High-order private multiplication in characteristic two revisited

Nicolas Bordes and Pierre Karpman

Université Grenoble Alpes, France

{nicolas.bordes,pierre.karpman}@univ-grenoble-alpes.fr

Abstract. We revisit the high-order masking schemes for private multiplication introduced by Belaïd *et al.* at EUROCRYPT 2016, and the matrix model for non-interference (NI) security that they develop in their follow-up work of CRYPTO 2017. This leads to two main results.
1) We generalise the theorems of CRYPTO 2017 so as to be able to apply them to masking schemes over any finite field — in particular $\mathbb{F}_2$ — and to be able to analyse the *strong* non-interference (SNI) security notion. This leads to an efficient algorithm that allows us to computationally check the (S)NI security of binary schemes up to order $d = 11$.
2) We propose new SNI and NI masking gadgets for multiplication over $\mathbb{F}_2$ (and any extension thereof) up to order 9 and 11 that improve the randomness complexity of the schemes of EUROCRYPT 2016 and of Ishai, Sahai and Wagner (CRYPTO 2003) respectively. A natural generalisation of the NI schemes is also conjectured to be secure at any order.

Keywords: High-order masking, multiplication gadgets, linear codes.

1 Introduction

Since their introduction in the late last century, side-channel attacks and in particular *Differential Power Analysis* (DPA) [KJJ99] have developed into one of the most efficient attack techniques on implementations of cryptographic primitives. The importance of this new threat and its practical relevance soon lead to the design of appropriate counter-measures, one of the most influential to date being the "ISW" private multiplication circuit of Ishai, Sahai and Wagner [ISW03]. This is a foremost example of a *masking scheme*, where sensitive data are split into several shares using a secret sharing scheme; the crux of the design is then to devise a way to perform field arithmetic over the shares without leaking too much information to the adversary in the process.

A major characteristic of a masking scheme is the *order* at which it is secure: in a probing model such as the one introduced by Ishai, Sahai and Wagner, a circuit secure at order d is such that no adversary can learn information about its input and output even when being given d intermediate values of its computation. The usefulness of increasing the security order is then justified by the fact that under reasonable assumptions, the number of measurements needed for a successful attack increases exponentially in d [DFS15].

Unfortunately, high-order schemes also come with a significant overhead, since the complexity of ISW multiplication is quadratic in d for three relevant metrics: to secure one field multiplication, one needs $2d(d + 1)$ sums, $(d + 1)^2$ products and $d(d + 1)/2$ fresh random masks. This lead to several attempts to find more efficient multiplication circuits, especially with respect to the last two metrics.

A number of new schemes for private multiplication were introduced in the past few years by Belaïd *et al.* [BBP+16,BBP+17]. At EUROCRYPT 2016, they design a new high-order scheme whose randomness complexity is decreased to $\approx d^2/4 + d$, and which can be easily instantiated over any finite field of characteristic two (they also give specific schemes with even lower cost up to order 4). The security of this multiplication is proven in the composable model of *non interference* (NI) from Barthe *et al.* [BBD+16]. This is slightly weaker than the *strong non-interference* (SNI) security achieved by ISW multiplication but remains of high practical relevance: for instance, one can replace half of the multiplications in a masked AES S-box computation by the ones of [BBP+16] while maintaining the overall strong SNI security for the entire S-box. At CRYPTO 2017, the same authors propose two new schemes, one with linear *bilinear multiplication* complexity, and the other with linear randomness complexity. However, those are complex to securely instantiate and cannot be done so over $\mathbb{F}_2$. As an example, over $\mathbb{F}_{2^8}$, Belaïd *et al.* originally manage to instantiate their algorithms at order 2 and 3 respectively and this was later slightly improved to 4 in both cases by Karpman and Roche [KR18]. In this second paper, Belaïd *et al.* also analyse the security of their

schemes thanks to a powerful matrix-based model that they introduce. This model is however not complete for schemes defined over small fields such as $\mathbb{F}_2$; while this was not a limitation for their schemes, it could preclude its direct application to more elementary cases.

On the implementation side, several recent work investigate the efficiency of high-order masking in practice [GR17,BDF+17,JS17,GJRS18,GPSS18]; they show in particular the increasing feasibility of masking block ciphers at quite high order such as 7, and the possibility of masking at very high order such as 31. From a technical point-of-view, these work share the common approach of exploiting bitslicing or vectorisation to amortise the overhead brought by the use of many shares. They also confirm the high cost of randomness generation; for instance, depending on the random number generator performance and the block cipher under consideration Journault and Standaert report that 68–92% of the time is spent generating fresh masks in their 32-share implementations [JS17]. Also, since bitslicing works with operations at the bit level, this strategy requires the masking to be performed over $\mathbb{F}_2$ and these work confirm the importance of high-order masking schemes over this field with low randomness complexity.

1.1 Our contributions

Our work brings two main contributions. We first extend the matrix model of [BBP+17] to be able to prove the security of schemes defined over any finite field, and $\mathbb{F}_2$ in particular; we also extend it to analyse SNI security, whereas it was only formulated in the NI case by Belaïd *et al.*. This in turn allows us to extend a strategy devised in [BBP+16] that efficiently searches for attacks on masking schemes and probabilistically proves their security: we use our results to make this approach complete and thence able to provide full security proofs, and give an efficient implementation of the resulting algorithm. For straight (software) multiplication schemes, the verification performance of our software beats the state-of-the-art maskVerif tool of Barthe *et al.* [BBC+19] by three orders of magnitude.

In a second contribution, we introduce new NI and SNI multiplications with reduced randomness complexity. Our NI scheme is a natural generalisation at any order of a specific scheme at order 4 introduced by Belaïd *et al.* [BBP+16, Algorithm 6], and it uses up to $3d/4$ fewer random masks than their generic improved multiplication ([BBP+16, Algorithm 4]). We verify its security over any field of characteristic two up to order 8 using maskVerif and up to order 11 using our own software, and conjecture that it remains secure at any order. Our SNI schemes are *ad hoc* transformations of our NI ones, and are proven secure up to order 8 using maskVerif and 9 using our software; they significantly improve on the state-of-the-art randomness complexity for orders more than 5: as an example, for the quite relevant order 7, our SNI multiplication only requires 20 fresh random masks compared to 28 for ISW.

1.2 Roadmap

We present the security models and extend the matrix approach from CRYPTO 2017 in Section 2. We then introduce our verification algorithm and discuss its implementation in Sections 3 and 4. We conclude with the definition of our new multiplication schemes and experimental results on their verification in Section 5.

1.3 Notation

We use $\mathbb{K}^{n\times m}$ to denote the ring of matrices of n rows and m columns over the field $\mathbb{K}$. We write $[\![a, b]\!]$ for the set of integers $\{a, \dots, b\}$. Matrices and vectors are named with bold upper- and lower-case variables respectively; $\boldsymbol{I}_n$, $\mathbf{0}_{n\times m}$, $\mathbf{1}_{n\times m}$ always denote the n-dimensional identity matrix and all-zero and all-one $n \times m$ matrices respectively, over any field $\mathbb{K}$.

2 Security models for masking schemes

2.1 Simulability and non-interference

We start by recalling the definitions of the models of non-interference (NI), tight non-interference (TNI) and strong non-interference (SNI), introduced by Barthe *et al.* at CCS 2016 [BBD+16]. Our presentation closely follows the one of Belaïd *et al.* [BBP+17].

Definition 1 (Gadgets). *Let $f : \mathbb{K}^n \to \mathbb{K}^m$, $u, v \in \mathbb{N}$; a* (u, v)-gadget *for the function f is a randomised circuit C such that for every tuple $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{K}^u)^n$ and every set of random coins $\mathcal{R}$, $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m) \leftarrow\!\!\shortmid C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \mathcal{R})$ satisfies:*

$$\left(\sum_{j=1}^{v} \boldsymbol{y}_{1,j}, \ldots, \sum_{j=1}^{v} \boldsymbol{y}_{m,j}\right) = f\left(\sum_{j=1}^{u} \boldsymbol{x}_{1,j}, \ldots, \sum_{j=1}^{u} \boldsymbol{x}_{m,j}\right).$$

One further defines $\boldsymbol{x}_i$ as $\sum_{j=1}^{u} \boldsymbol{x}_{i,j}$, and similarly for $\boldsymbol{y}_i$; $\boldsymbol{x}_{i,j}$ is called the jth share *of x_i.*

In this definition, a randomised circuit C is a directed acyclic graph whose vertices represent arithmetic operation *gates* (addition and multiplication) over $\mathbb{K}$ of arity two, or random gates of arity zero whose outputs are i.i.d. over $\mathbb{K}$ for every execution of the circuit, and recorded in the variable $\mathcal{R}$; the edges of the graph are *wires* that connect the input and output of the gates together so as to describe the full computation of a given function.

A *probe* on a circuit C is a map that for every execution $C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \mathcal{R})$ returns the value propagated on one of the wires of C. One may further distinguish between *external* probes on the output wires or output shares $\boldsymbol{y}_{i,j}$'s of C, and the remaining *internal* probes.

Definition 2 (t-Simulability). *Let C be a (u, v)-gadget for $f : \mathbb{K}^n \to \mathbb{K}^n$, and $\ell, t \in \mathbb{N}$. A set $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ of probes of C is said to be* t-simulable *if $\exists I_1, \ldots, I_n \subseteq [\![1, u]\!]$; $\#I_i \leq t$ and a randomised function $\pi : (\mathbb{K}^t)^n \to \mathbb{K}^\ell$ such that for any fixed $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in (\mathbb{K}^u)^n$, $\{p_1, \ldots, p_\ell\} \sim \{\pi(\{\boldsymbol{x}_{1,i},\, i \in I_1\}, \ldots, \{\boldsymbol{x}_{n,i},\, i \in I_n\})\}$.*

Less formally, a set $\mathcal{P}$ of probes on C is t-simulable if there exists a randomised function that perfectly simulates the distribution of $\{p_1, \ldots, p_\ell\}$ while requiring at most t shares of every input to C. It is important to remark here that a simulation has to be done for every *fixed* input $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, regardless of the fact that one may randomise these inputs across many executions of C.

Thanks to Definition 2, we may now define the following.

Definition 3 (d-Non-interference). *A (u, v)-gadget C for a function over $\mathbb{K}^n$ is* d-non-interfering *(or d-NI) if and only if any set of at most d probes on C is t-simulable, $t \leq d$.*

Definition 4 (d-Tight non-interference). *A (u, v)-gadget C for a function over $\mathbb{K}^n$ is* d-tight-non-interfering *(or d-TNI) if and only if any set of $t \leq d$ probes on C is t-simulable.*

Definition 5 (d-Strong non-interference). *A (u, v)-gadget C for a function over $\mathbb{K}^n$ is* d-strong non-interfering *(or d-SNI) if and only if for every set $\mathcal{P}_1$ of d_1 internal probes and every set $\mathcal{P}_2$ of d_2 external probes such that $d_1 + d_2 \leq d$, then $\mathcal{P}_1 \cup \mathcal{P}_2$ is d_1-simulable.*

It is clear that strong non-interference implies tight non-interference at the same order, which itself implies non-interference. Barthe *et al.* [BBD+16] showed that tight non-interference did not imply strong non-interference, but that the composition of a d-NI gadget with a d-SNI one is d-SNI, while the composition of two d-NI gadgets was not necessarily d-NI. On the other hand they also showed that non-interference and tight non-interference are in fact equivalent, which in proofs allows to select the most convenient notion.

2.2 Matrix model for non-interference

We now recall Theorem 3.5 from Belaïd *et al.* [BBP+17], which defines a powerful matrix model to analyze the (T)NI property of a gadget over a sufficiently large field $\mathbb{K}$ for which all probes are *bilinear*. We then generalise it as Theorem 12 to work with schemes over any finite field (and $\mathbb{F}_2$ in particular), and to also analyse SNI security.

In all of the following, we restrict our interest to gadgets for binary (multiplication) functions $f : \mathbb{K}^2 \to \mathbb{K}$, and the inputs to f (resp. their sharings in a gadget C) will be denoted a and b (resp. $\boldsymbol{a} = (\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{u-1})^t$, $\boldsymbol{b} = (\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{u-1})^t$). We also write the elements of $\mathcal{R}$ as a vector $\boldsymbol{r} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_R)^t$

Definition 6 (Bilinear probe). *A probe p on a $(d+1, v)$-gadget C for a function $f : \mathbb{K}^2 \to \mathbb{K}$ is called* bilinear *iff. it is an affine function in $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, $\boldsymbol{a}_i\boldsymbol{b}_j$, $\boldsymbol{r}_k$; $0 \leq i, j \leq d$, $1 \leq k \leq R$. Equivalently, p is bilinear iff. $\exists \boldsymbol{M} \in \mathbb{K}^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{K}^{d+1}$, $\boldsymbol{\sigma} \in \mathbb{K}^R$ and $\tau \in \mathbb{K}$ s.t. $p = \boldsymbol{a}^t\boldsymbol{M}\boldsymbol{b} + \boldsymbol{a}^t\boldsymbol{\mu} + \boldsymbol{b}^t\boldsymbol{\nu} + \boldsymbol{r}^t\boldsymbol{\sigma} + \tau$.*

Definition 7 (Functional dependence). *An expression* $\mathrm{E}(x_1, \ldots, x_n)$ *is said to* functionally depend *on* x_n *iff.* $\exists\, c_1, \ldots, c_{n-1}$ *s.t. the mapping* $x_n \mapsto \mathrm{E}(c_1, \ldots, c_{n-1}, x_n)$ *is not constant.*

We now introduce the following condition.

Condition 8 ([BBP+17, Condition 3.2]). *A set of bilinear probes* $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ *on a* $(d+1, v)$*-gadget* C *for a function* $f : \mathbb{K}^2 \to \mathbb{K}$ *satisfies Condition 8 iff.* $\exists\, \boldsymbol{\lambda} \in \mathbb{K}^\ell$, $\boldsymbol{M} \in \mathbb{K}^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu} \in \mathbb{K}^{d+1}$, *and* $\tau \in \mathbb{K}$ *s.t.* $\sum_{i=1}^{\ell} \boldsymbol{\lambda}_i p_i = \boldsymbol{a}^t \boldsymbol{M} \boldsymbol{b} + \boldsymbol{a}^t \boldsymbol{\mu} + \boldsymbol{b}^t \boldsymbol{\nu} + \tau$ *and all the rows of the block matrix* $\begin{pmatrix} \boldsymbol{M} & \boldsymbol{\mu} \end{pmatrix}$ *or all the columns of the block matrix* $\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{\nu}^t \end{pmatrix}$ *are non-zero.*

In other words, this condition states that there exists a linear combination of probes of $\mathcal{P}$ that does not functionally depend on any random scalar and that functionally depends on either all of the shares for a or all of the shares for b.

We are now ready to state the following theorem.

Theorem 9 ([BBP+17, Theorem 3.5]). *Let* $\mathcal{P}$ *be a set of bilinear probes on a* $(d+1, v)$*-gadget* C *for a function* $f : \mathbb{K}^2 \to \mathbb{K}$. *If* $\mathcal{P}$ *satisfies Condition 8, then it is not* d*-simulable. Furthermore, if* $\mathcal{P}$ *is not* d*-simulable and* $\#\mathbb{K} > d+1$, *then it satisfies Condition 8.*

The next immediate corollary is more useful in practice.

Corollary 10 ([BBP+17, Corollary 3.7]). *Let* C *be a* $(d+1, v)$*-gadget for a function* $f : \mathbb{K}^2 \to \mathbb{K}$ *for which all probes are bilinear. If* C *is* d*-NI, then there is no set of* d *probes on* C *satisfying Condition 8. Furthermore, if* $\#\mathbb{K} > d+1$ *and there is no set of* d *probes on* C *satisfying Condition 8, then* C *is* d*-NI.*

For the masking schemes of CRYPTO 2017 [BBP+17] the restriction $\#\mathbb{K} > d+1$ is never an issue, as they are defined over large fields; however, this condition means that one cannot directly apply Corollary 10 to prove the security of a scheme over a small field such as $\mathbb{F}_2$.

We now sketch a proof of the second statement of Theorem 9 as a preparation to extending it to any field.

Proof (Theorem 9 right to left, sketch). Let $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ be a set of bilinear probes that is not d-simulable. We call $\boldsymbol{R}$ the block matrix $\begin{pmatrix} \boldsymbol{\sigma}_1 & \cdots & \boldsymbol{\sigma}_\ell \end{pmatrix}$, where $\boldsymbol{\sigma}_i$ denotes as in Definition 6 the vector of random scalars on which p_i depends. Up to a permutation of its rows and columns, the reduced column echelon form $\boldsymbol{R}'$ of $\boldsymbol{R}$ is of the shape $\begin{pmatrix} \boldsymbol{I}_t & \boldsymbol{0}_{t,\ell-t} \\ \boldsymbol{N} & \boldsymbol{0}_t \end{pmatrix}$, where $t < \ell$ is the rank of $\boldsymbol{R}$ and $\boldsymbol{N}$ is arbitrary. If we now consider the formal matrix $\boldsymbol{P} = \begin{pmatrix} p_1 & \cdots & p_\ell \end{pmatrix}^t$ and multiply it by the change-of-basis matrix from $\boldsymbol{R}$ to $\boldsymbol{R}'$, we obtain the matrix $\boldsymbol{P}' = \begin{pmatrix} \boldsymbol{P}'_r & \boldsymbol{P}'_d \end{pmatrix}$ where $\boldsymbol{P}'_r$ represents t linear combinations $\{p'_1, \ldots, p'_t\}$ of probes that each depend on at least one random scalar which does not appear across any of the other linear combinations, and $\boldsymbol{P}'_d$ represents $\ell - t$ linearly independent linear combinations $\mathcal{P}' = \{p'_{t+1}, \ldots, p'_\ell\}$ of probes that do not depend on any random scalar. All of the $\{p'_1, \ldots, p'_t\}$ can then be simulated by independent uniform distributions without requiring the knowledge of any share, and as $\mathcal{P}$ is not d-simulable, $\mathcal{P}'$ cannot be d-simulable either. W.l.o.g., this means that for every share $\boldsymbol{a}_i$, there is at least one linear combination of probe in $\mathcal{P}'$ that depends on it. In other words, the matrix $\boldsymbol{D} = \begin{pmatrix} \boldsymbol{M}'_{t+1} & \boldsymbol{\mu}_{t+1} & \cdots & \boldsymbol{M}'_\ell & \boldsymbol{\mu}_\ell \end{pmatrix}$ that records this dependence has no zero row. We now finally want to show that there is a linear combination $\begin{pmatrix} \boldsymbol{\lambda}_{t+1} & \cdots & \boldsymbol{\lambda}_\ell \end{pmatrix}^t$ of elements of $\mathcal{P}'$ that satisfies Condition 8. This can be done by showing that $\exists\, \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{t+1} & \cdots & \boldsymbol{\Lambda}_\ell \end{pmatrix}^t$ s.t. $\boldsymbol{D}\boldsymbol{\Lambda}$ has no zero row, where the $\boldsymbol{\Lambda}_i$'s are the $(d+2) \times (d+2)$ scalar matrices of multiplication by the $\boldsymbol{\lambda}_i$'s. By the Schwartz-Zippel-DeMillo-Lipton lemma this is always the case as soon as $\#\mathbb{K} > d+1$ [Sch80], and this last step is the only one that depends on $\mathbb{K}$. □

We now wish to extend Theorem 9 and its corollary to any finite field $\mathbb{K}$. We do this using the TNI notion rather than NI, and so first state an appropriate straightforward adaptation of Condition 8:

Condition 11. *A set of bilinear probes* $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ *on a* $(d+1, v)$*-gadget* C *for a function* $f : \mathbb{K}^2 \to \mathbb{K}$ *satisfies Condition 11 iff.* $\exists\, \boldsymbol{\lambda} \in \mathbb{K}^\ell$, $\boldsymbol{M} \in \mathbb{K}^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu} \in \mathbb{K}^{d+1}$, *and* $\tau \in \mathbb{K}$ *s.t.* $\sum_{i=1}^{\ell} \boldsymbol{\lambda}_i p_i = \boldsymbol{a}^t \boldsymbol{M} \boldsymbol{b} + \boldsymbol{a}^t \boldsymbol{\mu} + \boldsymbol{b}^t \boldsymbol{\nu} + \tau$ *and the block matrix* $\begin{pmatrix} \boldsymbol{M} & \boldsymbol{\mu} \end{pmatrix}$ *(resp. the block matrix* $\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{\nu}^t \end{pmatrix}$*) has at least* $\ell + 1$ *non-zero rows (resp. columns).*

In other words, Condition 11 states that the expression $\sum_{i=1}^{\ell} \boldsymbol{\lambda}_i p_i$, which involves ℓ probes, functionally depends on at least $\ell+1$ shares of a or $\ell+1$ shares of b, and hence is a TNI attack. We will then show the following:

Theorem 12. *Let $\mathcal{P}$ be a set of at most d bilinear probes on a $(d+1,v)$-gadget C for a function $f : \mathbb{K}^2 \to \mathbb{K}$. If $\mathcal{P}$, is not d-simulable then $\exists \mathcal{P}' \subseteq \mathcal{P}$ s.t. $\mathcal{P}'$ satisfies Condition 11.*

Corollary 13 (Corollary of Theorems 9 and 12). *Let C be a $(d+1,v)$-gadget C for a function $f : \mathbb{K}^2 \to \mathbb{K}$ for which all probes are bilinear. If C is d-NI, then there is no set of d probes on C satisfying Condition 8. Furthermore, if there is no set of $t \leq d$ probes on C satisfying Condition 11, then C is d-NI.*[1]

The proof of Theorem 12 essentially relies on the following lemmas, conveniently formulated with linear codes:

Lemma 14. *Let $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) be an $[n_1,k]$ (resp. $[n_2,k]$, $n_2 > n_1$) linear code over a finite field $\mathbb{K}$. Let $\boldsymbol{G}_1 \in \mathbb{K}^{k\times n_1}$ and $\boldsymbol{G}_2 \in \mathbb{K}^{k\times n_2}$ be two generator matrices for $\mathcal{C}_1$ and $\mathcal{C}_2$ that have no zero column*[2]*. Then the concatenated code $\mathcal{C}_{1,2}$ of $\mathcal{C}_1$ and $\mathcal{C}_2$ generated by $\boldsymbol{G}_{1,2} := \begin{pmatrix}\boldsymbol{G}_1 & \boldsymbol{G}_2\end{pmatrix}$ has the following property: $\exists \boldsymbol{c} \in \mathcal{C}_{1,2}$ s.t. $\mathrm{wt}_1(\boldsymbol{c}) < \mathrm{wt}_2(\boldsymbol{c})$, where $\mathrm{wt}_1(\cdot)$ (resp. $\mathrm{wt}_2(\cdot)$) denotes the Hamming weight function restricted to the first n_1 (resp. last n_2) coordinates of $\mathcal{C}_{1,2}$.*

One may remark that if $\#\mathbb{K}$ is sufficiently large w.r.t. the parameters of the codes, then by the Schwartz-Zippel-DeMillo-Lipton lemma there exists a word in $\mathcal{C}_{1,2}$ of maximal wt_2 weight, and the conclusion immediately follows; yet this argument doesn't hold over any field.

Lemma 15. *The statement of Lemma 14 still holds if $\mathbb{K}$ is replaced by a matrix ring $\mathbb{K}'^{d\times d}$ and if $\boldsymbol{G}_1$ is defined over the subfield of the scalar matrices of $\mathbb{K}'^{d\times d}$.*

We first recall the following:

Definition 16 (Shortening of a linear code). *Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{K}$ generated by $\boldsymbol{G} \in \mathbb{K}^{k\times n}$. The* shortened code *$\mathcal{C}'$ w.r.t. coordinate $i \in [\![1,n]\!]$ is the subcode made of all codewords of $\mathcal{C}$ that are zero at coordinate i, with this coordinate then being deleted.*

We also give:

Definition 17 (Isolated coordinate). *Let $\boldsymbol{M} \in \mathbb{K}^{m\times n}$. A coordinate $i \in [\![1,n]\!]$ is called* isolated *for the row $\boldsymbol{M}_j$ of $\boldsymbol{M}$, $j \in [\![1,m]\!]$, iff. $\boldsymbol{M}_{j,i} \neq 0$ and $\forall j' \neq j \in [\![1,m]\!]$, $\boldsymbol{M}_{j',i} = 0$.*

And:

Procedure 18. We reuse the notation of the statement of Lemma 14. We *apply Procedure 18 on a row of $\boldsymbol{G}_{1,2}$* by doing the following: denote $\mathcal{I}_1$ (resp. $\mathcal{I}_2$) the (possibly empty) set of isolated coordinates on its first n_1 (resp. last n_2) columns; then if $\#\mathcal{I}_1 \geq \#\mathcal{I}_2$, shorten $\mathcal{C}_{1,2}$ w.r.t. all the coordinates in $\mathcal{I}_1 \cup \mathcal{I}_2$. Practically, this means deleting from $\boldsymbol{G}_{1,2}$ the row being processed and all the columns in $\mathcal{I}_1 \cup \mathcal{I}_2$. This results in a code $\mathcal{C}'_{1,2}$ generated by $\begin{pmatrix}\boldsymbol{G}'_1 & \boldsymbol{G}'_2\end{pmatrix}$ where $\boldsymbol{G}'_1 \in \mathbb{K}^{(k-1)\times n'_1}$ (resp. $\boldsymbol{G}'_2 \in \mathbb{K}^{(k-1)\times n'_2}$) is a submatrix of $\boldsymbol{G}_1$ (resp. $\boldsymbol{G}_2$) and $n'_1 < n_1$, $n'_2 < n_2$, $n'_1 < n'_2$, and none of the columns of $\boldsymbol{G}'_{1,2}$ is zero. One may also remark that since $\boldsymbol{G}'_1$ is of rank $k-1$, we have $k-1 \leq n'_1$.

We are now ready to prove Lemmas 14 and 15.

Proof (Lemma 14). We prove this lemma by induction using Procedure 18.

In a first step, one successively applies Procedure 18 to all the rows of $\boldsymbol{G}_{1,2}$ until either there is no row for which applying the procedure results in a shortening, or the dimension of the shortened code reaches 1.

In the latter case, this means that $\boldsymbol{G}'_{1,2} \in \mathbb{K}^{1\times(n'_1+n'_2)}$ is of full weight $n'_1+n'_2$, $n'_1 < n'_2$. This induces a codeword c of $\mathcal{C}$ s.t. $\mathrm{wt}_1(c) = n'_1$ and $\mathrm{wt}_2(c) = n'_2$, so we are done.

In the former case, one is left with a matrix $\boldsymbol{G}'_{1,2} \in \mathbb{K}^{k'\times(n'_1+n'_2)}$, $k' > 1$. One then computes the reduced row echelon form of $\boldsymbol{G}'_{1,2}$ and applies Procedure 18 again on the resulting matrix. Now either the application of Procedure 18 leads to a shortened code of dimension 1 and then we are done as above, or we are left with a matrix $\boldsymbol{G}''_{1,2} \in \mathbb{K}^{k''\times(n''_1+n''_2)}$ which can be of two forms:

[1] As Condition 11 directly implies an attack, one could also formulate this corollary solely in terms of this condition.

[2] Those can be assumed to exist w.l.o.g., as otherwise one of the codes would in fact have a smaller length.

1. $k'' = n_1''$. Up to permutation of its columns, $\boldsymbol{G}''_{1,2}$ can be written as:

$$\left(\boldsymbol{I}_{n_1''} \,\middle|\, \boldsymbol{I}_{n_1''}\ \boldsymbol{I}_{n_1''}\ *\right),$$

where $*$ is arbitrary. The left $k'' \times n_1''$ block is justified from $\boldsymbol{G}''_{1,2}$ being in reduced row echelon form and having full rank. The right $k'' \times n_2''$ block is justified from the fact that every row of the left block has exactly one isolated coordinate; since no simplification can be done anymore to $\boldsymbol{G}''_{1,2}$ by applying Procedure 18, this means that those rows have at least two isolated coordinates on the right block. This is enough to conclude on the existence of a codeword of $\mathcal{C}$ satisfying the desired property.
Recall that it is not possible to have $k'' > n_1''$ from the last remark in Procedure 18. The only remaining case is then:
2. $k'' < n_1''$. Up to a permutation of its columns, the rank-k'' matrix $\boldsymbol{G}''_{1,2}$ can be written as:

$$\left(\boldsymbol{I}_{k''}\ *_L \,\middle|\, \boldsymbol{I}_{k''}\ \boldsymbol{I}_{k''}\ *_R\right),$$

and it has no zero column (since the elementary row operations are invertible). One then applies Lemma 14 inductively on the code generated by the submatrix $\boldsymbol{G}'''_{1,2} := \left(*_L \,\middle|\, \boldsymbol{I}_{k''}\ *_R\right)$ which is of strictly smaller length. Let $c''' = \boldsymbol{\lambda} \boldsymbol{G}'''_{1,2}$ be a codeword of this latter code that satisfies the desired property, then $\boldsymbol{\lambda} \boldsymbol{G}''_{1,2}$ also satisfies it for $\mathcal{C}_{1,2}$, which concludes the proof.

□

Proof (Lemma 15). The proof simply consists in remarking that all the steps of the proof of Lemma 14 can be carried out in the modified setting of Lemma 15. Mainly:

— Definitions 16 and 17 and Procedure 18 naturally generalise to matrices over rings, and the application of Procedure 18 is unchanged.
— Recall that by induction the left $k' \times n_1'$ submatrix is always of full rank k', which is also the rank of $\boldsymbol{G}'_{1,2}$. Since $\boldsymbol{G}_1$ is defined over scalar matrices, Gauß-Jordan elimination can be computed as if over a field.

□

The proof of Theorem 12 then follows.

Proof (Theorem 12). We start similarly from the proof of Theorem 9, and use the same notation: let $\mathcal{P}'$ be a set of $\ell - t$ linearly independent linear combinations of probes of $\mathcal{P}$ that do not depend on any random scalar, and let $\boldsymbol{D} = \left(\boldsymbol{M}'_{t+1}\ \boldsymbol{\mu}_{t+1} \cdots \boldsymbol{M}'_{\ell}\ \boldsymbol{\mu}_{\ell}\right)$ be the matrix that records the dependence of these probes on every share $\boldsymbol{a}_i$.We will show that $\exists \mathcal{P}'' \subseteq \mathcal{P}$ that satisfies Condition 11. To do this, we introduce two new indicator matrices:

— Let $\boldsymbol{\Pi} \in \mathbb{K}^{(d+2)\times(d+2)^{(\ell-t)\times\ell}}$ be s.t. for every $p' \in \mathcal{P}'$ it records in its rows its dependence on the probes of $\mathcal{P}$ as scalar matrices; that is, $\boldsymbol{\Pi}$ is s.t. $p'_i = \sum_{j=1}^{\ell} \boldsymbol{\pi}_{i,j} p_j$ where $\boldsymbol{\pi}_{i,j}$ is the scalar on the diagonal of the scalar matrix $\boldsymbol{\Pi}_{i,j}$. W.l.o.g., we may assume that every probe of $\mathcal{P}$ appears at least once in a linear combination of $\mathcal{P}'$, otherwise it is simply discarded, so $\boldsymbol{\Pi}$ has no zero column.
— Let $\boldsymbol{\Delta} \in \mathbb{K}^{(d+2)\times(d+2)^{(\ell-t)\times(d+1)}}$ be the matrix that for every $p' \in \mathcal{P}'$ records in its rows its dependence on the shares $\boldsymbol{a}_i$s; that is if the bilinear probe p'_i can be written as $p'_i = \boldsymbol{a}^t \boldsymbol{M}' \boldsymbol{b} + \boldsymbol{a}^t \boldsymbol{\mu}' + \boldsymbol{b}^t \boldsymbol{\nu}' + \tau'$, then $\boldsymbol{\Delta}_{i,j}$ is set to the diagonal matrix of the j^{th} row of $\left(\boldsymbol{M}'\ \boldsymbol{\mu}'\right)$. Note that since by assumption $\boldsymbol{D}$ has no zero row, $\boldsymbol{\Delta}$ has no zero *column*.

Now we invoke Lemma 15 with $\boldsymbol{\Pi}$ as $\boldsymbol{G}_1$ and $\boldsymbol{\Delta}$ as $\boldsymbol{G}_2$ the generator matrices for the concatenated code $\mathcal{C}_{1,2}$. Let $\boldsymbol{c} \in \mathcal{C}_{1,2}$ be a codeword that satisfies $\mathrm{wt}_1(\boldsymbol{c}) < \mathrm{wt}_2(\boldsymbol{c})$; this translates to a linear combination of $\ell'' := \mathrm{wt}_1(\boldsymbol{c})$ probes of $\mathcal{P}'' \subseteq \mathcal{P}$ that (as linear combinations of elements of $\mathcal{P}'$) does not depend on any randomness and s.t. the associated matrix $\left(\boldsymbol{M}''\ \boldsymbol{\mu}''\right)$ has $\mathrm{wt}_2(\boldsymbol{c}) \geq \ell'' + 1$ non-zero rows (by applying the inverse transformation from $\boldsymbol{\Delta}$ to $\boldsymbol{D}$), hence $\mathcal{P}''$ satisfies Condition 11.

□

Finally, the proof of Corollary 13 is immediate from Theorems 9 and 12.

2.3 Matrix model for strong non-interference

We now wish to adapt the approach of Theorems 9 and 12 to be able to prove that a scheme is SNI. This is in fact quite straightforward, and it mostly consists in defining a suitable variant of Condition 11 and in applying Lemma 15 to well-chosen matrices, to show again that there is a subset of probes that satisfies the condition whenever there is an attack.

Condition 19. *A set of $\ell = \ell_1 + \ell_2$ bilinear probes $\mathcal{P} = \{p_1, \ldots, p_\ell\}$ on a $(d+1, v)$-gadget C for a function $f : \mathbb{K}^2 \to \mathbb{K}$, of which ℓ_1 are internal, satisfies Condition 19 iff. $\exists \boldsymbol{\lambda} \in \mathbb{K}^\ell$, $\boldsymbol{M} \in \mathbb{K}^{(d+1)\times(d+1)}$, $\boldsymbol{\mu}, \boldsymbol{\nu} \in \mathbb{K}^{d+1}$, and $\tau \in \mathbb{K}$ s.t. $\sum_{i=1}^{\ell} \boldsymbol{\lambda}_i p_i = \boldsymbol{a}^t \boldsymbol{M} \boldsymbol{b} + \boldsymbol{a}^t \boldsymbol{\mu} + \boldsymbol{b}^t \boldsymbol{\nu} + \tau$ and the block matrix $\begin{pmatrix} \boldsymbol{M} & \boldsymbol{\mu} \end{pmatrix}$ (resp. the block matrix $\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{\nu}^t \end{pmatrix}$) has at least $\ell_1 + 1$ non-zero rows (resp. columns).*

Theorem 20. *Let $\mathcal{P}$ be a set of at most d bilinear probes on a $(d+1, v)$-gadget C for a function $f : \mathbb{K}^2 \to \mathbb{K}$, of which ℓ_1 are internal. If $\mathcal{P}$ is not ℓ_1-simulable then $\exists \mathcal{P}' \subseteq \mathcal{P}$ s.t. $\mathcal{P}'$ satisfies Condition 19.*

Proof. We reuse the notation of Theorems 9 and 12. The proof is essentially the same as the one of Theorem 12, except that we only account of internal probes in $\boldsymbol{\Pi}$. Let $\mathcal{P}'$ be a set of $\ell - t$ linearly independent linear combinations of probes of $\mathcal{P}$ that do not depend on any random scalar, and let $\boldsymbol{D} = \begin{pmatrix} \boldsymbol{M}'_{t+1} & \boldsymbol{\mu}_{t+1} & \cdots & \boldsymbol{M}'_{\ell} & \boldsymbol{\mu}_{\ell} \end{pmatrix}$ be the matrix that records the dependence of these probes on every share $\boldsymbol{a}_i$. From the assumption that $\mathcal{P}$ is not ℓ_1-simulable, we have that w.l.o.g., $\boldsymbol{D}$ has at least $\ell_1 + 1$ non-zero rows.

- Let $\boldsymbol{\Pi} \in \mathbb{K}^{(d+2)\times(d+2)^{(\ell-t)\times\ell_1}}$ be s.t. for every $p' \in \mathcal{P}'$ it records in its rows its dependence on the ℓ_1 *internal* probes (w.l.o.g. $\{p_1, \ldots, p_{\ell_1}\}$) of $\mathcal{P}$ as scalar matrices; that is, $\boldsymbol{\Pi}$ is s.t. $p'_i = \sum_{j=1}^{\ell_1} \boldsymbol{\pi}_{i,j} p_j + \sum_{j=\ell_1+1}^{\ell} \alpha_j p_j$, where $\boldsymbol{\pi}_{i,j}$ is the scalar on the diagonal of the scalar matrix $\boldsymbol{\Pi}_{i,j}$ and the α_js are unimportant. W.l.o.g., we may assume that every internal probe of $\mathcal{P}$ appears at least once in a linear combination of $\mathcal{P}'$, otherwise it is simply discarded, so $\boldsymbol{\Pi}$ has no zero column.
- Let $\boldsymbol{\Delta} \in \mathbb{K}^{(d+2)\times(d+2)^{(\ell-t)\times d'}}$ be the matrix that for every $p' \in \mathcal{P}'$ records in its rows its dependence on the shares $\boldsymbol{a}_i$s. If a row of $\boldsymbol{D}$ is all zero, the corresponding column is not included in $\boldsymbol{\Delta}$, and since $\boldsymbol{D}$ has at least $\ell_1 + 1$ non-zero rows, $\boldsymbol{\Delta}$ has at least $d' \geq \ell_1 + 1$ columns none of which are zero.

Now we invoke Lemma 15 with $\boldsymbol{\Pi}$ as $\boldsymbol{G}_1$ and $\boldsymbol{\Delta}$ as $\boldsymbol{G}_2$ the generator matrices for the concatenated code $\mathcal{C}_{1,2}$. Let $\boldsymbol{c} \in \mathcal{C}_{1,2}$ be a codeword that satisfies $\mathrm{wt}_1(\boldsymbol{c}) < \mathrm{wt}_2(\boldsymbol{c})$; this translates to a linear combination of $\ell'' := \mathrm{wt}_1(\boldsymbol{c})$ internal probes to which one can add a linear combination of up to ℓ_2 external probes s.t. it does not depend on any randomness and the associated matrix $\begin{pmatrix} \boldsymbol{M}'' & \boldsymbol{\mu}'' \end{pmatrix}$ has $\mathrm{wt}_2(\boldsymbol{c}) \geq \ell'' + 1$ non-zero rows. The set $\mathcal{P}'' \subseteq \mathcal{P}$ of these internal and external probes thus satisfies Condition 19. □

And we then have the immediate corollary:

Corollary 21. *Let C be a $(d+1, v)$-gadget for a function $f : \mathbb{K}^2 \to \mathbb{K}$ for which all probes are bilinear. If there is no set of $t \leq d$ probes on C satisfying Condition 19, then C is d-SNI.*

2.4 Security of binary schemes over finite fields of characteristic two

Let C be a d-NI or SNI gadget for a function defined over $\mathbb{F}_2$; a natural question is whether its security is preserved if it is lifted to an extension $\mathbb{F}_{2^n}$. Indeed, the probes available to the adversary are the same in the two cases, but the latter offers more possible linear combinations $\sum_{i=1}^{\ell} \boldsymbol{\lambda}_i p_i$, since the $\boldsymbol{\lambda}_i$s are no longer restricted to $\{0, 1\}$. We answer this question positively, and give a simple proof based on Theorems 12 and 20.

Theorem 22. *Let C be a d-NI (resp. d-SNI) gadget for a function $f : \mathbb{F}_2^2 \to \mathbb{F}_2$, then for any n, the natural lifting $\widehat{C}$ of C to $\widehat{f} : \mathbb{F}_{2^n}^2 \to \mathbb{F}_{2^n}$ is also d-NI (resp. d-SNI).*

Proof. We only prove the d-NI case, the d-SNI one being similar. From Corollary 13, it is sufficient to show that if $\nexists \mathcal{P}$ for C that satisfies Condition 11, then the same holds for $\widehat{C}$. We do this by showing the following contrapositive: if a set of probes $\mathcal{P}$ is not d-simulable for $\widehat{C}$, then it is not d-simulable either for C.

From the proofs of Theorems 9 and 12, if $\mathcal{P}$ is not d-simulable for $\widehat{C}$, then there is a matrix $\widehat{\boldsymbol{D}}$ that leads to the existence of $\mathcal{P}'$ s.t. Condition 11 is satisfied. All we need to do is showing that a similar matrix $\boldsymbol{D}$ can also be found for C. Since C is defined over $\mathbb{F}_2$, the matrices $\boldsymbol{R}$ and $\boldsymbol{P}$, and thence $\widehat{\boldsymbol{R}}$ and $\widehat{\boldsymbol{P}}$ have all their coefficients in $\{0, 1\}$. As 1 is its own inverse, the change-of-basis matrix from $\widehat{\boldsymbol{R}}$ to $\widehat{\boldsymbol{R}}'$ is also binary; equivalently, this means that the Gauß-Jordan elimination of $\widehat{\boldsymbol{R}}$ can be done in the subfield $\mathbb{F}_2$. Thus one only has to take $\boldsymbol{D} = \widehat{\boldsymbol{D}}$ to satisfy Condition 11 on C. □

This result is quite useful as it means that the security of a binary scheme only needs to be proven once in $\mathbb{F}_2$, even if it is eventually used in one or several extension fields. Proceeding thusly is in particular beneficial in terms of verification performance, since working over $\mathbb{F}_2$ limits the number of linear combinations to consider and may lead to some specific optimisations (cf. *e.g.* Sections 3 and 4).

Remark. This result somewhat seems to be folklore and (a slight variant) is in fact already implicitly used by Barthe *et al.* in their masking compiler [BBD+15], since they use gadgets defined over an arbitrary structure $(\mathbb{K}, 0, 1, \oplus, \ominus, \odot)$. However, we could not find a proof of the exact statement given above, which is the one we need in our case to justify the correctness of the algorithm of the next section.

3 An algorithm for checking non-interference

In this section, we present a new efficient algorithm to check if a scheme is (strong) non-interfering. This algorithm is a modification of the one presented by Belaïd *et al.* at EUROCRYPT 2016 [BBP+16, Section 8], and its correctness crucially relies on Theorems 12 and 20; it thus only applies to schemes for which all probes are bilinear, but this is not a hard restriction in practice.

In all of the following we assume that the field $\mathbb{K}$ over which the scheme is defined is equal to $\mathbb{F}_2$, which means that we will simultaneously assess its security in that field and all its extensions (cf. Section 2.4). Some discussion of implementation in the NI case for schemes natively defined over larger fields (meaning that shares or random masks may be multiplied by constants not in $\{0, 1\}$) for which the new Theorem 12 is not needed can be found in [KR18].

We start by introducing some vocabulary and by recalling the algorithm from Belaïd *et al.*.

Definition 23 (Elementary probes). *A probe p is called* elementary *if it is of the form $p = \boldsymbol{a}_i \boldsymbol{b}_j$ (elementary* deterministic *probe) or $p = \boldsymbol{r}_i$ (elementary* random *probe).*

Definition 24 (Shares indicator matrix). *Let p be a bilinear probe. We call* shares indicator matrix *and write $\boldsymbol{M}_p$ the matrix $\boldsymbol{M}$ from Definition 6.*

Definition 25 (Randomness indicator matrix). *Let p be a bilinear probe. We call* randomness indicator matrix *and write $\boldsymbol{\sigma}_p$ the column matrix $\boldsymbol{\sigma}$ from Definition 6.*

3.1 The algorithm from EUROCRYPT 2016

At EUROCRYPT 2016, Belaïd *et al.* presented an efficient probabilistic algorithm to find potential attacks against the d-privacy notion[3] for masking schemes for the multiplication over $\mathbb{F}_2$. By running the algorithm many times and not detecting any attack, one can also establish the security of a scheme up to some probability, but deriving a deterministic counterpart is less trivial. This algorithm works as follows.

Consider a scheme on which all possible probes $\mathcal{P}$ are bilinear, and let $\boldsymbol{H}_{\mathcal{P}} := \left(\boldsymbol{\sigma}_p\right), p \in \mathcal{P}$ be the block matrix constructed from the all the corresponding randomness indicator matrices. The algorithm of [BBP+16, Section 8] starts by finding a set of fewer than d probes whose sum[4] does not depend on any randomness. That is to say, it is looking for a vector $\boldsymbol{x}$ such that $\boldsymbol{H}_{\mathcal{P}} \cdot \boldsymbol{x} = \boldsymbol{0}$ and $\mathrm{wt}(\boldsymbol{x}) \leq d$. This can be immediately reformulated as a coding problem, as one is in fact searching for a codeword of weight less than d in the dual code of $\boldsymbol{H}_{\mathcal{P}}$. This search can then be performed using any information set decoding algorithm, and Belaïd *et al.* used the original one of Prange [Pra62].[5] Once such a set has been found, it is tested against [BBP+16, Condition 2] (which is similar to Condition 8) to determine if it is a valid attack against the d-NI notion, and [BBP+16, Condition 1] to determine if it is an attack for d-privacy. This procedure is then repeated until an attack is found or one has gained sufficient confidence in the security of the scheme.

3.2 Dimension reduction

Before describing our algorithm itself, we discuss a few ways to significantly decrease the size of the eventual enumeration space.

[3] It can also be trivially modified to check attack against NI security.
[4] That is, the only non-trivial linear combination over $\mathbb{F}_2$ that depends on all the elements of the set.
[5] One may remark that since information set decoding relies on Gaussian elimination, the cost of one step of the algorithm of Belaïd *et al.* increases more than linearly in the size of $\mathcal{P}$.

Removing elementary deterministic probes. To make the above procedure more efficient, an important observation made by Belaïd *et al.* is that if the sum of every probe of a given set does not functionally depend on some $\boldsymbol{a}_i$ or $\boldsymbol{b}_j$, it is always possible to make it so by adding a corresponding elementary probe $\boldsymbol{a}_i\boldsymbol{b}_j$. This can be used to check, say, d-NI security by simply comparing the number of missing $\boldsymbol{a}_i$ or $\boldsymbol{b}_j$ to $d - \mathrm{wt}(\boldsymbol{x})$. This allows to reduce the number of probes that one has to include in $\mathcal{P}$ (and thus the dimension of $\boldsymbol{H}_{\mathcal{P}}$), making the algorithm more efficient.

Removing elementary random probes. The preceding remark can be extended to random probes as well: if the sum of every probe of a given set functionally depends on some $\boldsymbol{r}_i$, it is always possible to make it not so by adding the corresponding elementary probes. We thus also remove such probes from the input set of our algorithm, which no longer contains any elementary probe at all.[6]

Removing redundant probes. To further reduce the size of the space to explore during the verification, we will filter out some probes from the set $\mathcal{P}$ that can always be replaced by "better" ones. To do this while ensuring the correctness of our verification algorithm, we first define the following:

Definition 26 (Equipotent sets). *Let $\mathcal{P} := \cup_{k=0}^{v} \mathcal{P}_k$ and $\mathcal{P}' := \cup_{k=0}^{v} \mathcal{P}'_k$ be two sets of probes on a $(d+1, v)$-gadget C for a function $f : \mathbb{F}_2^2 \to \mathbb{F}_2$ for which all probes are bilinear, where $\mathcal{P}_k$ (resp. $\mathcal{P}'_k$) denotes the probes on the wires of C that are connected to the output share $\boldsymbol{c}_k$. Then $\mathcal{P}$ and $\mathcal{P}'$ are called* equipotent *iff.:*

- $\#\mathcal{P}' \leq \#\mathcal{P}$
- $\forall k, \forall \boldsymbol{\lambda} \in \mathbb{K}^{\#\mathcal{P}_k}, \exists \boldsymbol{\lambda}' \in \mathbb{K}^{\#\mathcal{P}'_k}$, $\mathrm{wt}(\boldsymbol{\lambda}') \leq \mathrm{wt}(\boldsymbol{\lambda})$ *for which, reusing the notation of Definition 6, $\sum_{p_i \in \mathcal{P}} \boldsymbol{\lambda}_i p_i$ and $\sum_{p'_i \in \mathcal{P}'} \boldsymbol{\lambda}'_i p'_i$ are such that:*
  - $\boldsymbol{\sigma} = \boldsymbol{\sigma}'$
  - $\mathrm{supp}(\boldsymbol{M}) \subseteq \mathrm{supp}(\boldsymbol{M}')$, $\mathrm{supp}(\boldsymbol{\mu}) \subseteq \mathrm{supp}(\boldsymbol{\mu}')$, $\mathrm{supp}(\boldsymbol{\nu}) \subseteq \mathrm{supp}(\boldsymbol{\nu}')$, *where by $\mathrm{supp}(\boldsymbol{A})$ we denote the set of non-zero coefficients of $\boldsymbol{A}$.*

Lemma 27. *If two linear combinations of probes $\sum \boldsymbol{\lambda}_i p_i$ and $\sum \boldsymbol{\lambda}'_i p'_i$ functionally depend on disjoint sets of elementary probes and shares $\boldsymbol{a}_i\boldsymbol{b}_j$, $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$, then their sum functionally depends on the union of those sets.*

Proof. Immediate, since using the notation of Definition 6, the supports of $\boldsymbol{M}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu}$ are disjoint from the ones of $\boldsymbol{M}'$, $\boldsymbol{\mu}'$, $\boldsymbol{\nu}'$. □

Proposition 28. *Let $\mathcal{P}$ and $\mathcal{P}'$ be two equipotent sets of probes on a $(d+1, v$-gadget C for a function $f : \mathbb{F}_2^2 \to \mathbb{F}_2$ for which all probes are bilinear and for which all output shares functionally depend on pairwise disjoint sets of elementary probes and shares $\boldsymbol{a}_i\boldsymbol{b}_j$, $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$. Then if $\mathcal{Q} \subseteq \mathcal{P}$ satisfies Condition 11, $\exists\, \mathcal{Q}' \subseteq \mathcal{P}'$, $\#\mathcal{Q}' \leq \#\mathcal{Q}$ that also satisfies Condition 11.*

Proof. Let us write $\mathcal{Q}$ as $\cup_{k=0}^{v} \mathcal{Q}_k$ (resp. $\mathcal{Q}'$ as $\cup_{k=0}^{v} \mathcal{Q}'_k$) where $\mathcal{Q}_k$ (resp. $\mathcal{Q}'_k$) denotes the probes on the wires of C that are connected to the output share $\boldsymbol{c}_k$. Let $\sum_{p_i \in \mathcal{Q}} \boldsymbol{\lambda}_i p_i$ denote one linear combination of elements of $\mathcal{Q}$ whose existence is guaranteed by its satisfying Condition 11, which we rewrite as: $\sum_k \sum_{p_i \in \mathcal{Q}_k} \boldsymbol{\lambda}_i^{(k)} p_i$. For each $\boldsymbol{\lambda}^{(k)}$, let $\boldsymbol{\lambda}'^{(k)}$ be the coefficients for one of the linear combination of elements of $\mathcal{Q}'_k$ whose existence is guaranteed by $\mathcal{Q}$ and $\mathcal{Q}'$ being equipotent. Then by applying Lemma 27 to each of $\sum_k \sum_{p_i \in \mathcal{Q}_k} \boldsymbol{\lambda}_i^{(k)} p_i$ and $\sum_k \sum_{p_i \in \mathcal{Q}'_k} \boldsymbol{\lambda}_i'^{(k)} p_i$, it follows that the latter does not functionally depend on any elementary random probe $\boldsymbol{r}_i$, and the elementary deterministic probes and shares on which it functionally depends is a superset of the ones on which depends the former; thus $\mathcal{Q}'$ satisfies Condition 11. □

We will see in Section 5 how Proposition 28 can be used in practice to significantly improve verification performance. The nature of the probes that can be removed of course depends on the scheme under consideration, and we will later detail how to do this for our gadgets.

3.3 A new algorithm based on enumeration

We now describe a new algorithm based on a partial enumeration of $\wp(\mathcal{P})$. The correctness relies on Corollaries 13 and 21, and dimension reduction is used to further decrease the size of the set $\mathcal{P}$. We discuss our implementation of this algorithm in Section 4.

[6] Note that this means that one would not detect the existence of an attack that would use *only* elementary probes. However, it is easy to see from their definitions that ℓ such probes functionally depend on at most ℓ shares, and so can never lead to a non-trivial attack.

Checking a scheme for non-interference. We now state the following:

Proposition 29. *Let C be a $(d+1, v)$-gadget for a function $f : \mathbb{F}_2^2 \to \mathbb{F}_2$ for which all probes are bilinear, and $\mathcal{Q}_0$ be a set of n_0 non-elementary probes on C that functionally depends on n_a shares $\boldsymbol{a}_i$s, n_b shares $\boldsymbol{b}_j$s, and n_r random scalars $\boldsymbol{r}_i$s. Let $\mathcal{Q}_1$ be one of the smallest sets of elementary probes needed to complete $\mathcal{Q}_0$ such that $\mathcal{Q}_0 \cup \mathcal{Q}_1$ satisfies Condition 11 and functionally depends on all the $\boldsymbol{a}_i$s or all the $\boldsymbol{b}_i$s.*[7] *Then $n_1 := \#\mathcal{Q}_1 = n_r + (d + 1 - \max(n_a, n_b))$.*

Proof. An elementary probe functionally depends on either one $\boldsymbol{r}_i$ or one $\boldsymbol{a}_i$ and one $\boldsymbol{b}_j$, but not both. Thus, the minimum number of elementary probes needed to cancel every $\boldsymbol{r}_i$ and to add the $d+1-n_a$ (resp. $d+1-n_b$) missing $\boldsymbol{a}_i$s (resp. $\boldsymbol{b}_j$s) in $\mathcal{Q}_0$ is $n_r+(d+1-n_a)$ (resp. $n_r+(d+1-n_b)$). Thus, $\#\mathcal{Q}_1 = \min(n_r + d + 1 - n_a, n_r + d + 1 - n_b) = n_r + d + 1 - \max(n_a, n_b)$. □

This proposition can then be used in a straightforward way to check if a scheme is d-NI. To do so, one simply has to enumerate every set $\mathcal{Q}_0 \in \wp(\mathcal{P}')$ of d non-elementary probes or fewer and to check if $n_0 + n_1 \leq d$. By Corollary 13, if no such set $\mathcal{Q}_0$ can be completed as in Proposition 29 and still contain fewer than d probes, then the scheme is d-NI.

Checking a scheme for strong non-interference. We proceed as above.

Proposition 30. *Let C be a $(d+1, v)$-gadget for a function $f : \mathbb{F}_2^2 \to \mathbb{F}_2$ for which all probes are bilinear, and $\mathcal{Q}_0$ be a set of n_0 non-elementary probes on C that functionally depends on n_a shares $\boldsymbol{a}_i$s, n_b shares $\boldsymbol{b}_j$s, and n_r random scalars $\boldsymbol{r}_i$s. Let n_I denote the number of internal probes in $\mathcal{Q}_0$. Then there is a set $\mathcal{Q}_1$ of n_r elementary random probes such that $\mathcal{Q}_0 \cup \mathcal{Q}_1$ satisfies Condition 19 iff. $\max(n_a, n_b) > n_I + n_r$.*

Proof. Recall that all elementary probes are internal. If $\mathcal{Q}_0$ does not satisfy Condition 19, then adding an elementary deterministic probe increases by at most one the number of non-zero rows, while increasing by one the total number of probes, so this completed set does not satisfy $\mathcal{Q}_0$ either. It is thus enough to only consider random probes in $\mathcal{Q}_1$.

For $\mathcal{Q} = \mathcal{Q}_0 \cup \mathcal{Q}_1$ to satisfy Condition 19, it is necessary to cancel all the potential randomness $\boldsymbol{r}_i$s on which $\mathcal{Q}_0$ depends; so $\mathcal{Q}_1$ must be the (possibly empty) set of the n_r corresponding elementary random probes. Now $\mathcal{Q}$ contains $n_I + n_r$ internal probes and it functionally depends on n_a $\boldsymbol{a}_i$s and n_b $\boldsymbol{b}_j$s. Thus it satisfies Condition 19 iff. $\max(n_a, n_b) > n_I + n_r$. □

This proposition can then be used in a straightforward way to check if a scheme is d-SNI. To do so, one simply has to enumerate every set $\mathcal{Q}_0 \in \wp(\mathcal{P}')$ of d non-elementary probes or fewer and to check if $\max(n_a, n_b) > n_I + n_r$ and $n_0 + n_r \leq d$. If no such set satisfying this condition is found, then the scheme is d-SNI by Corollary 21.

4 Implementation

We now describe an efficient `C` implementation of the algorithm of the previous section over $\mathbb{F}_2$ that we use to prove the security of our schemes of Section 5 up to order $d = 11$ over any field of characteristic two (see Section 5.5 for details). Our software is publicly available at: https://github.com/NicsTr/binary_masking.

4.1 Data structures and vectorisation

To evaluate if a set of probes $\mathcal{P}$ may lead to an attack, it is convenient to define the following:

Definition 31 (Attack matrix). *The* attack matrix $\boldsymbol{A}_{\mathcal{P}}$ *of a set of probes $\mathcal{P}$ is defined as the sum of the share indicator matrices of the probes in $\mathcal{P}$:*

$$\boldsymbol{A}_{\mathcal{P}} = \sum_{p \in \mathcal{P}} \boldsymbol{M}_p.$$

Definition 32 (Noise matrix). *The* noise matrix $\boldsymbol{B}_{\mathcal{P}}$ *of a set of probes $\mathcal{P}$ is defined as the sum of the randomness indicator matrices of the probes in $\mathcal{P}$:*

$$\boldsymbol{B}_{\mathcal{P}} = \sum_{p \in \mathcal{P}} \boldsymbol{\sigma}_p.$$

[7] This additional constraint is not in itself necessary, but it simplifies the overall algorithm.

One can then simply compute the quantities n_a, n_b and n_r needed in Propositions 29 and 30 as the number of non-zero rows or columns of these two matrices. To analyse a given scheme, one then just has to provide a full description of $\boldsymbol{M}_p$ and $\boldsymbol{\sigma}_p$ for every non-elementary probe. Additionally, since Proposition 30 requires to compute the number of internal probes n_I in a set, those are described separately.

We inline all data structures and store them in either standard or vector registers. $\boldsymbol{A}_{\mathcal{P}}$ is stored twice, once row-wise and once column-wise, in order to avoid the otherwise costly transposition needed to compute both its row and its column "Hamming weight". For schemes at order $d \leq 15$, each row or column fits within a 16-bit words leading to a quite efficient vectorised Hamming weight computation, as shown in Figure 1. We also provide a slower implementation for schemes at higher order; in this case actually proving the security with our algorithm is intractable due to the combinatorial explosion of the number of sets to consider, yet a partial run may still be able to detect attacks, in the fashion of the original algorithm from EUROCRYPT 2016.

```
int popcount256_16(__m256i v)
{
        return __builtin_popcountl(_mm256_cmpgt_epi16_mask(v, _mm256_setzero_si256()));
}
```

Fig. 1: Hamming weight computation of a vector of dimension 16 over 16-bit words using AVX512VL and AVX512BW; an only slightly less efficient alternative can be used with only AVX2.

4.2 Amortised enumeration & parallelisation

Recall that to prove the security of a scheme at order d, the algorithm of Section 3 requires to enumerate all the $\sum_{i=1}^{d} \binom{n}{i}$ subsets of an appropriately filtered set of probes $\mathcal{P}$ of size n. For a subset $\mathcal{P}' \subseteq \mathcal{P}$ of size ℓ, a naïve approach in computing, say, $\boldsymbol{A}_{\mathcal{P}'}$ would need $\ell - 1$ additions for every such $\mathcal{P}'$. However, a well-known optimisation for such an enumeration is instead to go through all the subsets of a fixed weight in a way that ensures that two consecutive sets $\mathcal{P}'$ and $\mathcal{P}''$ only differ by two elements. One can then compute, say, $\boldsymbol{A}_{\mathcal{P}''}$ efficiently by updating $\boldsymbol{A}_{\mathcal{P}'}$ with one addition and one subtraction. We do this in our implementation by using a so-called "revolving-door algorithm", see *e.g.* [Knu11, Algorithm R], for the Nijenhuis-Wilf-Tang-Liu "combination Gray code"[NW78,LT73].

The enumeration can also be easily parallelised, and the main challenge is to couple it with the above amortised approach. This can in fact be done quite efficiently, as the combination Gray code that we use possesses an efficient *unranking* map from the integers to arbitrary configurations [Wal]. One can then easily divide a full enumeration of a total of n combinations into j jobs by starting each of them independently at one of the configurations given by the unranking of $i \times n/j$, $i \in [\![0, j[\![$.

4.3 Integration with the tool from EUROCRYPT 2016

We reuse the tool of [BBP+16][8] and complete it to convert a high-level description of a masking scheme into the `C` description of its probes' indicator matrices that we need.

This tool already provides a `sage` parser for a simple text format, where each line corresponds to an output share and consists in a series of space-separated elementary probes "`sij`" (standing for $\boldsymbol{a}_i\boldsymbol{b}_j$) or `ri` (standing for $\boldsymbol{r}_i$). The scheduling of the operations needed to compute the output shares is important, as it determines the probes available to the adversary. In that respect, the parser uses by default an implicit left-to-right scheduling. As an example the scheme whose output shares are defined as:

$$\begin{aligned}
\boldsymbol{c}_0 &= ((((\boldsymbol{a}_0\boldsymbol{b}_0 \oplus \boldsymbol{r}_0) \oplus \boldsymbol{a}_0\boldsymbol{b}_1) \oplus \boldsymbol{a}_1\boldsymbol{b}_0) \oplus \boldsymbol{r}_1) \\
\boldsymbol{c}_1 &= ((((\boldsymbol{a}_1\boldsymbol{b}_1 \oplus \boldsymbol{r}_1) \oplus \boldsymbol{a}_1\boldsymbol{b}_2) \oplus \boldsymbol{a}_2\boldsymbol{b}_1) \oplus \boldsymbol{r}_2) \\
\boldsymbol{c}_2 &= ((((\boldsymbol{a}_2\boldsymbol{b}_2 \oplus \boldsymbol{r}_2) \oplus \boldsymbol{a}_2\boldsymbol{b}_0) \oplus \boldsymbol{a}_0\boldsymbol{b}_2) \oplus \boldsymbol{r}_0)
\end{aligned}$$

[8] Available at https://github.com/fabrice102/private_multiplication.

is described by the file:

```
s00 r00 s01 s10 r01
s11 r01 s12 s21 r02
s22 r02 s20 s02 r00
```

5 New private multiplication schemes

In this section we present new algorithms for private multiplication over fields of characteristic two. We first introduce a new family of NI schemes in Section 5.1 and their extensions to SNI security in Section 5.2. This is followed by a discussion about their possible applications and experimental verification results.

5.1 Non-interfering schemes

We propose a new family of $(d+1, d+1)$-gadgets for multiplication that we have experimentally verified to be d-NI for $d \in [\![1, 11]\!]$ (see Section 5.5 for more details), and that we conjecture to be d-NI for any d. This family is designed as a generalisation of the specific 4-NI scheme given in [BBP+16, Algorithm 6].

We start with a convenient notation:

Definition 33 (Pair of shares).
Let $(\boldsymbol{a}_i \boldsymbol{b}_j)$, $i, j \in [\![0, d]\!]$ be the input shares of a $(d+1, v)$ gadget. We define $\hat{\boldsymbol{\alpha}}_{i,j}$ as:

$$\hat{\boldsymbol{\alpha}}_{i,j} = \begin{cases} \boldsymbol{a}_i \boldsymbol{b}_j & \text{if } i = j \\ \boldsymbol{a}_i \boldsymbol{b}_j + \boldsymbol{a}_j \boldsymbol{b}_i & \text{otherwise} \end{cases}$$

Our family of schemes is then defined by the following Algorithm 1, of which we provide an implementation at https://github.com/NicsTr/binary_masking.

Periodicity. By construction, the shape of the schemes of Algorithm 1 exhibits a period of four in the order d. For a given d, one has to add $(d+1)^2$ different $\boldsymbol{a}_i \boldsymbol{b}_j$ terms. Following our algorithm, one first includes the $(d+1)$ shares of the form $\boldsymbol{a}_i \boldsymbol{b}_i$ each in one output share, and all of the following input shares are coming by two; one is thus left with $((d+1)^2 - (d+1))/2 = d(d+1)/2$ pairs. Those remaining pairs as well fresh random values are then added to the $d+1$ outputs $\boldsymbol{c}_i$ in sequence. Since each random mask is included exactly twice, the shape of the scheme is then determined by $d(d+1)/2 \mod 2(d+1)$, that is to say by $d \mod 4$.

Complexity. We give explicit formulas for the sum, product, and randomness complexity of Algorithm 1 in Table 1, and compare it with the generic scheme of Belaïd *et al.* from EUROCRYPT 2016.[9] The worst case randomness complexity of our scheme occurs when $d \equiv 1 \mod 4$, and the best when $d \equiv 0 \mod 4$, then needing $3d/4$ fewer random masks than [BBP+16, Algorithm 4]. The product complexity of both algorithms is equal and entirely due to the computation of the $\boldsymbol{a}_i \boldsymbol{b}_j$ terms, but our algorithm is better in terms of sum complexity. This latter is somewhat irrelevant for masking over large fields, since the cost is then dominated by the product and randomness complexity; however, for masking over $\mathbb{F}_2$, products and sums have a similar cost.

We also give explicit figures for $d \leq 11$ in Appendix A, Table 6.

5.2 Strong non-interfering schemes

We do not propose a generic family of new SNI schemes, but use a heuristic that successfully converts (most of) our new NI multiplications into SNI ones. The idea is to notice that from the point of view of an SNI attacker, an external probe (*i.e.* a probe on an output share) is especially powerful as it is not counted in the number of shares available to the simulator. We thus simply try to reduce the impact of such probes by systematically masking the output shares with fresh randomness.

Applying this heuristic results in *ad hoc* SNI multiplication gadgets experimentally verified to be secure up to order 9. However, this same heuristic seems to fail starting from $d = 10$, and our verification software finds attacks within seconds against all the candidate SNI gadgets that we tried for that order and beyond.

[9] Recall however that they also provide better algorithms at order 2, 3 and 4.

Algorithm 1: A d-NI $(d+1, d+1)$-gadget for multiplication over fields of characteristic two

Input : $\mathcal{S} = \{\hat{\boldsymbol{\alpha}}_{i,j}, 0 \leq i \leq j \leq d\}$
Input : $\mathcal{R} = \{\boldsymbol{r}_i\}$, $i \in \mathbb{N}$
Output: $(\boldsymbol{c}_i)_{0 \leq i \leq d}$, such that $\sum_{i=0}^{d} \boldsymbol{c}_i = \sum_{i=0}^{d} \boldsymbol{a}_i \sum_{i=0}^{d} \boldsymbol{b}_i$

for $i \leftarrow\!\!\shortmid 0$ **to** d **do**
  $\boldsymbol{c}_i \leftarrow\!\!\shortmid \hat{\boldsymbol{\alpha}}_{i,i}$
  $\mathcal{S} \leftarrow\!\!\shortmid \mathcal{S} \setminus \{\hat{\boldsymbol{\alpha}}_{i,i}\}$
end
$\mathcal{R}' \leftarrow\!\!\shortmid \{\}$
$j \leftarrow\!\!\shortmid 1$
while $\mathcal{S} \neq \emptyset$ **do**
  for $i \leftarrow\!\!\shortmid 0$ **to** d **do**
    if $j \equiv 1 \mod 2$ **then**
      $\boldsymbol{c}_i \leftarrow\!\!\shortmid \boldsymbol{c}_i + \boldsymbol{r}_{\frac{(j-1)}{2}.(d+1)+i}$
      $\mathcal{R}' \leftarrow\!\!\shortmid \mathcal{R}' \cup \left\{\boldsymbol{r}_{\frac{(j-1)}{2}.(d+1)+i}\right\}$
    else
      $\boldsymbol{c}_i \leftarrow\!\!\shortmid \boldsymbol{c}_i + \boldsymbol{r}_{\frac{(j-2)}{2}.(d+1)+(i+1 \mod (d+1))}$
      $\mathcal{R}' \leftarrow\!\!\shortmid \mathcal{R}' \setminus \left\{\boldsymbol{r}_{\frac{(j-2)}{2}.(d+1)+(i+1 \mod (d+1))}\right\}$
    end
    if $\mathcal{S} \neq \emptyset$ **then**
      $\boldsymbol{c}_i \leftarrow\!\!\shortmid \boldsymbol{c}_i + \hat{\boldsymbol{\alpha}}_{i,((i+j) \mod (d+1))}$
      $\mathcal{S} \leftarrow\!\!\shortmid \mathcal{S} \setminus \{\hat{\boldsymbol{\alpha}}_{i,((i+j) \mod (d+1))}\}$
    else
      break
    end
  end
  $j \leftarrow\!\!\shortmid j + 1$
end
$k \leftarrow\!\!\shortmid \#\mathcal{R}'$
for $i \leftarrow\!\!\shortmid 0$ **to** d **do**
  $\boldsymbol{c}_i \leftarrow\!\!\shortmid \boldsymbol{c}_i + \boldsymbol{r}_{\frac{(j-1)}{2}(d+1)+(i+1 \mod k)}$
end

Table 1: Complexity comparison for d-NI multiplication

Metric	[BBP+16, Algorithm 4]	Algorithm 1
Sum	$\frac{d(7d+10)}{4}$ (d even) $\frac{(7d+1)(d+1)}{4}$ (d odd)	$(d+1)(d+2\lceil\frac{d}{4}\rceil)$ ($d \not\equiv 1 \mod 4$) $(d+1)(2\lceil\frac{d}{4}\rceil + d - 1) + 2$ ($d \equiv 1 \mod 4$)
Product	$(d+1)^2$	$(d+1)^2$
Randomness	$(\frac{d^2}{4}) + d$ (d even) $\frac{(d^2-1)}{4} + d$ (d odd)	$(d+1)\lceil\frac{d}{4}\rceil$ ($d \not\equiv 1 \mod 4$) $(d+1)\lceil\frac{d}{4}\rceil - \frac{d+1}{2} + 1$ ($d \equiv 1 \mod 4$)

We give the full description of our *ad-hoc* SNI gadgets at https://github.com/NicsTr/binary_masking , and provide explicit cost figures in Table 6, Appendix A. It is important to note that these gadgets do not always improve on ISW multiplication [ISW03]: our 1-SNI multiplication is more expensive, and the cost is identical at order 2,3,4 and 6. Yet they give notable improvements for the remaining orders 5,7,8 and 9; for instance, our 7-SNI multiplication uses only 20 fresh random masks compared to 28 for ISW —a close to 30% improvement.

5.3 Examples

We present two d-NI gadgets and the corresponding d-SNI variants in Figure 2, using the concise notation of the tool from EUROCRYPT 2016 recalled in Section 4.3.

The description of our other gadgets can be found at https://github.com/NicsTr/binary_masking.

```
s00 r00 s01 s10 r01 s02 s20 r07 s03 s30 r08
s11 r01 s12 s21 r02 s13 s31 r08 s14 s41 r09
s22 r02 s23 s32 r03 s24 s42 r09 s25 s52 r10
s33 r03 s34 s43 r04 s35 s53 r10 s36 s63 r11
s44 r04 s45 s54 r05 s46 s64 r11 s40 s04 r12
s55 r05 s56 s65 r06 s50 s05 r12 s51 s15 r13
s66 r06 s60 s06 r00 s61 s16 r13 s62 s26 r07
```

(a) 6-NI multiplication, 14 random masks

```
s00 r00 s01 s10 r01 s02 s20 r07 s03 s30 r08 r14 r20
s11 r01 s12 s21 r02 s13 s31 r08 s14 s41 r09 r15 r14
s22 r02 s23 s32 r03 s24 s42 r09 s25 s52 r10 r16 r15
s33 r03 s34 s43 r04 s35 s53 r10 s36 s63 r11 r17 r16
s44 r04 s45 s54 r05 s46 s64 r11 s40 s04 r12 r18 r17
s55 r05 s56 s65 r06 s50 s05 r12 s51 s15 r13 r19 r18
s66 r06 s60 s06 r00 s61 s16 r13 s62 s26 r07 r20 r19
```

(b) 6-SNI multiplication, 21 random masks

```
s00 r00 s01 s10 r01 s02 s20 r06 s03 s30 r07
s11 r01 s12 s21 r02 s13 s31 r07 s14 s41 r08
s22 r02 s23 s32 r03 s24 s42 r08 s25 s52 r09
s33 r03 s34 s43 r04 s35 s53 r09 r06
s44 r04 s45 s54 r05 s40 s04
s55 r05 s50 s05 r00 s51 s15
```

(c) 5-NI multiplication, 10 random masks

```
s00 r00 s01 s10 r01 s02 s20 r06 s03 s30 r07 r10
s11 r01 s12 s21 r02 s13 s31 r07 s14 s41 r08 r11
s22 r02 s23 s32 r03 s24 s42 r08 s25 s52 r09
s33 r03 s34 s43 r04 s35 s53 r09 r06
s44 r04 s45 s54 r05 s40 s04 r10
s55 r05 s50 s05 r00 s51 s15 r11
```

(d) 5-SNI multiplication, 12 random masks

Fig. 2: 5-and-6 NI and SNI gadgets for multiplication.

5.4 Applications

Example of the AES field inversion. To show the usefulness of their new NI multiplication, Belaïd *et al.* improve in [BBP+16, Proposition 7.7] the SNI field inversion circuit of [BBD+15] (which is the costliest part of computing a masked AES S-box). This improvement consists in replacing two of the four SNI multiplications by NI multiplications, and since they did not propose any new SNI schemes, they use ISW multiplication for the other two.

We use the same example to demonstrate in Table 2 the impact of our new schemes at order 8 and 9. In the former case, this reduces the need for fresh randomness by 25% and the number of sums by more than 15%; in the latter, there is again a reduction of fresh randomness by nearly 25% and of sums by about 12%. It is interesting to notice that the randomness improvements due to the NI and SNI gadgets are markedly different in these two cases: at order 8 both of our gadgets reduce the cost in equal proportion, but at order 9 most of the gain is due to the SNI one.

Finally, note that the algorithms of [BBP+17] could in principle give even better results, but there is no known instantiation of them over $\mathbb{F}_{2^8}$ at order more than 4 [KR18].

Bitsliced high-order masking. A recent trend in practical implementations of high-order masking schemes is to use a parallel or bitsliced approach, see *e.g.* [GR17,BDF+17,JS17,GJRS18,GPSS18]. In that respect, since our schemes are naturally defined over $\mathbb{F}_2$, they seem to be good candidates to replace the algorithms of [BBP+16] and [ISW03] whenever these are used. This would for instance not easily be the case for the algorithms of [BBP+17] even if high-order instantiations could be

Table 2: Cost of the d-SNI AES field inversion at order $d = 8$ and $d = 9$ using the circuit of [BBP+16, Section 7.2]. All quantities are over $\mathbb{F}_{2^8}$.

	$d = 8$		$d = 9$	
	[BBP+16]	§5	[BBP+16]	§5
Sums	$2 \times 132 + 2 \times 144 = 552$	$2 \times 108 + 2 \times 126 = 468$	$2 \times 160 + 2 \times 180 = 680$	$2 \times 142 + 2 \times 150 = 600$
Products	$4 \times 81 = 324$	$4 \times 81 = 324$	$4 \times 100 = 400$	$4 \times 100 = 400$
Random masks	$2 \times 24 + 2 \times 36 = 120$	$2 \times 18 + 2 \times 27 = 90$	$2 \times 29 + 2 \times 45 = 148$	$2 \times 26 + 2 \times 30 = 112$

found as they require multiplications by elements of extension fields, which is incompatible with a bitsliced approach.[10] Yet, as is shown by Goudarzi *et al.* [GJRS18], different tradeoffs may be available for optimised implementations of different schemes. It would thus be an interesting future work to precisely assess the cost of vectorised implementations of our gadgets.

5.5 Verification performance

We conclude this section with an evaluation of the performance of our verification software from Sections 3 and 4, that we use to prove the security of our schemes.

Probes filtering. Following the results of Section 3.2, we use a filtering process to reduce the initial set of probes that one has to enumerate to prove security into a smaller equipotent subset. For our schemes, this means removing probes of the form: $\hat{\boldsymbol{\alpha}}_{*,*} + \sum(\boldsymbol{r}_* + \hat{\boldsymbol{\alpha}}_{*,*}) + \boldsymbol{r}_* + \boldsymbol{a}_*\boldsymbol{b}_*$,[11] and the equipotency of the filtered set and the original one is verified by an exhaustive check on the subsets corresponding to every output share. Intuitively, the idea is that one can always replace in an attack a probe of the above form with one that includes one extra $\boldsymbol{a}_j\boldsymbol{b}_i$ term, *i.e.* one of the form $\hat{\boldsymbol{\alpha}}_{*,*} + \sum(\boldsymbol{r}_* + \hat{\boldsymbol{\alpha}}_{*,*}) + \boldsymbol{r}_* + \hat{\boldsymbol{\alpha}}_{*,*}$, since the latter only adds an additional functional dependence on the input shares "for free".

The concrete impact of filtering on the verification performance of our schemes can be seen in Table 3, where we give the size of the attack sets to enumerate before and after this filtering.

Performance. For order $d \leq 10$, we have run our software on a single core of the `retourdest` server, which features a single Intel Xeon Gold 6126 at 2.60 GHz. The corresponding timings are given in Table 3. At peak performance, we are able to enumerate $\approx 2^{27.5}$ candidate attack sets per second for NI verification, while SNI performance is slightly worse.

Using filtered set significantly improves verification time, especially at high order. For instance, the running times of 2 and 6 hours for NI and SNI multiplication at order 9 are an order of magnitude faster than the 3 and 6 days initially spent before we implemented filtering. This optimisation was also essential in allowing to check the security of our 10-NI gadget in less than one calendar day (using parallelisation); it would otherwise have taken a rather costly 1 core-year.

We also tested a multi-threaded implementation of our software on schemes at order $8 \sim 10$, using all 12 physical cores of the same Xeon Gold 6126; the results are shown in Table 4. While we do not have many data points, the speed-up offered by the parallelisation seems to be close to linear, albeit slightly less for NI verification: the 9-SNI multi-threaded wall time is ≈ 11.7 times less than the single-threaded one, and multi-threading for 9- and 10-NI saves a factor ≈ 9.7.

The largest instance that we verified is the NI multiplication at order $d = 11$. We relied heavily on parallelisation to enumerate the $\approx 2^{52.72}$ possible attack sets,[12] using up to 16 nodes of the *Dahu* cluster.[13] Each node has two 16-core Intel Xeon Gold 6130 at 2.10 GHz, and when using

[10] On high-end architectures, one might still be able to pack several multiplications using an instruction such as `pclmulqdq`, but that would arguably not be "pure" bitslicing.

[11] This corresponds exactly to the probes made of an even number of $\boldsymbol{a}_*\boldsymbol{b}_*$ terms.

[12] This is after filtering of the initial $\approx 2^{59}$ sets.

[13] https://ciment.univ-grenoble-alpes.fr/wiki-pub/index.php/Hardware:Dahu

hyperthreading allows to enumerate $\approx 2^{31.38}$ sets per second. This overall lead to a wall-time of only three days.

Comparison with maskVerif. We have also verified the security of our gadgets at order 6 to 8 with the maskVerif tool from Barthe *et al.* [BBC+19].[14] Due to system constraints, we could not run the verification on `retourdest`, and instead defaulted to the older `hpac`, which features an Intel Xeon E5-4620 at 2.20 GHz. Yet since the maskVerif implementation is parallel and may use up to four threads, we believe that comparison with Table 3 is nonetheless meaningful.

The running times are summarised in Table 5. Comparing with Table 3, it is notable that our own software is faster by three orders of magnitude, for instance taking one minute to check 8-NI multiplication *versus* two days for maskVerif. However, it must be noted that maskVerif is much more generic than our software, and is in particular able to analyse the security of hardware countermeasures in presence of glitches.

Table 3: Running time of our verification software (sequential)

Order d		$\log_2$(number of sets) Before/After filtering	Wall time Best (after filtering)
1	NI	2.6/2.6	$<$ 0.01 sec.
	SNI	2.6/2.6	$<$ 0.01 sec.
2	NI	6.3/5.5	$<$ 0.01 sec.
	SNI	6.3/5.5	$<$ 0.01 sec.
3	NI	10.4/8.9	$<$ 0.01 sec.
	SNI	11.2/9.96	$<$ 0.01 sec.
4	NI	15.0/12.6	$<$ 0.01 sec.
	SNI	16.6/15.0	$<$ 0.01 sec.
5	NI	21.2/18.6	$<$ 0.01 sec.
	SNI	21.7/19.3	$<$ 0.01 sec.
6	NI	27.1/23.9	0.09 sec.
	SNI	28.7/26.2	0.59 sec.
7	NI	32.7/28.7	2.43 sec.
	SNI	33.6/30.0	7.52 sec.
8	NI	38.5/33.7	1 min. 17 sec.
	SNI	40.3/36.3	9 min. 28 sec.
9	NI	45.6/40.5	2 h. 18 min.
	SNI	46.3/41.6	6 h. 30 min.
10	NI	52.6/47.10	9 days 3h.

Table 4: Running time of our verification software (parallel, 12 threads)

Order d		Wall time
8	NI	7.43 sec.
	SNI	47.0 sec.
9	NI	14 min. 20 sec.
	SNI	33 min. 20 sec.
10	NI	22 h. 30 min.

[14] Available at https://gitlab.com/benjgregoire/maskverif.

Table 5: Running time of maskVerif (parallel, up to 4 threads) [BBC+19]

Order d		Wall time
6	NI	2 min. 44 sec.
	SNI	9 min. 0 sec.
7	NI	1 h. 39 min.
	SNI	5 h. 54 min.
8	NI	2 days 10h.
	SNI	13 days 6h.

Acknowledgments

We thank Clément Pernet for his part in the proof of Lemma 14, Yann Rotella for an early discussion on the possibility of further filtering, and the authors of [BBC+19] for providing us access to an up-to-date version of maskVerif.

This work is partially supported by the French National Research Agency in the framework of the *Investissements d'avenir* programme (ANR-15-IDEX-02).

Some of the computations presented in this paper were performed using the GRICAD infrastructure (https://gricad.univ-grenoble-alpes.fr), which is partially supported by the Equip@Meso project (ANR-10-EQPX-29-01) of the *Investissements d'Avenir* programme.

References

BBC+19. Gilles Barthe, Sonia Belaïd, Gaëtan Cassiers, Pierre-Alain Fouque, Benjamin Grégoire, and François-Xavier Standaert. maskVerif: Automated Verification of Higher-Order Masking in Presence of Physical Defaults. In *ESORICS 2019 (to appear)*, 2019. Available at https://eprint.iacr.org/2018/562.

BBD+15. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. Compositional Verification of Higher-Order Masking: Application to a Verifying Masking Compiler. *IACR Cryptology ePrint Archive*, 2015:506, 2015.

BBD+16. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong Non-Interference and Type-Directed Higher-Order Masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 116–129. ACM, 2016.

BBP+16. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness Complexity of Private Circuits for Multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648. Springer, 2016.

BBP+17. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Private Multiplication over Finite Fields. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017*, volume 10403 of *Lecture Notes in Computer Science*, pages 397–426. Springer, 2017.

BDF+17. Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model. In Coron and Nielsen [CN17], pages 535–566.

CN17. Jean-Sébastien Coron and Jesper Buus Nielsen, editors. *Advances in Cryptology — EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, 2017.

dah. The Dahu cluster. https://ciment.univ-grenoble-alpes.fr/wiki-pub/index.php/Hardware:Dahu.

DFS15. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

FG18. Junfeng Fan and Benedikt Gierlichs, editors. *Constructive Side-Channel Analysis and Secure Design — COSADE 2018*, volume 10815 of *Lecture Notes in Computer Science*. Springer, 2018.

GJRS18. Dahmun Goudarzi, Anthony Journault, Matthieu Rivain, and François-Xavier Standaert. Secure Multiplication for Bitslice Higher-Order Masking: Optimisation and Comparison. In Fan and Gierlichs [FG18], pages 3–22.

GPSS18. Benjamin Grégoire, Kostas Papagiannopoulos, Peter Schwabe, and Ko Stoffelen. Vectorizing Higher-Order Masking. In Fan and Gierlichs [FG18], pages 23–43.

GR17. Dahmun Goudarzi and Matthieu Rivain. How Fast Can Higher-Order Masking Be in Software? In Coron and Nielsen [CN17], pages 567–597.

ISW03. Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

JS17. Anthony Journault and François-Xavier Standaert. Very High Order Masking: Efficient Implementation and Security Evaluation. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643. Springer, 2017.

KJJ99. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael J. Wiener, editor, *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

Knu11. Donald E. Knuth. *Combinatorial Algorithms, Part 1*, volume 4A of *The Art of Computer Programming*. Addison Wesley, 2011.

KR18. Pierre Karpman and Daniel S. Roche. New Instantiations of the CRYPTO 2017 Masking Schemes. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018*, volume 11273 of *Lecture Notes in Computer Science*, pages 285–314. Springer, 2018.

LT73. C. N. Liu and Donald T. Tang. Enumerating Combinations of m Out of n Objects [G6] (Algorithm 452). *Commun. ACM*, 16(8):485, 1973.

NW78. Albert Nijenhuis and Herbert S. Wilf. *Combinatorial algorithms for computers and calculators*. Academic Press, second edition, 1978.

Pra62. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Information Theory*, 8(5):5–9, 1962.

Sch80. Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980.

Wal. Timothy R. Walsh. A simple sequencing and ranking method that works on almost all gray codes. Unpublished research report. Available at: https://www.labunix.uqam.ca/~walsh_t/papers/sequencing_and_ranking.pdf.

A Explicit costs of multiplication gadgets

We only provide a comparison of our gadgets with [ISW03] and [BBP+16, Alg. 4]. In the latter case, better algorithms were already provided by the authors up to order 4. The main range of interest of our gadgets is thence at order 5 and beyond.

Table 6: Explicit costs of multiplication gadgets

Order d		[BBP+16, Alg. 4] (NI) [ISW03] (SNI)		§5.1 (NI) §5.2 (SNI)	
		Random masks	Sums	Random masks	Sums
1	NI	•	•	2	5
	SNI	1	4	2	5
2	NI	3	12	3	12
	SNI	3	12	3	12
3	NI	5	22	4	20
	SNI	6	24	6	24
4	NI	8	38	5	30
	SNI	10	40	10	40
5	NI	11	54	10	50
	SNI	15	60	12	54
6	NI	15	78	14	70
	SNI	21	84	21	84
7	NI	19	100	16	88
	SNI	28	112	20	96
8	NI	24	132	18	108
	SNI	36	144	27	126
9	NI	29	160	26	142
	SNI	45	180	30	150
10	NI	35	200	33	176
11	NI	41	234	36	204