Q_2, it is clear that after all the t probes of P are processed, Q_1 and Q_2 contain at most t probes each.

Let us now prove that all the probes of P can be perfectly simulated by the probes of $Q_1 \cup Q_2$. For probes of P belonging to six first sets, the exact same values were added to Q_1 (for the three first sets) or Q_2 (for Set 4 to 6) thus the simulation is trivial. For probes of P in Set 7, $-r_i$ was added to Q_1 and $b_0 + \sum_{j=1}^{d}(\delta_{i,j}s_i + b_j)$ to Q_2. The multiplication of these two probes perfectly simulate the initial probe of P. The same conclusions can be made for probes of P in Sets 8 and 9 since each time probes were added to Q_1 and Q_2 so that their product corresponds to the initial probe of P. $\square$

Proof (Theorem 4.3). From Lemma 4.4, any set P of $t \leq d$ probes in Algorithm 1 can be perfectly simulated by probes of two sets Q_1 and Q_2 of cardinal at most t and containing probes involving only the a_i's and the r_i's for Q_1 and probes involving only the b_i's and the s_i's for Q_2.

From Condition 4.2, any combination of the t probes of Q_1 either depend on strictly less than t a_i's or it is functionally dependent on at least one r_i. Thanks to Theorem 3.5 and the fact that $q > d + 1$, the t probes of Q_1 can be perfectly simulated using at most t shares a_i. The same statement can be made for the probes of Q_2. Therefore, from Lemma 4.4, any set of $t \leq d$ probes on Algorithm 1 can be perfectly simulated by at most t shares a_i and t shares b_i, which proves that Algorithm 1 is d-TNI.

Since from Proposition 4.1, SharingCompress$[2d + 1 : d + 1]$ is d-SNI, from the composition theorems established in [3], Algorithm 4 is d-SNI. $\square$

4.3 Probabilistic Construction

In order to prove the existence of a matrix $\boldsymbol{\gamma}$ which satisfies Condition 4.1 for q large enough (but only exponential in $d \log d$), we state Theorem 4.5 that makes use of the non-constructive "probabilistic method." More precisely, we prove that if one chooses $\boldsymbol{\gamma}$ uniformly at random in $\mathbb{F}_q^{d\times d}$, the probability that the matrix $\boldsymbol{\gamma}$ satisfies Condition 4.2 is more than zero, when q is large enough. The proof of Theorem 4.5 uses probability but the existence of a matrix $\boldsymbol{\gamma}$ which satisfies Condition 4.2 (for q large enough) is guaranteed without any possible error.

Theorem 4.5. *For any $d \geq 1$, for any prime power q, if $\boldsymbol{\gamma}$ is chosen uniformly in $\mathbb{F}_q^{d\times d}$, then*

$$\Pr[\boldsymbol{\gamma} \textit{ satisfies Condition 4.2}] \geq 1 - 2 \cdot (12d)^d \cdot d \cdot q^{-1} \quad .$$

In particular, for any $d \geq 1$, there exists an integer $Q = O(d)^{d+1}$, such that for any prime power $q \geq Q$, there exists a matrix $\boldsymbol{\gamma} \in \mathbb{F}_q^{d \times d}$ satisfying Condition 4.2.

As when $\boldsymbol{\gamma}$ is uniformly random, so is $\boldsymbol{\delta}$, Theorem 4.5 immediately follows from the following proposition and the union bound.

Proposition 4.6. *For any $d \geq 1$, for any prime power q, if $\boldsymbol{\gamma}$ is chosen uniformly in $\mathbb{F}_q^{d \times d}$, then*

$$\Pr[\boldsymbol{\gamma} \textit{ satisfies Condition 4.1}] \geq 1 - (12d)^d \cdot d \cdot q^{-1} \ .$$

In particular, for any $d \geq 1$, there exists an integer $Q = O(d)^{d+1}$, such that for any prime power $q \geq Q$, there exists a matrix $\boldsymbol{\gamma} \in \mathbb{F}_q^{d \times d}$ satisfying Condition 4.1.

The proof of this proposition is very technical and is provided in the full version.

Remark 4.7. Note that the constants in the previous proof are not the best possible and can be improved. In the following, we present explicit constructions for small values of d.

4.4 Small Cases

We show here the instantiation for $d = 2$. The case for $d = 3$ is similar and is provided in details in the full version.

Let $d = 2$. Let us now explicitly instantiate our construction for any non-prime field $\mathbb{F}_q$ where $q = p^k$, $k \geq 2$. Let ξ be any element in $\mathbb{F}_q \setminus \mathbb{F}_p$. A possible instantiation is:

$$\boldsymbol{\gamma} = \begin{pmatrix} \xi & \xi+1 \\ \xi+1 & \xi \end{pmatrix}, \quad \boldsymbol{\delta} = \begin{pmatrix} -\xi+1 & -\xi \\ -\xi & -\xi+1 \end{pmatrix} \ .$$

The computed shares are hence:

- $c_0 = (a_0 + (r_1 + a_1) + (r_2 + a_2)) \cdot (b_0 + (s_1 + b_1) + (s_2 + b_2))$
- $c_1 = -r_1 \cdot (b_0 + ((-\xi + 1)s_1 + b_1) + (-\xi s_2 + b_2))$
- $c_2 = -r_2 \cdot (b_0 + (-\xi s_1 + b_1) + ((-\xi + 1)s_2 + b_2))$
- $c_3 = -s_1 \cdot (a_0 + (\xi r_1 + a_1) + ((\xi + 1)r_2 + a_2))$
- $c_4 = -s_2 \cdot (a_0 + ((\xi + 1)r_1 + a_1) + (\xi r_2 + a_2))$

Let us now prove that this scheme satisfies Condition 4.2. Let us consider the matrices $\boldsymbol{L}$ and $\boldsymbol{M}$ as defined in Condition 4.1:

$$\boldsymbol{L} = \left(\begin{array}{c|cc|cc|cc|cc|cc|cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right)$$

$$\boldsymbol{M} = \left(\begin{array}{c|cc|cc|cc|cc|cc|cc|cc|cc} 0 & 0 & 0 & 1 & 0 & 1 & 0 & \xi & 0 & \xi+1 & 0 & 1 & 1 & \xi & \xi & \xi+1 & \xi+1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \xi+1 & 0 & \xi & 0 & 1 & 0 & \xi+1 & 0 & \xi \end{array}\right)$$

We will prove that, for any vector $\boldsymbol{v}$ such that $\mathsf{hw}(\boldsymbol{v}) \leq 2$, it holds that if $\boldsymbol{M} \cdot \boldsymbol{v} = \mathbf{0}_2$, then $\boldsymbol{L} \cdot \boldsymbol{v}$ has a 0 coefficient.

Let us start by the case $\mathsf{hw}(\boldsymbol{v}) = 1$. If $\boldsymbol{M} \cdot \boldsymbol{v} = \mathbf{0}_2$, the only non-zero coefficient of $\boldsymbol{v}$ clearly must be in one of the first $1 + d = 3$ coordinates. Denote by i the index of this coefficient. Since $i \leq 3$, from the definition of $\boldsymbol{L}$, we have $\boldsymbol{L} \cdot \boldsymbol{v} = \boldsymbol{I}_3 \cdot (v_1, v_2, v_3)^\intercal$, and thus its i-th coefficient is equal to the non-zero coefficient of $\boldsymbol{v}$ but the two other coefficients of $\boldsymbol{L} \cdot \boldsymbol{v}$ are equal to 0. This concludes this case.

Let us tackle the case $\mathsf{hw}(\boldsymbol{v}) = 2$. Note that $\boldsymbol{L} \cdot \boldsymbol{v}$ hence corresponds to a linear combination of exactly two columns of $\boldsymbol{L}$. By construction of $\boldsymbol{L}$, all first columns (until the occurrence of $\boldsymbol{T}_d$) are of Hamming weight 1. Consequently, for $\boldsymbol{L} \cdot \boldsymbol{v}$ to have only non-zero coefficients, at least one of the $3 \cdot d = 6$ last coordinates of $\boldsymbol{v}$ must be non-zero. The corresponding columns of $\boldsymbol{L}$ have two possible values : $(1, 1, 0)^\top$ or $(1, 1, 1)^\top$. Let us consider the cases where one coordinate of $\boldsymbol{v}$ corresponding to a column $(1, 1, 0)^\top$ is set. The corresponding column in $\boldsymbol{M}$ is of the form $(\alpha, 0)^\top$, where α can be $1, \xi, \xi + 1$. In order for $\boldsymbol{L} \cdot \boldsymbol{v}$ to have only non-zero coefficients, the other non-zero coordinate of $\boldsymbol{v}$ must correspond to a column of $\boldsymbol{L}$ where the last coefficient is non-zero. However, for all of these columns, the corresponding column of $\boldsymbol{M}$ is always of the form (λ, β), with $\beta \neq 0$, in which case $\boldsymbol{M} \cdot \boldsymbol{v} \neq \mathbf{0}_2$. It just remains to consider the case where one non-zero coordinate of $\boldsymbol{v}$ corresponds to a column $(1, 1, 1)^\top$ of $\boldsymbol{L}$. The corresponding columns in $\boldsymbol{M}$ can be $(1, 1)^\top$, $(\xi, \xi + 1)^\top$, or $(\xi + 1, \xi)^\top$. Note that for no other column in $\boldsymbol{L}$ one can retrieve a corresponding column in $\boldsymbol{M}$ whose coefficients are both non-zero. Consequently, both non-zero coordinates of $\boldsymbol{v}$ must correspond to columns $(1, 1, 1)^\top$ of $\boldsymbol{L}$. Since no two vectors among $(1, 1)$, $(\xi, \xi + 1)$, and $(\xi + 1, \xi)$ are proportional, then we always have $\boldsymbol{M} \cdot \boldsymbol{v} \neq \mathbf{0}_2$.

The exact same reasoning can be held for $\boldsymbol{\delta}$, since no two vectors among $(1, 1)$, $(-\xi + 1, -\xi)$, $(-\xi, -\xi + 1)$ are proportional.

5 Construction with Linear Randomness Complexity

In this section, we describe a construction that only requires a linear randomness complexity. That is, our $(d + 1, d + 1)$-gadget only uses d random scalars. In particular, our construction breaks the linear bound of $d + 1$ random scalars (for order $d \geq 3$) proven in [4]. There is no contradiction since this lower bound is proven only in $\mathbb{F}_2$. Our construction is described below and once again makes use of a matrix of scalars that needs to satisfy certain properties, as explained later in this section.

5.1 Construction

Construction. Let $\boldsymbol{\gamma} = (\gamma_{i,j})_{\substack{0 \leq i \leq d \\ 1 \leq j \leq d}} \in \mathbb{F}_q^{(d+1) \times d}$ be a constant matrix (with $d + 1$ rows instead of d for the previous construction).

Following the previous gadget with the objective of minimizing the randomness complexity, we can construct a multiplication $(d+1, d+1)$-gadget which outputs the shares $(c_0, \ldots, c_d)$ defined as follows:

$$c_i = a_0 b_i + \sum_{j=1}^{d} (\gamma_{i,j} r_j + a_j b_i) \, ,$$

for $0 \leq i \leq d$. The gadget is formally depicted in Algorithm 5 and is correct under the condition that for any $0 \leq j \leq d$,

$$\sum_{i=0}^{d} \gamma_{i,j} = 0 \ \ .$$

Algorithm 5 New Construction with Linear Randomness

```
Require: $\boldsymbol{a} = (a_0, \ldots, a_d), \boldsymbol{b} = (b_0, \ldots, b_d)$
Ensure: $\boldsymbol{c} = (c_0, \ldots, c_d)$ such that $\sum_{i=0}^{d} c_i = (\sum_{i=0}^{d} a_i) \cdot (\sum_{i=0}^{d} b_i)$
  for $i = 1$ to $d$ do
      $c_i \leftarrow a_0 b_i$
  for $j = 1$ to $d$ do
      $r_j \leftarrow \$$
      for $i = 0$ to $d$ do
          $c_i \leftarrow c_i + (\gamma_{i,j} r_j + a_j b_i)$
  return $(c_0, \ldots, c_d)$
```

We remark that if this construction is secure, it breaks the randomness complexity lower bound of $d+1$ random bits proven in [4] when $q = 2$. Furthermore, it is the first construction with a linear number of random scalars (in d). Previously, the construction with the best randomness complexity used a quasi-linear number of random scalars [4].

However, as for our construction in Section 4.1, the construction is clearly not secure for every matrix $\boldsymbol{\gamma}$. For example, if $\boldsymbol{\gamma}$ is a matrix of zeros, the gadget is clearly not private, let alone NI or SNI. Actually, it is not even clear that there exists a matrix $\boldsymbol{\gamma}$ for which the gadget is private. We prove in the following that this is indeed the case if the finite field is large enough and we provide explicit choices of the matrix $\boldsymbol{\gamma}$ for small orders $d \in \{2, 3\}$ over small finite fields.

Condition on $\boldsymbol{\gamma}$. Similarly to Section 4.1, the following condition is necessary for the above construction to be d-NI.

Condition 5.1. *Let $\ell = (2d+4) \cdot d + 1$. Let $\boldsymbol{I}_d \in \mathbb{F}_q^{d \times d}$ be the identity matrix, $\mathbf{0}_{m \times n} \in \mathbb{F}_q^{m \times n}$ be a matrix of zeros (when $n = 1$, $\mathbf{0}_{m \times n}$ is also written $\mathbf{0}_m$), $\mathbf{1}_{m \times n} \in \mathbb{F}_q^{m \times n}$ be a matrix of ones, $\boldsymbol{D}_{\boldsymbol{\gamma},j} \in \mathbb{F}_q^{d \times d}$ be the diagonal matrix such* *that* $D_{\boldsymbol{\gamma},j,i,i} = \gamma_{j,i}$, $\boldsymbol{T}_d \in \mathbb{F}_q^{d\times d}$ *be the upper-triangular matrix with just ones,* $\boldsymbol{T}_{\boldsymbol{\gamma},j} \in \mathbb{F}_q^{d\times d}$ *be the upper-triangular matrix for which* $T_{\boldsymbol{\gamma},j,i,k} = \gamma_{j,i}$ *for* $i \le k$. *Let* $\omega_0, \ldots, \omega_d$ *be* $(d+1)$ *indeterminates and we consider the field of rational fractions* $\mathbb{F}_q(\omega_0, \ldots, \omega_d)$. *In other words, we have:*

$$\boldsymbol{I}_d = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix} \qquad \boldsymbol{D}_{\boldsymbol{\gamma},j} = \begin{pmatrix} \gamma_{j,1} & 0 & \ldots & 0 \\ 0 & \gamma_{j,2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & \gamma_{j,d} \end{pmatrix}$$

$$\boldsymbol{T}_d = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 0 & 1 & & 1 \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{pmatrix} \qquad \boldsymbol{T}_{\boldsymbol{\gamma},j} = \begin{pmatrix} \gamma_{j,1} & \gamma_{j,1} & \ldots & \gamma_{j,1} \\ 0 & \gamma_{j,2} & & \gamma_{j,2} \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & \gamma_{j,d} \end{pmatrix}$$

We define the following matrices:

$$\boldsymbol{L'} = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c} 1 & \boldsymbol{0}_{1\times d} & \boldsymbol{0}_{1\times d} & \boldsymbol{0}_{1\times d} & \boldsymbol{0}_{1\times d} & \ldots & \boldsymbol{0}_{1\times d} & \omega_0 \boldsymbol{1}_{1\times d} & \omega_1 \boldsymbol{1}_{1\times d} & \ldots & \omega_d \boldsymbol{1}_{1\times d} \\ \boldsymbol{0}_d & \boldsymbol{I}_d & \boldsymbol{0}_{d\times d} & \omega_0 \boldsymbol{I}_d & \omega_1 \boldsymbol{I}_d & \ldots & \omega_d \boldsymbol{I}_d & \omega_0 \boldsymbol{T}_d & \omega_1 \boldsymbol{T}_d & \ldots & \omega_d \boldsymbol{T}_d \end{array} \right)$$

$$\boldsymbol{M'} = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c} \boldsymbol{0}_d & \boldsymbol{0}_{d\times d} & \boldsymbol{I}_d & \boldsymbol{D}_{\boldsymbol{\gamma},0} & \boldsymbol{D}_{\boldsymbol{\gamma},1} & \ldots & \boldsymbol{D}_{\boldsymbol{\gamma},d} & \boldsymbol{T}_{\boldsymbol{\gamma},0} & \boldsymbol{T}_{\boldsymbol{\gamma},1} & \ldots & \boldsymbol{T}_{\boldsymbol{\gamma},d} \end{array} \right)$$

where $\boldsymbol{L'} \in \mathbb{F}_q(\omega_0, \ldots, \omega_d)^{(d+1)\times \ell}$ *and* $\boldsymbol{M'} \in \mathbb{F}_q^{d\times \ell}$.
Condition 5.1 is satisfied for a matrix $\boldsymbol{\gamma}$ *if for any vector* $\boldsymbol{v} \in \mathbb{F}_q^{\ell}$ *of Hamming weight* $\mathsf{hw}(\boldsymbol{v}) \le d$ *such that* $\boldsymbol{L'} \cdot \boldsymbol{v}$ *contains no coefficient equal to* 0 *then* $\boldsymbol{M'} \cdot \boldsymbol{v} \neq \boldsymbol{0}_d$.

5.2 Security Analysis

Lemma 5.1. *Each probe contains at most one share* b_i *of* b.

Proof. A probe can only target the partial expression of an output or an entire output. In this construction, each output c_i is built with a single share b_i of b. Therefore, a probe can contain at most one such share. □

Corollary 5.2. *Any set of at most* d *probes contains at most* d *shares of* b.

Proposition 5.3. *The above construction with* d *random scalars is* d*-NI, if* $\boldsymbol{\gamma}$ *satisfies Condition 5.1.*

Proof. From Condition 5.1, any combination of at most d probes in our construction is either functionally dependent on at most d shares a_i or on at least one random scalar. Furthermore, using in addition Corollary 5.2, any combination of at most d probes is functionally dependent on at most d shares b_i. Therefore, thanks to Theorem 3.5 and the fact that $q > d+1$, the construction is d-NI. □

5.3 Probabilistic Construction

As in the previous section, in order to prove the existence of a matrix $\boldsymbol{\gamma}$ which satisfies Condition 4.2 for q large enough (but only exponential in $d \log d$), we state Theorem 5.4 that makes also use of the non-constructive "probabilistic method." Its proof is detailed in the full version.

Theorem 5.4. *For any $d \geq 1$, for any prime power q, if $\boldsymbol{\gamma}$ is chosen uniformly in $\boldsymbol{\gamma} \in \mathbb{F}_q^{(d+1)\times d}$ under the condition that $\sum_{i=0}^{d} \gamma_{i,j} = 0$ for $0 \leq i \leq d$, then*

$$\Pr[\boldsymbol{\gamma} \text{ satisfies Condition 4.2}] \geq 1 - d(d+1) \cdot (12d)^d \cdot q^{-1}$$

In particular, for any $d \geq 1$, there exists an integer $Q = O(d^{d+2})$, such that for any prime power $q \geq Q$, there exists a matrix $\boldsymbol{\gamma} \in \mathbb{F}_q^{d\times d}$ satisfying Condition 5.1.

5.4 Small Cases

We show here the instanciation for $d \in \{2, 3\}$.

$\boldsymbol{d = 2}$. Let d equal 2. Let us now explicitly instantiate our construction for any non-prime field $\mathbb{F}_q$ where $q = p^k$, $k \geq 2$. Let ξ be any element in $\mathbb{F}_q \setminus \mathbb{F}_p$. A possible instantiation is:

$$\boldsymbol{\gamma} = \begin{pmatrix} 1 & \xi \\ \xi & 1 \\ -\xi - 1 & -\xi - 1 \end{pmatrix}.$$

The computed shares are hence:

- $c_0 = a_0 b_0 + (1 \cdot r_1 + a_1 b_0) + (\xi \cdot r_2 + a_2 b_0)$
- $c_1 = a_0 b_1 + (\xi \cdot r_1 + a_1 b_1) + (1 \cdot r_2 + a_2 b_1)$
- $c_2 = a_0 b_2 + ((-\xi - 1) \cdot r_1 + a_1 b_2) + (-\xi - 1) \cdot r_2 + a_2 b_2)$

Let us now prove that this scheme satisfies Condition 5.1. The reasoning is similar to the proof in Section 4.4.

In order for $\boldsymbol{M'} \cdot \boldsymbol{v}$ to be null, and for $\boldsymbol{L'} \cdot \boldsymbol{v}$ to be of full Hamming weight, we observe that the two non-zero coefficients of $\boldsymbol{v}$ must correspond to two columns of full Hamming weight of $\boldsymbol{M'}$. However, no two vectors in $(1, \xi), (\xi, 1), (-\xi - 1, -\xi - 1)$ are proportional. This ensures that Condition 5.1 is satisfied for $\boldsymbol{\gamma}$.

$\boldsymbol{d = 3}$. Let d equal 3. Let us now explicitly instantiate our construction for any non-prime field $\mathbb{F}_q$ where $q = 2^k$, $k \geq 4$. Let ξ be any element in $\mathbb{F}_q \setminus \mathbb{F}_p$. A possible instantiation is:

$$\boldsymbol{\gamma} = \begin{pmatrix} 1 & \xi & \xi + 1 \\ 1 & \xi^2 + 1 & \xi \\ 1 & \xi + 1 & \xi^2 + \xi + 1 \\ 1 & \xi^2 + \xi + 1 & \xi + 1 \end{pmatrix}.$$

The computed shares are hence:

- $c_0 = a_0b_0 + (1 \cdot r_1 + a_1b_0) + (\xi \cdot r_2 + a_2b_0) + ((\xi+1) \cdot r_3 + a_3b_0)$
- $c_1 = a_0b_1 + (1 \cdot r_1 + a_1b_1) + ((\xi^2+1) \cdot r_2 + a_2b_1) + (\xi \cdot r_3 + a_3b_1)$
- $c_2 = a_0b_2 + (1 \cdot r_1 + a_1b_2) + ((\xi+1) \cdot r_2 + a_2b_2) + ((\xi^2+\xi+1\xi) \cdot r_3 + a_3b_2)$
- $c_3 = a_0b_3 + (1 \cdot r_1 + a_1b_3) + ((\xi^2+\xi+1) \cdot r_2 + a_2b_3) + ((\xi+1) \cdot r_3 + a_3b_3)$

Let us now prove that this scheme satisfies Condition 5.1. The reasoning is similar to the proof in Section 4.4. We check the non-proportionality of the relevant vectors $(1,\xi,\xi+1), (1,\xi^2+1,\xi), (1,\xi+1,\xi^2+\xi+1), (1,\xi^2+\xi+1,\xi+1)$, and finish by computing all left determinants using a computer algebra system. It follows that this construction satisfies Condition 5.1.

5.5 Lower Bound

Let us now show a lower bound on the randomness complexity of d-NI multiplication gadgets satisfying the following condition.

Condition 5.2. *A multiplication gadget satisfies Condition 5.2 if the output shares are affine functions (over $\mathbb{F}_q$) of the products a_ib_j and of the input shares a_i and b_j (coefficients of the affine functions may depend on the random scalars). In other words, each output share c_i can be written as (possibly after expansion and simplification):*

$$c_i = \boldsymbol{a}^\intercal \cdot \boldsymbol{M_i}(\boldsymbol{r}) \cdot \boldsymbol{b} \;+\; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu}_i(\boldsymbol{r}) \;+\; \boldsymbol{\nu}_i^\intercal(\boldsymbol{r}) \cdot \boldsymbol{b} \;+\; \tau_i(\boldsymbol{r}),$$

where $\boldsymbol{M_i}(\boldsymbol{r}) \in \mathbb{F}_q^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}_i(\boldsymbol{r}) \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\nu}_i(\boldsymbol{r}) \in \mathbb{F}_q^{d+1}$, and $\tau_i(\boldsymbol{r}) \in \mathbb{F}_q$ are arbitrary functions of the vector $\boldsymbol{r} \in \mathbb{F}_q^R$ of random scalars.

This condition is very weak. In particular, it does not restrict output shares to be bilinear and do not restrict internal values of the circuit at all. All the d-NI multiplication gadgets we know [4, 10, 18, 24] including the ours in Sections 4.1 and 5.1 satisfy this condition. We first need the following lemma.

Lemma 5.5. *Let $\boldsymbol{U} \in \mathbb{F}_q^{(d+1)\times(d+1)}$ be the matrix of ones. Let $\boldsymbol{M}, \boldsymbol{M'}$ be two matrices in $\mathbb{F}_q^{(d+1)\times(d+1)}$ such that $\boldsymbol{M} + \boldsymbol{M'} = \boldsymbol{U}$. Then all the columns or all the rows of $\boldsymbol{M}$, or all the columns or all the rows of $\boldsymbol{M'}$ are non-zero.*

Proof. Let us prove the lemma by contraposition. We suppose that both $\boldsymbol{M}$ and $\boldsymbol{M'}$ have a column of zeros and a row of zeros. Let us suppose that the i-th row of $\boldsymbol{M}$ is a zero row and the j-th column of $\boldsymbol{M'}$ is a zero column. Then $M_{i,j} = M'_{i,j} = 0 \neq 1 = U_{i,j}$ and $\boldsymbol{M} + \boldsymbol{M'} \neq \boldsymbol{U}$. □

We can now state our lower bound.

Proposition 5.6. *Let C be a d-NI multiplication gadget satisfying Condition 5.2. Then C uses more than $\lfloor (d-1)/2 \rfloor$ random scalars (i.e., $R \geq d/2$).*

A d-NI multiplication gadget satisfying Condition 5.2 thus requires a linear number of random scalars in d. We recall our construction in Section 5.1 uses d random scalars, which is linear in d.

Proof. Let us suppose that C uses only $R \leq \lfloor (d-1)/2 \rfloor$ random scalars. Let $k = \lfloor d/2 \rfloor$. Let us construct a set of probes which cannot be simulated by at most d shares of each input a and b. As C satisfies Condition 5.2, we can write:

$$\begin{aligned} c_0 + \cdots + c_k &= \boldsymbol{a}^\intercal \cdot \boldsymbol{M}(\boldsymbol{r}) \cdot \boldsymbol{b} \;+\; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu}(\boldsymbol{r}) \;+\; \boldsymbol{\nu}^\intercal(\boldsymbol{r}) \cdot \boldsymbol{b} \;+\; \tau(\boldsymbol{r})\,, \\ c_{k+1} + \cdots + c_d &= \boldsymbol{a}^\intercal \cdot \boldsymbol{M'}(\boldsymbol{r}) \cdot \boldsymbol{b} \;+\; \boldsymbol{a}^\intercal \cdot \boldsymbol{\mu'}(\boldsymbol{r}) \;+\; \boldsymbol{\nu'}^\intercal(\boldsymbol{r}) \cdot \boldsymbol{b} \;+\; \tau'(\boldsymbol{r})\,, \end{aligned}$$

where $\boldsymbol{M}(\boldsymbol{r}), \boldsymbol{M'}(\boldsymbol{r}) \in \mathbb{F}_q^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}(\boldsymbol{r}), \boldsymbol{\mu}(\boldsymbol{r}) \in \mathbb{F}_q^{d+1}$, $\boldsymbol{\nu}(\boldsymbol{r}), \boldsymbol{\nu}(\boldsymbol{r}) \in \mathbb{F}_q^{d+1}$, and $\tau(\boldsymbol{r}), \tau'(\boldsymbol{r}) \in \mathbb{F}_q$ are arbitrary functions of the vector $\boldsymbol{r} \in \mathbb{F}_q^R$ of random scalars.

Let $\boldsymbol{U} \in \mathbb{F}_q^{(d+1)\times(d+1)}$ be the matrix of ones. As $\sum_{i=0}^{d} c_i = ab = \boldsymbol{a}^\intercal \cdot \boldsymbol{U} \cdot \boldsymbol{b}$ by correctnesss of C, we have $\boldsymbol{M}(\boldsymbol{r}) + \boldsymbol{M'}(\boldsymbol{r}) = \boldsymbol{U}$. In particular, when $\boldsymbol{r} = \boldsymbol{0}$ (for example), Lemma 5.5 ensures that $c_0 + \cdots + c_k$ or $c_{k+1} + \cdots + c_d$ functionally depends on every a_i $(0 \leq i \leq d)$ or on every b_j $(0 \leq j \leq d)$. Therefore, one of the following set of probes cannot be simulated by at most d shares of each input a and b:

$$\{r_1, \ldots, r_R, c_0, \ldots, c_k\} \quad \text{and} \quad \{r_1, \ldots, r_R, c_{k+1}, \ldots, c_d\}\ .$$

We conclude by remarking that $R+(k+1) \leq \lfloor (d-1)/2 \rfloor + \lfloor d/2 \rfloor + 1 \leq d$, as either $d-1$ or d is odd and so either $\lfloor (d-1)/2 \rfloor \leq (d-1)/2 - 1$ or $\lfloor d/2 \rfloor \leq d/2 - 1$. $\square$

Acknowledgements. The second author was supported by the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236. The third author was supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government. The fourth and fifth authors were supported in part by the European Union's H2020 Programme under grant agreement number ICT-731591 (REASSURE). The fifth author was supported in part by the French ANR project BRUTUS, ANR-14-CE28-0015.

References

1. Balasch, J., Faust, S., Gierlichs, B.: Inner product masking revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 486–510. Springer, Heidelberg (Apr 2015)
2. Balasch, J., Faust, S., Gierlichs, B., Verbauwhede, I.: Theory and practice of a leakage resilient masking scheme. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 758–775. Springer, Heidelberg (Dec 2012)

3. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16. pp. 116–129. ACM Press (Oct 2016)
4. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness complexity of private circuits for multiplication. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 616–648. Springer, Heidelberg (May 2016)
5. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness complexity of private circuits for multiplication. Cryptology ePrint Archive, Report 2016/211 (2016), full version of [4]. http://eprint.iacr.org/2016/211
6. Carlet, C., Prouff, E.: Polynomial evaluation and side channel analysis. In: Ryan, P.Y.A., Naccache, D., Quisquater, J. (eds.) The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday. Lecture Notes in Computer Science, vol. 9100, pp. 315–341. Springer (2016), http://dx.doi.org/10.1007/978-3-662-49301-4_20
7. Carlet, C., Prouff, E., Rivain, M., Roche, T.: Algebraic decomposition for probing security. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 742–763. Springer, Heidelberg (Aug 2015)
8. Carlet, C., Prouff, E., Rivain, M., Roche, T.: Algebraic decomposition for probing security. Cryptology ePrint Archive, Report 2016/321 (2016), full version of [7]. http://eprint.iacr.org/2016/321
9. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (Aug 1999)
10. Coron, J.S., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 410–424. Springer, Heidelberg (Mar 2014)
11. Coron, J., Prouff, E., Roche, T.: On the use of Shamir's secret sharing against side-channel analysis. In: Mangard, S. (ed.) Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7771, pp. 77–90. Springer (2012)
12. Coron, J.S., Roy, A., Vivek, S.: Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 170–187. Springer, Heidelberg (Sep 2014)
13. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Heidelberg (May 2014)
14. Duc, A., Faust, S., Standaert, F.X.: Making masking security proofs concrete - or how to evaluate the security of any leaking device. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 401–429. Springer, Heidelberg (Apr 2015)
15. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from leakage: the computationally-bounded and noisy cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (May 2010)

16. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Çetin Kaya., Paar, C. (eds.) CHES'99. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (Aug 1999)
17. Gross, H., Mangard, S., Korak, T.: Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. IACR Cryptology ePrint Archive 2016, 486 (2016), http://eprint.iacr.org/2016/486, To appear in the proceedings of CARDIS 2016.
18. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (Aug 2003)
19. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (Aug 1996)
20. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptology 24(2), 292–321 (2011), http://dx.doi.org/10.1007/s00145-010-9085-7
21. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (May 2013)
22. Prouff, E., Roche, T.: Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 63–78. Springer, Heidelberg (Sep / Oct 2011)
23. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 764–783. Springer, Heidelberg (Aug 2015)
24. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (Aug 2010)