Provably Secure Hybrid Inner Product and Boolean Masking via Composable Conversion

Jaeseung Han and Dong-Guk Han

Department of Financial Information Security, Kookmin University, Seoul, Republic of Korea,
{jae1115,christa}@kookmin.ac.kr

Abstract. Masking is a representative side-channel countermeasure that provides provable security. Among masking schemes, Boolean masking (BM) is widely adopted due to its simple sharing structure, while inner product masking (IPM) and code-based masking (CM) have been studied as alternatives that achieve a higher security order with the same number of shares—a property known as security order amplification in the bit-probing model. Recent work by Gaspoz and Dhooghe (TCHES 2025) proposed an IPM multiplication gadget and CM gadgets with provable bit-level security; however, the overhead of CM gadgets for linear operations, the overhead of IPM multiplication, and the lack of a complete provably secure implementation exploiting IPM security order amplification remain open challenges.

In this paper, we address all three challenges. First, we propose BM-to-IPM and IPM-to-BM conversion gadgets satisfying bit t-MIMO-SNI in the bit-probing model, enabling composable and provably secure interoperation between the two masking domains. Second, we optimize the TCHES 2025 IPM multiplication gadget via Row Packing and Reduction in Rows, reducing the fresh random bit requirement from $\frac{1}{2}t(n^2-1)k^2(k+1)$ to $tk(n-1)(kn+W)$ bits with a proportional reduction in XOR gates, while maintaining bit t-SNI security. Third, we present a hybrid IPM-BM framework in which multiplications are performed in IPM with fewer shares and all Boolean linear operations are handled share-wise in BM at no additional randomness cost, and show that this hybrid approach requires significantly fewer gates and random bits than a pure CM approach. As a concrete instantiation, we implement a second-order masked AES-128 with a 2-share IPM / 3-share BM hybrid architecture, prove that the implementation satisfies bit 2-PINI, and evaluate its practical side-channel security via first- and second-order TVLA on an ARM Cortex-M4 with up to one million traces. To the best of our knowledge, this is the first end-to-end cryptographic implementation that provably preserves IPM's security order amplification in the bit-probing model.

Keywords: Inner Product Masking · Boolean Masking · Side-channel Analysis · Probing Security · Non-interference · AES · Software

1 Introduction

Cryptographic implementations are vulnerable to side-channel attacks such as Differential Power Analysis (DPA) [KJJ99] and Correlation Power Analysis (CPA) [BCO04], which exploit physical leakage from a device to recover secret keys independently of the underlying algorithm's mathematical strength. Masking is one of the most widely studied and deployed countermeasures against such attacks: it randomizes the computation by splitting each sensitive variable into n shares. If any subset of at most t shares is statistically independent of the secret, then the masking achieves t-th order security.

Among masking schemes, Boolean masking (BM) [CJRR99, GP99] is the predominant choice in practice due to its simple structure and seamless applicability to binary-field oper-

ations. In addition to BM, there are various masking methods depending on purpose such as multiplicative masking [GT03], affine masking [FMPR11], polynomial masking [PR11, GM11], inner product masking (IPM) [BFG15, WSY+16, BFG+17], direct sum masking (DSM) [BCC+14, PGS+17], and code-based masking (CM) [WYS19, WMCS20, WCG+22]. BM achieves only a probing security order of $t = n - 1$, meaning that an adversary placing $t + 1$ probes suffices to recover the secret. To achieve higher security at a fixed share count, researchers have investigated DSM, CM and, in particular, IPM. In an $\bar{L}$-IPM encoding, a secret x is represented as shares $\bar{x} = (x_0, \ldots, x_{n-1})$ satisfying $x = \sum_{i=0}^{n-1} L_i x_i$ for a public vector $\bar{L} = (L_0, L_1, \ldots, L_{n-1})$. Poussier et al. [PGS+17] formalized the key advantage of IPM by introducing the *bit-probing model* and showing that, under linear leakage, IPM achieves a strictly higher security order than BM with the same number of shares—a phenomenon called *security order amplification*. For example, a 2-share IPM over $\mathbb{F}_{2^4}$ with an optimally chosen $\bar{L}$ attains second-order bit-probing security, matching the protection level of a 3-share BM implementation. This property is particularly attractive in scenarios where the number of shares is constrained by hardware resources (e.g., register pressure in embedded processors), as IPM can provide a higher security order without increasing the share count.

Prior work and remaining challenges. Wang et al. [WSY+16] proposed Boolean matrix product masking, a hybrid encoding in which the first share is multiplied by a public non-singular Boolean matrix while the remaining shares follow standard BM. They proved probing security in the word-level model and observed security order amplification for linear leakage functions. However, the matrix structure is embedded in the encoding itself, so all gadgets—including linear ones—must account for the matrix product. Afterwards, Wang et al. [WMCS20] proposed gadgets for both linear and nonlinear operations under CM; however, their constructions were designed and proven secure in the word-level probing model and do not preserve security order amplification at the bit level. This was confirmed experimentally by Wu et al. [WCG+22], who showed that higher-order IPM gadgets built on Wang et al.'s method suffer a measurable reduction in practical security order. Gaspoz and Dhooghe [GD25a] addressed this by proposing the first IPM multiplication gadget proven bit t-SNI secure, empirically confirming that security order amplification is maintained during multiplication. In a companion work, Gaspoz and Dhooghe [GD25b] further proposed basic CM gadgets proven bit t-PINI, including Toffoli, bit-swap, and bit-XOR operations, enabling bit t-PINI secure CM implementations of arbitrary circuits. Nevertheless, three challenges remain open after [GD25a, GD25b]:

1. **Overhead of linear operations in IPM.** In BM, Boolean linear operations such as bit-swap or bit-XOR can be computed share-wise without any fresh randomness, trivially satisfying probing security. By contrast, the CM gadgets of [GD25b] require fresh randomness for such basic operations, incurring significant overhead that exceeds a simple share-wise repetition.

2. **Overhead of IPM multiplication.** The secure IPM multiplication of [GD25a] including preprocessing gadget requires fresh random bits and XOR gates substantially more than the BM baseline.

3. **Provably secure block cipher using IPM security.** No complete, provably secure cryptographic implementation exploiting IPM's security order amplification end-to-end has been demonstrated.

Our contributions. We resolve all three challenges and present a provably and practically secure hybrid IPM-BM masking.
1. Composable IPM-BM conversion (Section 3). We propose BM-TO-IPM and IPM-TO-BM conversion gadgets that satisfy bit t-MIMO-SNI in the bit-probing model.

MIMO-SNI is the natural compositional notion for gadgets with asymmetric input/output share counts and multiple inputs or outputs, and it implies bit t-PINI, enabling trivial composition with PINI linear gadgets. Our conversion gadgets are the first to provably preserve security order amplification across the IPM-BM boundary, and they enable an efficient hybrid design in which Boolean linear operations are handled in BM while multiplications are performed in IPM.

2. Optimized IPM multiplication (Section 4). Building on [GD25a], we reduce the fresh randomness and XOR gates of the IPM multiplication gadget via two optimizations *Row Packing* and *Reduction in Rows* cutting the random bit requirement from $\frac{1}{2}t(n^2-1)k^2(k+1)$ to $tk(n-1)(kn+W)$ bits while maintaining bit t-SNI security. The XOR gate count of the preprocessing gadget (Gadget-1) is likewise reduced proportionally.

3. Hybrid IPM-BM construction (Section 5). Using the proposed IPM-BM conversion gadgets, we present the hybrid IPM-BM framework. Unlike the hybrid encoding of [WSY+16], which embeds the matrix structure into the sharing itself and requires all gadgets to operate under the modified encoding, our approach separates the IPM and BM domains entirely through composable conversion gadgets proven secure in the bit-probing model. This domain separation allows each masking scheme to be used where it is most efficient, without altering the encoding structure of either scheme. As a concrete instantiation, we design a second-order bit-probing secure masked AES-128 with a 2-share IPM / 3-share BM hybrid architecture: multiplications, whose cost scales as $O(n^2)$ in the share count, are performed in the 2-share IPM, while all Boolean linear operations (AddRoundKey, ShiftRows, MixColumns, and the field isomorphism in SubBytes) are handled share-wise in the 3-share BM. We prove that the complete S-box satisfies bit t-PINI. The implementation is fully bitsliced, processing all 16 state bytes in parallel. We further show that our hybrid approach requires significantly fewer gates and random bits than a pure code-based masking approach using the CM gadgets of [GD25b], quantitatively justifying the hybrid design.

4. Practical leakage evaluation (Section 6). We evaluate the second-order hybrid masked AES on an ARM Cortex-M4 using non-specific fixed-vs-random TVLA [GJJR11] at both first and second order. No statistically significant leakage is detected at either order with up to 1 million traces, while the unmasked (RNG-off) baseline leaks immediately, confirming both the soundness of the measurement setup and the practical effectiveness of the proposed scheme. In addition, we report the cost and breakdown of the second-order hybrid IPM-BM AES.

Paper organization. Section 2 recalls the necessary background on masking and security definitions. Section 3 presents the IPM-BM conversion gadgets and their security proofs. Section 4 describes the optimized IPM multiplication. Section 5 applies the combined gadgets to AES. Section 6 reports the practical side-channel evaluation and the cost of our implementation. Section 7 concludes.

2 Preliminaries

2.1 Notations

We denote the finite field of order 2^k as $\mathbb{F}_{2^k}$ and its k-dimensional vector space over $\mathbb{F}_2$ as $\mathbb{F}_2^k$. Elements of $\mathbb{F}_{2^k}$ and $\mathbb{F}_2^k$ are represented by lower-case letters, while matrices are denoted by bold upper-case letters. We write $\oplus$ for field addition in $\mathbb{F}_{2^k}$ (or XOR over $\mathbb{F}_2^k$) and $\cdot$ for field multiplication in $\mathbb{F}_{2^k}$.

For a positive integer n, we denote by $[n]$ the set $\{0,1,\ldots,n-1\}$. An n-share masking of a secret $x \in \mathbb{F}_{2^k}$ is denoted $\bar{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_{2^k}^n$. We use $x[i]$ to indicate the

i-th bit of x when x is viewed as a vector in $\mathbb{F}_2^k$ and $x[0]$ is the least significant bit of x. Individual shares are labeled $x_i \in \mathbb{F}_{2^k}$ for $i \in [n]$.

For a matrix $\mathbf{A}$, we write $\mathbf{A}[i,*]$ (resp. $\mathbf{A}[*,j]$) to denote the i-th row (resp. j-th column) of $\mathbf{A}$ and $\mathbf{A}[i,j]$ to denote the element of i-th row and j-th column. Given a linear code $\mathcal{C}$, we denote its dual minimal distance by $d_{\mathcal{C}}^{\perp}$, i.e., the minimum Hamming weight of any non-zero codeword in its dual code $\mathcal{C}^{\perp}$.

2.2 Security Model for Masking

We recall the security models and notions used throughout this paper. We recall the probing model at both the word level and the bit level, and then introduce composable security notions that guarantee probing security under composition.

Probing model. The probing security model was introduced by Ishai, Sahai, and Wagner [ISW03].

Definition 1 (Probing model)**.** A t-th order probing adversary can observe the values of at most t wires (called *probes*) in a circuit. A circuit is *t-probing secure* if the joint distribution of any t probed wires is independent of the secret inputs.

Word-level and bit-level probes. We distinguish between two levels of granularity for probes.

Definition 2 (Word-level probe [BFG+17])**.** A word-level probe reveals an entire element of $\mathbb{F}_{2^k}$.

Definition 3 (Bit-level probe [PGS+17])**.** A bit-level probe (bit-probe) reveals a single bit regardless of the field in which shares are encoded.

The security order of a masking scheme may differ between the word-level and bit-level models. In the word-level model, an n-share BM or IPM achieves security order $t_w = n-1$. In the bit-level model, IPM can achieve a strictly higher security order $t_b > n-1$, as formalized in Subsection 2.4.

Composable security. Barthe et al. [BBD+16] introduced the NI, SNI notions to enable provably secure *composition* of gadgets.

Definition 4 (t-NI [BBD+16])**.** A gadget is t-NI if for any set of t_1 internal probes such that $t_1 \le t$, all probes can be perfectly simulated using at most t_1 shares of each input.

Definition 5 (t-SNI [BBD+16])**.** A gadget is t-SNI if for any set of t_1 internal probes and any set of t_2 output probes such that $t_1 + t_2 \le t$, all probes can be perfectly simulated using at most t_1 shares of each input.

As a result, SNI gadgets support secure composition; in particular, a composition of t-SNI gadgets where each sharing feeds into at most one gadget is again t-SNI [BBD+16]. However, an arbitrary composition of SNI gadgets is not guaranteed to be probing secure. The following security notion guarantees probing security for arbitrary compositions.

Cassiers and Standaert [CS20] introduced Probe Isolating Non-Interference (PINI), which permits *trivial composition*: any composition of PINI gadgets is again PINI, without requiring intermediate refresh gadgets even for linear operations. In the following, $x_{i,*}$ and $y_{i,*}$ denote all input bits and all output bits, respectively, belonging to share index i.

Definition 6 (t-PINI [CS20])**.** A gadget is t-PINI if for any set P of t_1 internal probes and any set A of t_2 share indices (for output probes) such that $t_1 + t_2 \le t$, there exists a set B of at most t_1 share indices such that $P \cup y_{A,*}$ can be simulated using $x_{A \cup B,*}$.

Proposition 1 (PINI composability [CS20])**.** *Any composite gadget made of t-PINI composing gadgets is t-PINI.*

The Multiple-Input Multiple-Output SNI (MIMO-SNI) notion [CS20] is a natural extension of SNI to gadgets with multiple outputs. Any t-MIMO-SNI gadget is also t-PINI, and thus benefits from trivial composition.

Definition 7 (t-MIMO-SNI [CS20])**.** A gadget with m inputs and n outputs is t-MIMO-SNI if for any set I of t_1 internal probes and any sets O_i of share indices for each output $i \in [n]$, such that there exists t_2 with $t_1 + t_2 \leq t$ and $|O_i| \leq t_2$ for all i, the probes $I \cup \bigcup_i y_{O_i,i}$ can be simulated using at most t_1 input shares.

Proposition 2 ([CS20])**.** *Any t-MIMO-SNI gadget is t-PINI.*

A composition of gadgets is MIMO-SNI if it satisfies several conditions. One key condition is defined below.

Definition 8 (Single-Path-NI-Built gadget (SP-NIB) [CS20])**.** A composite gadget G is SP-NIB if it is implemented with only NI gadgets and SNI refreshes, and if for any pair of vertices u, v in the corresponding simplified computation graph there exists at most one path from u to v.

If a composition of gadgets is SP-NIB and no pair of inputs or outputs is simultaneously connected to an internal node, then the composite gadget is MIMO-SNI. The following proposition formalizes this.

Proposition 3 ([CS20])**.** *A composite gadget G is t-MIMO-SNI if it satisfies the three following conditions. (i) G is SP-NIB. (ii) For any pair of output nodes u_1, u_2 there is no node v such that there is a path from v to u_1 and a path from v to u_2. (iii) For any pair of input nodes u_1, u_2 there is no node v such that there is a path from u_1 to v and a path from u_2 to v.*

The original definitions of composable security generally use word-level probes. Their bit-level counterparts are obtained by replacing each word-level unit—probes, inputs, and outputs—with individual bits. For example, in bit t-SNI, 't_1 internal probes' refers to t_1 individual bits of internal wires, and 'each input' refers to each input bit.

2.3 Boolean Masking

BM is a widely deployed countermeasure against side-channel attacks, introduced by Chari et al. [CJRR99] and Goubin–Patarin [GP99].

Definition 9 (Boolean Masking)**.** A t-th order Boolean masking of a secret $x \in \mathbb{F}_{2^k}$ is an $(n = t + 1)$-share encoding $\bar{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_{2^k}^n$ where $x_1, \ldots, x_{n-1}$ are drawn uniformly at random from $\mathbb{F}_{2^k}$, and $x_0 = x \oplus x_1 \oplus \cdots \oplus x_{n-1}$.

The key property of BM is that any subset of at most $t = n - 1$ shares is statistically independent of the secret x. Computations on Boolean-masked data are performed using *gadgets*, which are randomized circuits that take masked inputs and produce masked outputs while preserving the secret. The most fundamental gadget is the ISW multiplication gadget [ISW03], which securely computes $z = x \cdot y$ given $\bar{x}$ and $\bar{y}$, requiring $\binom{n}{2}$ fresh random field elements and n^2 field multiplications.

Linear operations (e.g., XOR, linear maps) can be implemented *trivially* on Boolean-masked data by applying the operation share-wise, requiring no additional randomness.

2.4 Inner Product Masking

IPM was introduced by Balasch et al. [BFG15, BFG+17] as an alternative to BM with enhanced security properties.

Definition 10 ($\bar{L}$-IPM [GD25a])**.** An $\bar{L}$-inner product masking ($\bar{L}$-IPM) of $x \in \mathbb{F}_{2^k}$ with a public parameter $\bar{L} = (1, L_1, \ldots, L_{n-1}) \in \mathbb{F}_{2^k}^n \setminus \{0\}$ is a vector $(x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_{2^k}^n$ such that

$$x = \sum_{i=0}^{n-1} L_i \cdot x_i = x_0 \oplus L_1 x_1 \oplus \cdots \oplus L_{n-1} x_{n-1}.$$

A *uniform* $\bar{L}$-IPM is one where $x_1, \ldots, x_{n-1}$ are drawn uniformly at random from $\mathbb{F}_{2^k}$, and x_0 is determined by the equation above.

In contrast to BM (where all $L_i = 1$), the public vector $\bar{L}$ introduces algebraic structure into the sharing via multiplication with field elements. This structure enables a higher security order in the *bit-probing model* at the cost of more complex gadgets, a phenomenon known as *security order amplification*.

Security order amplification. Poussier et al. [PGS+17] formalized the security order amplification of IPM by identifying it as a special instance of DSM. In DSM, a secret $[x]_2 \in \mathbb{F}_2^k$ is encoded as $z = [x]_2\mathbf{G} + y\mathbf{H} \in \mathbb{F}_2^{nk}$, where $\mathbf{G}$ and $\mathbf{H}$ are generator matrices of two binary codes $\mathcal{C}$ and $\mathcal{D}$. An $\bar{L}$-IPM with public vector $\bar{L} = (1, L_1, \ldots, L_{n-1})$ corresponds to such a DSM encoding over $\mathbb{F}_2^{nk}$ with

$$\mathbf{G} = \begin{pmatrix} \mathbf{I}_k & \mathbf{0}_k & \cdots & \mathbf{0}_k \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} \mathbf{M}_1 & \mathbf{I}_k & \mathbf{0}_k & \cdots & \mathbf{0}_k \\ \mathbf{M}_2 & \mathbf{0}_k & \mathbf{I}_k & \cdots & \mathbf{0}_k \\ \vdots & \vdots & & \ddots & \vdots \\ \mathbf{M}_{n-1} & \mathbf{0}_k & \cdots & \mathbf{0}_k & \mathbf{I}_k \end{pmatrix},$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{M}_i \in \mathbb{F}_2^{k \times k}$ is the binary matrix representation of multiplication by L_i in $\mathbb{F}_{2^k}$.

Proposition 4 ([PGS+17])**.** *The bit-probing security order of a DSM encoding defined by binary codes $\mathcal{C}$ and $\mathcal{D}$ (with generators $\mathbf{G}$ and $\mathbf{H}$) equals $d_{\mathcal{D}}^{\perp} - 1$, where $d_{\mathcal{D}}^{\perp}$ is the dual minimal distance of $\mathcal{D}$.*

Consequently, the bit-probing security order of an $\bar{L}$-IPM equals $d_{\bar{L}}^{\perp} - 1$, where $d_{\bar{L}}^{\perp}$ is the dual minimal distance of the binary code generated by $\mathbf{H}$ above. Under the assumption of *linear leakage functions* (i.e., leakage is linear in the bits of each share), this bit-probing security order coincides with the statistical security order in the bounded moment model [PGS+17]. This strictly exceeds the word-level security order $t_w = n - 1$ when $\bar{L}$ is chosen to maximize $d_{\bar{L}}^{\perp}$. Optimal parameters for $\bar{L}$ have been studied in [CGC+21]; representative examples are shown in Table 1.

Table 1: Optimal bit-probing security orders t_b for $\bar{L}$-IPM [CGC+21]. Here $t_w = n - 1$ denotes the word-level security order.

$\mathbb{F}_{2^k}^n$	$d_{\bar{L}}^{\perp}$	$t_b = d_{\bar{L}}^{\perp} - 1$	$t_w = n - 1$
$n = 2$, $k = 4$	3	2	1
$n = 2$, $k = 8$	4	3	1
$n = 3$, $k = 4$	6	5	2
$n = 3$, $k = 8$	8	7	2

3 Composable Conversion Between Inner Product and Boolean Masking

This section presents our construction of bit t-MIMO-SNI conversion gadgets between IPM and BM. We first explain the design rationale and the structural differences that the conversion gadget must accommodate. We then describe the detailed algorithms for each direction, establish their correctness, and prove bit t-MIMO-SNI security.

3.1 Design Overview

There have already been conversion algorithms between different types of masking. Notably, Goubin proposed a conversion between BM and arithmetic masking [Gou01], and Coron et al. extended this conversion to an arbitrary higher security order [CGV14]. A conversion between CM and BM was also indirectly addressed in [WMCS20] through an algorithm called CodeL. CodeL is a generalized gadget for evaluating Boolean-linear functions under CM, and is structured in three parts: Part A decodes the CM into an additive sharing (BM), Part B applies the Boolean-linear function on the BM shares, and Part C re-encodes the result back into a CM codeword. However, since this construction was designed and proven secure in the word-level probing model, it does not preserve the *security order amplification* of IPM as characterized by the bit-probing model—and therefore provides no additional bit-level security (e.g., third-order bit security with two shares). This limitation is noted in Section 4.1 of [WMCS20], and was confirmed in practice in [WCG+22], where applying the CodeL-based approach to IPM was shown to break the order amplification. In this work, we design a conversion gadget that is secure in the bit-probing model and thereby preserves the security order amplification of IPM throughout computation.

Before describing the construction, we highlight two structural differences between IPM and BM gadgets in the bit-probing model that the conversion must account for: the number of shares and the number of gadget inputs and outputs.

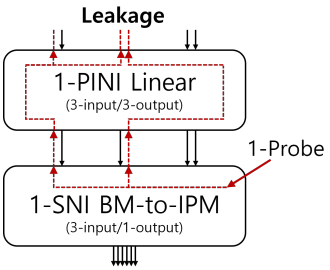
Number of shares. Compared to BM, IPM achieves a higher security order for the same number of shares. Consequently, for a fixed target security order, the BM and IPM require different numbers of shares. To ensure efficiency of both the conversion gadget and the overall algorithm, we design the conversion so that the security order is identical on both sides, while the number of shares differs between IPM and BM accordingly.

Number of gadget inputs and outputs. In the bit-probing model, an element of $\mathbb{F}_{2^k}$ is represented as k independent bits. This independence carries over to BM: since each bit of the secret is shared independently, a BM sharing of an element of $\mathbb{F}_{2^k}$ decomposes into k independent groups of bit shares, which can be treated as k independent inputs or outputs in a masking gadget. In contrast, for a non-trivial IPM (i.e., one that is not equivalent to BM), the shares of different bits are not independent. As a concrete example, consider $\mathbb{F}_{2^2} = \mathbb{F}_2[X]/(X^2 + X + 1)$ with $L_0 = 1$, $L_1 = X$, and a 2-share encoding of $x \in \mathbb{F}_{2^2}$:

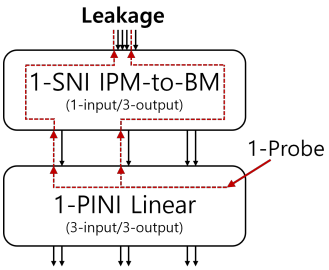
2-share BM: $z_0[0] = x[0] \oplus r[0],\ z_0[1] = x[1] \oplus r[1],\ z_1[0] = r[0],\ z_1[1] = r[1].$
2-share IPM: $z_0[0] = x[0] \oplus r[1],\ z_0[1] = x[1] \oplus r[0] \oplus r[1],\ z_1[0] = r[0],\ z_1[1] = r[1].$

In BM, the pair $(x_0[0], x_1[0])$ depends only on $r[0]$, and $(x_0[1], x_1[1])$ only on $r[1]$, so the two bits' shares are independent and can be treated as separate gadget inputs or outputs. In IPM, however, $z_0[1]$ depends on both $r[0]$ and $r[1]$, coupling it to $z_0[0]$, $z_1[0]$, and $z_1[1]$. Hence the bit shares of an IPM encoding cannot be treated as independent inputs or outputs.

Based on this observation, we define the conversion gadget interface as follows: an IPM sharing counts as one input or output unit, while a BM sharing counts as k independent input or output units.



(a) Case of 1-SNI BM-to-IPM conversion



(b) Case of 1-SNI IPM-to-BM conversion

Figure 1: Example of leakage by composition of a 1-SNI conversion and 1-PINI gadget (red arrow is 1-bit internal probe, red dashed lines are the propagated probes and the clustered lines represent shares of single input or output)

Choice of security model. Due to the asymmetry in the number of shares and inputs/outputs, the resulting BM-to-IPM and IPM-to-BM gadgets take a non-standard form with differing share counts on the two sides. We adopt MIMO-SNI as the security notion for the conversion gadgets. Alternative notions—PINI and SNI—each present limitations in this setting. Since the input and output share counts differ, PINI, which is defined in terms of share indices, cannot be applied directly. And since the gadgets have multiple inputs or multiple outputs, adopting plain SNI may break composability with the PINI gadgets (typically trivial BM linear gadgets) that precede or follow the conversion. We demonstrate in Figure 1 that an SNI conversion can cause leakage when composed with PINI gadgets. Figure 1a shows that a 1-bit internal probe of a 1-SNI BM-to-IPM conversion propagates to shares of two inputs. If the indices of the propagated shares are distinct, then 1-probing security is broken in the subsequent 1-PINI gadget. On the other hand, Figure 1b shows that a 1-bit internal probe of a 1-PINI gadget propagates to shares of two inputs. Since 1-SNI does not guarantee 1-probing security under two output probes, 1-probing security is broken in the 1-SNI IPM-to-BM conversion. These problems are solved by using an MIMO-SNI conversion gadget. Unlike in Figure 1, an MIMO-SNI conversion does not propagate a single probe into more than one input and guarantees probing security against multiple output probes.

3.2 Construction and Correctness

We assume throughout that BM sharing uses $t+1$ shares satisfying t-th order security, IPM sharing uses n shares satisfying the same t-th order security, and $L_0 = 1$.

3.2.1 BM-to-IPM Conversion

BM-TO-IPM (Algorithm 1) proceeds in three steps: fresh randomness generation, computation of the zeroth IPM share z_0, and computation of the remaining shares $z_1, \ldots, z_{n-1}$.

Step 1: Randomness generation. To cover all $t+1$ input BM shares, we sample $t(n-1)$ uniform random elements from $\mathbb{F}_{2^k}$, indexed as $r_{i,j}$ for $i \in [t+1]$ and $j \in [n]$. For each i, we set $r_{i,0} = \sum_{j=1}^{n-1} L_j r_{i,j}$, so that $(r_{i,0}, r_{i,1}, \ldots, r_{i,n-1})$ forms an IPM sharing of zero.

Step 2: Computing z_0. Each BM share x_i is first refreshed with $r_{i,0}$ to obtain $y_i = x_i + r_{i,0}$. The zeroth IPM share is then the cumulative sum $z_0 = \sum_{i=0}^{t} y_i$.

Step 3: Computing $z_1, \ldots, z_{n-1}$. The remaining $n-1$ IPM shares are set to $z_j = \sum_{i=0}^{t} r_{i,j}$ for $j \in \{1, \ldots, n-1\}$.

The proof of correctness is as follows:

$$z_0 = \sum_{i=0}^{t}(x_i + r_{i,0}) = \sum_{i=0}^{t}\left(x_i + \sum_{j=1}^{n-1} L_j r_{i,j}\right), \qquad z_j = \sum_{i=0}^{t} r_{i,j} \quad (j \geq 1).$$

Then,

$$\sum_{j=0}^{n-1} L_j z_j = z_0 + \sum_{j=1}^{n-1} L_j z_j = \sum_{i=0}^{t}\left(x_i + \sum_{j=1}^{n-1} L_j r_{i,j}\right) + \sum_{j=1}^{n-1} L_j \sum_{i=0}^{t} r_{i,j} = \sum_{i=0}^{t} x_i = x.$$

3.2.2 IPM-to-BM Conversion

IPM-TO-BM (Algorithm 2) proceeds in three steps: bit assignment, SNI refresh, and masking conversion.

Step 1: Bit assignment. We first collect the IPM shares by bit index, grouping them into BM-like bit shares before refreshing. Specifically, for each output bit index b, we assign the relevant bits from the IPM shares to an intermediate array $y_i[b]$ such that $x[b] = \sum_{i=0}^{t_b} y_i[b]$. Here t_b denotes the number of intermediate bits for index b, which is determined by the IPM parameter $\bar{L}$ and satisfies $t_b \geq t$ since the IPM provides t-th order security. This step involves only re-indexing and introduces no computational overhead; in a concrete implementation with a fixed $\bar{L}$, both t_b and the assignment indices are compile-time constants.

Step 2: SNI refresh. For each bit index b, a t_b-SNI refresh is applied to the $(t_b + 1)$-bit array $(y_0[b], \ldots, y_{t_b}[b])$. Since t_b may vary across bit indices (depending on $\bar{L}$), the refresh cost per bit also varies. We use the SNI refresh from [BBD+16], which requires $t_b(t_b+1)/2$ random bits per bit index. The total randomness consumed is $\sum_b t_b(t_b+1)/2$ bits.

Step 3: Masking conversion. The refreshed $(t_b + 1)$-bit BM is compressed into a $(t + 1)$-bit BM. Since $t \leq t_b$, the first t shares are set directly as $x_i[b] = y'_i[b]$ for $i \in [t]$, and the last share absorbs the remaining terms: $x_t[b] = \sum_{i=t}^{t_b} y'_i[b]$.

The proof of correctness is as follows. For all $b \in [k]$:

$$x_i[b] = y'_i[b] \;\text{ for } i \in [t], \qquad x_t[b] = \sum_{i=t}^{t_b} y'_i[b].$$

Hence,

$$\sum_{i=0}^{t} x_i[b] = \sum_{i=0}^{t-1} y'_i[b] \oplus \sum_{i=t}^{t_b} y'_i[b] = \sum_{i=0}^{t_b} y'_i[b] = x[b].$$

3.3 Example

We illustrate Algorithm 1 and Algorithm 2 through a concrete example achieving second-order bit security, with a 3-share BM and a 2-share IPM. The IPM is defined over $\mathbb{F}_2[X]/(X^4 + X + 1)$ with $L_0 = 1$ and $L_1 = X^2 + X$.

3.3.1 BM-to-IPM Conversion

The input is $\bar{x} = (x_0, x_1, x_2) \in \mathbb{F}_2[X]/(X^4 + X + 1)^3$ and the output is $\bar{z} = (z_0, z_1) \in \mathbb{F}_2[X]/(X^4 + X + 1)^2$. The randomness generation step requires $12 = (t + 1)(n - 1)k$ bits, used to sample three independent uniform elements $r_{0,1}, r_{1,1}, r_{2,1}$ from $\mathbb{F}_2[X]/(X^4 + X + 1)$.

Algorithm 1 BM-TO-IPM($\bar{x}$) - Bit t-MIMO-SNI BM-to-IPM Conversion

Input: BM shares $\bar{x} = (x_0, x_1, ..., x_t) \in \mathbb{F}_{2^k}^{t+1}$ such that $x = \sum_{i=0}^{t} x_i$
Output: IPM shares $\bar{z} = (z_0, z_1, ..., z_{n-1}) \in \mathbb{F}_{2^k}^{n}$ such that $x = \sum_{i=0}^{n-1} L_i z_i$

▷ Refresh Randomness
for $i = 0$ to t **do**
  for $j = 1$ to $n-1$ **do**
    $r_{i,j} \xleftarrow{\$} \mathbb{F}_{2^k}$
  end for
  $r_{i,0} \leftarrow \sum_{j=1}^{n-1} L_j r_{i,j}$
end for
▷ Compute z_0
for $i = 0$ to t **do**
  $y_i \leftarrow x_i + r_{i,0}$
end for
$z_0 \leftarrow \sum_{i=0}^{t} y_i$
▷ Compute $z_1, z_2, \ldots, z_{n-1}$
for $j = 1$ to $n-1$ **do**
  $z_j \leftarrow \sum_{i=0}^{t} r_{i,j}$
end for
return $\bar{z} = (z_0, z_1, ..., z_{n-1})$

Since $L_1 = X^2 + X$, multiplication by L_1 expands bit-wise as follows:

$$(L_1 r)[3] = r[1] \oplus r[2], \qquad (L_1 r)[2] = r[0] \oplus r[1] \oplus r[3],$$
$$(L_1 r)[1] = r[0] \oplus r[2], \qquad (L_1 r)[0] = r[2] \oplus r[3].$$

The output shares z_0 and z_1, expressed as bit-wise operations over the inputs and the sampled randomness, are:

$$\begin{aligned}
z_0[3] &= (x_0[3] \oplus r_{0,1}[1] \oplus r_{0,1}[2]) \oplus (x_1[3] \oplus r_{1,1}[1] \oplus r_{1,1}[2]) \oplus (x_2[3] \oplus r_{2,1}[1] \oplus r_{2,1}[2]),\\
z_0[2] &= (x_0[2] \oplus r_{0,1}[0] \oplus r_{0,1}[1] \oplus r_{0,1}[3]) \oplus (x_1[2] \oplus r_{1,1}[0] \oplus r_{1,1}[1] \oplus r_{1,1}[3])\\
&\quad \oplus (x_2[2] \oplus r_{2,1}[0] \oplus r_{2,1}[1] \oplus r_{2,1}[3]),\\
z_0[1] &= (x_0[1] \oplus r_{0,1}[0] \oplus r_{0,1}[2]) \oplus (x_1[1] \oplus r_{1,1}[0] \oplus r_{1,1}[2]) \oplus (x_2[1] \oplus r_{2,1}[0] \oplus r_{2,1}[2]),\\
z_0[0] &= (x_0[0] \oplus r_{0,1}[2] \oplus r_{0,1}[3]) \oplus (x_1[0] \oplus r_{1,1}[2] \oplus r_{1,1}[3]) \oplus (x_2[0] \oplus r_{2,1}[2] \oplus r_{2,1}[3]),
\end{aligned}$$

$$z_1[3] = r_{0,1}[3] \oplus r_{1,1}[3] \oplus r_{2,1}[3], \qquad z_1[2] = r_{0,1}[2] \oplus r_{1,1}[2] \oplus r_{2,1}[2],$$
$$z_1[1] = r_{0,1}[1] \oplus r_{1,1}[1] \oplus r_{2,1}[1], \qquad z_1[0] = r_{0,1}[0] \oplus r_{1,1}[0] \oplus r_{2,1}[0].$$

3.3.2 IPM-to-BM Conversion

The input is $\bar{z} = (z_0, z_1) \in \mathbb{F}_2[X]/(X^4 + X + 1)^2$ and the output is $\bar{x} = (x_0, x_1, x_2) \in \mathbb{F}_2[X]/(X^4 + X + 1)^3$.

We first perform the bit assignment step to satisfy $x[b] = \bigoplus_{i=0}^{t_b} y_i[b]$. This step only re-indexes bits and introduces no computational overhead; in a fixed-L implementation, the assignment pattern is a compile-time constant. For $\mathbb{F}_2[X]/(X^4+X+1)$ with $L_1 = X^2+X$, the bit assignments are:

$$\begin{aligned}
&y_0[3] = z_0[3], && y_1[3] = z_1[1], && y_2[3] = z_1[2], &&\\
&y_0[2] = z_0[2], && y_1[2] = z_1[0], && y_2[2] = z_1[1], && y_3[2] = z_1[3],\\
&y_0[1] = z_0[1], && y_1[1] = z_1[0], && y_2[1] = z_1[2], &&\\
&y_0[0] = z_0[0], && y_1[0] = z_1[2], && y_2[0] = z_1[3]. &&
\end{aligned}$$

Algorithm 2 IPM-TO-BM($\bar{z}$) - Bit t-MIMO-SNI IPM-to-BM Conversion

Input: IPM shares $\bar{z} = (z_0, z_1, ..., z_{n-1}) \in \mathbb{F}_{2^k}^n$ such that $x = \sum_{i=0}^{n-1} L_i z_i$
Output: BM shares $\bar{x} = (x_0, x_1, ..., x_t) \in \mathbb{F}_{2^k}^{t+1}$ such that $x = \sum_{i=0}^{t} x_i$

▷ Bit Assignments that satisfy $x[b] = \bigoplus_{i=0}^{t_b} y_i[b]$
for $b = 0$ to $k-1$ **do**
  $i \leftarrow 0$
  for $j = 0$ to $n-1$ **do**
    for $l = 0$ to $k-1$ **do**
      if $\mathrm{L}_j[b][l] = 1$ **then**
        $y_i[b] \leftarrow z_j[l]$
        $i \leftarrow i + 1$
      end if
    end for
  end for
  $t_b \leftarrow i - 1$
end for
▷ Refresh and Masking Conversion
for $b = 0$ to $k-1$ **do**
  $(y'_0[b], \ldots, y'_{t_b}[b]) \leftarrow SNI_Refresh_b(y_0[b], \ldots, y_{t_b}[b])$ ▷ t_b − SNI refresh
  for $i = 0$ to $t-1$ **do**
    $x_i[b] \leftarrow y'_i[b]$
  end for
  $x_t[b] \leftarrow 0$
  for $i = t$ to t_b **do**
    $x_t[b] \leftarrow x_t[b] \oplus y'_i[b]$
  end for
end for
return $\bar{x} = (x_0, x_1, ..., x_t)$

For each bit index b, we have $x[b] = \bigoplus_{i=0}^{t_b} y_i[b]$ since $\bigoplus_{i=0}^{t_b} y_i[b] = (z_0 + L_1 z_1)[b]$. We then apply an SNI refresh to each bit index. Since $t_0 = t_1 = t_3 = 2$ and $t_2 = 3$, the triples $(y_0[b], y_1[b], y_2[b])$ for $b \in \{0, 1, 3\}$ are fed into a 2-SNI refresh, and the quadruple $(y_0[2], y_1[2], y_2[2], y_3[2])$ is fed into a 3-SNI refresh, yielding refreshed values $y'_i[b]$ for all i, b.

The final output shares $x_i[b]$ are computed as:

$$
\begin{aligned}
&x_0[3] = y'_0[3], && x_1[3] = y'_1[3], && x_2[3] = y'_2[3],\\
&x_0[2] = y'_0[2], && x_1[2] = y'_1[2], && x_2[2] = y'_2[2] \oplus y'_3[2],\\
&x_0[1] = y'_0[1], && x_1[1] = y'_1[1], && x_2[1] = y'_2[1],\\
&x_0[0] = y'_0[0], && x_1[0] = y'_1[0], && x_2[0] = y'_2[0].
\end{aligned}
$$

3.4 Security Proofs

We prove that BM-TO-IPM and IPM-TO-BM each satisfy bit t-MIMO-SNI. As noted in Subsection 3.1, we treat BM-TO-IPM as a k-input, 1-output gadget, and IPM-TO-BM as a 1-input, k-output gadget. In the MIMO-SNI framework, one output probe corresponds to one share of a single output—i.e., one bit across all k outputs.

Theorem 1. *Algorithm 1 (BM-TO-IPM) is t-MIMO-SNI where* $t = d_L^{\perp} - 1$.

Proof. We enumerate all probe groups in Algorithm 1, and consider an arbitrary set of t_1 internal probes and t_2 output probes with $t_1 + t_2 \leq t$. We construct a set of input probe indices I with $|I| \leq t_1$ and show that all probes can be perfectly simulated from I.

The complete set of probe groups (with universal quantifiers $\forall i \in [t+1]$, $\forall j \in [n] \setminus \{0\}$, $\forall b \in [k]$ omitted) is:

(1) $x_i[b]$

(2) $r_{i,j}[b]$

(3) $(L_j r_{i,j})[b]$

(4) $(\sum_{m=0}^{j} L_m r_{i,m})[b]$

(5) $(x_i + r_{i,0})[b]$

(6) $(\sum_{m=0}^{i}(x_m + r_{m,0}))[b]$

(7) $(\sum_{m=0}^{i} r_{m,j})[b]$

Groups (1)–(5) are internal probes; groups (6) and (7) are internal probes when $i \neq t$ and output probes when $i = t$. We define the input probe index set I as a set of pairs (i, b), constructed as follows:

1. Initially I is empty.
2. For every probe of group (1), add (i, b) to I.
3. For every probe of group (2), add nothing to I.
4. For every probe of group (3), add (i, b) to I.
5. For every probe of group (4), add (i, b) to I.
6. For every probe of groups (5), (6), or (7), add nothing to I.

Since elements are added to I only in cases (1), (3), and (4), we have $|I| \leq t_1$. We now show that a simulator S with access to the inputs indexed by I can perfectly simulate all probe groups.

Group (1): The probe $x_i[b]$ is directly the input with index $(i, b) \in I$, so all group (1) probes can be simulated.

Groups (2), (3), (4), (7): These probes depend solely on randomness and are simulated by the simulator as independent uniform random values.

Group (5): Consider a probe $(x_i + r_{i,0})[b]$. The value $r_{i,0}[b]$ is a deterministic function of $\{r_{i,j}[\ell] : j \in [n] \setminus \{0\}, \ell \in [k]\}$ via the $\bar{L}$-IPM structure. We distinguish two cases.

Case A: No probe in groups (2), (3), (4), or (7) determines $r_{i,0}[b]$. By the bit t-probing security of $\bar{L}$-IPM, fewer than t probes of the random shares $r_{i,j}$ (groups (2) or (7)) are insufficient to determine $r_{i,0}[b]$ unless probes from groups (3) or (4) also constrain the same i. Since no such probe exists in this case, $r_{i,0}[b]$ is uniformly distributed from the simulator's view, and hence $(x_i + r_{i,0})[b]$ is uniform; the simulator outputs a fresh uniform random value.

Case B: Some probe in groups (3) or (4) constrains $r_{i,0}[b]$. Then $(i, b) \in I$ by construction, so the simulator knows $x_i[b]$. Moreover, the constraining probes from groups (3) and (4) have already been simulated (as they depend only on randomness), so the simulator can reconstruct $r_{i,0}[b]$ and compute $(x_i + r_{i,0})[b]$.

Group (6): The probe $(\sum_{m=0}^{i}(x_m + r_{m,0}))[b]$ is the cumulative XOR of group (5) values for indices $0, \ldots, i$. If none of the indices $(0, b), \ldots, (i, b)$ is constrained by groups (3) or (4) or (5), then each summand is uniformly random and independent (by Case A above), and the simulator assigns a fresh uniform random value to the cumulative sum. Otherwise, the simulator applies Case B for each constrained index and Case A for each unconstrained index, combining the results to produce the cumulative sum. □

Theorem 2. *Algorithm 2 (IPM-TO-BM) is t-MIMO-SNI where $t = d_{\bar{L}}^{\perp} - 1$.*

Proof. We enumerate all probe groups in Algorithm 2 and consider an arbitrary set of t_1 internal probes and, for each of the k outputs, an arbitrary set of t_2 output probes, with $t_1 + t_2 \leq t$. We use the t_b-SNI property of the per-bit refresh gadget $\mathsf{SNI_Refresh}_b$ to show that all probes can be simulated from at most t_1 input bits.

The complete probe groups (with quantifiers $\forall i \in [t]$ $(i \neq t)$, $\forall j \in [n]$, $\forall m \in [t_b]$, $\forall b \in [k]$ omitted) are:

(1) $z_j[b]$
(2) $y_m[b]$
(3) A wire in $\mathsf{SNI_Refresh}_b$
(4) $y'_m[b]$
(5) $x_i[*]$
(6) $x_t[*]$

Groups (1)–(4) are internal probes and groups (5), (6) are output probes. Since group (6) has no variable index, $t_2^{(6)}$ is either 0 or 1. Let the number of probes of the groups (1), (2), (3) and (4) be $t_1^{(1)}, t_1^{(2)}, t_1^{(3)}, t_1^{(4)}$, respectively, where $t_1 = \sum_\ell t_1^{(\ell)}$. And let the number of probes of the groups (5), (6) per output bit index be $t_2^{(5)}, t_2^{(6)}$ ($t_2^{(6)} = 0$ or 1), respectively, where $t_2 = t_2^{(5)} + t_2^{(6)}$. Then, the number of internal probes is t_1, the number of each output's probes is t_2 and we assume that $t_1 + t_2 \leq t$.

Since all groups (4), (5) and (6) probes are a value or XOR combinations of $\mathsf{SNI_Refresh}_b$ output bits, simulating the relevant $\mathsf{SNI_Refresh}_b$ outputs is sufficient to simulate groups (4), (5) and (6). For each bit index $b \in [k]$, the number of $\mathsf{SNI_Refresh}_b$ output bits needed is at most $t_1^{(4)} + t_2^{(5)} + t_2^{(6)}(t_b - t + 1)$ for simulation of groups (4)–(6) ($\because$ probes of the group (6) is XOR combinations of $(t_b - t + 1)$ bits of $\mathsf{SNI_Refresh}_b$ output).

Groups (1), (2) are input probes of $\mathsf{SNI_Refresh}_b$ and group (3) consists of internal probes of $\mathsf{SNI_Refresh}_b$. Then, $\mathsf{SNI_Refresh}_b$ has at most $(t_1^{(1)} + t_1^{(2)} + t_1^{(3)})$ bits internal probes and $t_1^{(4)} + t_2^{(5)} + t_2^{(6)}(t_b - t + 1)$ bits output probes.

Therefore, the total number of probes within $\mathsf{SNI_Refresh}_b$ is at most

$$(t_1^{(1)}+t_1^{(2)}+t_1^{(3)})+(t_1^{(4)}+t_2^{(5)}+t_2^{(6)}(t_b-t+1)) = t_1+(t_2^{(5)}+t_2^{(6)})+t_2^{(6)}(t_b-t) \leq t_1+t_2-t+t_b = t_b,$$

where the first equality uses $t_1 = t_1^{(1)} + t_1^{(2)} + t_1^{(3)} + t_1^{(4)}$ and the second inequality uses $t_2 = t_2^{(5)} + t_2^{(6)}$ and $0 \leq t_2^{(6)} \leq 1$. In other words, if we have at most the above-mentioned number of probes in $\mathsf{SNI_Refresh}_b$, then we can perfectly simulate all probe groups about $\mathsf{SNI_Refresh}_b$. By the t_b-SNI property of $\mathsf{SNI_Refresh}_b$, all probe groups can be simulated from at most $(t_1^{(1)} + t_1^{(2)} + t_1^{(3)})$ input bits of $\mathsf{SNI_Refresh}_b$.

Since groups (1), (2) and (3) are internal probes, the summation of the number of group (1)–(3) probes for $\forall b \in [k]$ is $(t_1^{(1)} + t_1^{(2)} + t_1^{(3)})$. In addition, input of $\mathsf{SNI_Refresh}_b$ is directly assigned from the input of Algorithm 2.

Therefore, the entire set of probes (1)–(6) can be simulated from at most $t_1^{(1)} + t_1^{(2)} + t_1^{(3)} (\leq t_1)$ input bits, completing the proof. □

4 Optimized Multiplication in Inner Product Masking

This section introduces an optimized IPM multiplication algorithm that reduces randomness and XOR gate count while preserving bit t-SNI security, building on the construction of [GD25a]. We present the design rationale, detailed construction and correctness, a worked example, the security proof, and a complexity comparison.

4.1 Design Overview

We first recall the IPM multiplication of [GD25a] and then describe two optimization ideas that reduce its randomness consumption.

Baseline: [GD25a] IPM multiplication. Given n-share IPM encodings $\bar{x}$ and $\bar{y}$ of x and y, the (unrefreshed) output share decomposition is:

$$z = xy = \left(\sum_{i=0}^{n-1} L_i x_i\right)\left(\sum_{i=0}^{n-1} L_i y_i\right),$$

which naturally decomposes into n output shares $z_0, \ldots, z_{n-1}$ satisfying $z = \sum_i L_i z_i$:

$$z_i = x_i y_0 + L_1 x_i y_1 + \cdots + L_{n-1} x_i y_{n-1}, \quad i = 0, \ldots, n-1.$$

To achieve bit t-SNI, each $\mathbb{F}_{2^k}$ multiplication $x_i y_j$ is computed in a bit-level matrix form. Specifically, a single field multiplication is expressed as the row-wise sum of a $\frac{k(k+1)}{2} \times k$ binary matrix $\tilde{\mathbf{C}}$ of bit-AND monomials (e.g., $x_i[a]y_j[b]$). Each row of $\tilde{\mathbf{C}}$ is independently refreshed with an $\mathsf{IPM}_{\mathsf{free}}(0)$ sharing to mask the intermediate values. Here, $\mathsf{IPM}_{\mathsf{free}}(0)$ denotes the XOR of t independent copies of an IPM sharing of zero. The gadget also applies a column-wise refresh across $(x_0y_0, x_1y_0, \ldots, x_{n-1}y_0)$ to ensure the final output shares are also masked. In total, $\frac{k(k+1)}{2} \times (n+1)$ instances of $\mathsf{IPM}_{\mathsf{free}}(0)$ are required, each costing $t(n-1)k$ random bits, giving a total of $\frac{1}{2}t(n^2-1)k^2(k+1)$ random bits per multiplication.

Optimization 1: Row Packing. The $\frac{k(k+1)}{2} \times k$ matrix $\mathbf{C}$ is generally sparse—not all positions are occupied, and shifting a monomial within its column does not affect correctness (since only the row sum matters). We therefore pack the monomials to reduce the number of rows from $\frac{k(k+1)}{2}$ to $W \leq \frac{k(k+1)}{2}$, directly reducing the number of $\mathsf{IPM}_{\mathsf{free}}(0)$ instances used and hence the total randomness from $t(n+1)\frac{k(k+1)}{2}k(n-1)$ to $t(n+1)Wk(n-1)$ bits. This transformation does not affect which bit-AND monomials are masked or the structure of the refreshes, so it does not affect security.

Optimization 2: Reduction in Rows. After Row Packing, some rows of the matrix still contain duplicate monomials (the same AND term appearing in multiple columns) or zero entries. In the baseline, each row refresh uses Wk independent random bits—one per matrix position. However, since positions with the same AND term share the same underlying value and zero positions need no masking, it suffices to use one random bit per *distinct* bit-AND monomial case. This reduces the random bits used per $\mathsf{IPM}_{\mathsf{free}}(0)$ instance from Wk to k^2. Note that this optimization is applied only to the nW refreshes corresponding to $(x_iy_1, \ldots, x_iy_{n-1})$ for each i; the W refreshes for the column-wise direction $(x_0y_0, \ldots, x_{n-1}y_0)$ are left unchanged to ensure every distinct bit-AND monomial in x_iy_0 receives an independent random bit. The combined effect of both optimizations reduces the total randomness from previous to:

$$t(n+1)Wk(n-1) = t(nWk(n-1) + Wk(n-1)) \geq t(nk^2(n-1) + Wk(n-1)) \text{ bits.}$$

Simplifying the equation gives $tk(n-1)(kn+W)$ bits.

4.2 Construction and Correctness

Before describing the algorithm, we introduce a few definitions to simplify notation.

For a target security order t and field $\mathbb{F}_{2^k}$, we apply Row Packing to redefine the basic binary multiplication matrix from $\mathbf{C} \in \mathbb{F}_2^{\frac{k(k+1)}{2} \times k}$ to $\mathbf{C}' \in \mathbb{F}_2^{W \times k}$ where $W \leq \frac{k(k+1)}{2}$. For each row index $w \in [W]$, we apply Reduction in Rows to define a subset $\mathbb{S}_w \subseteq \mathbb{F}_{2^k}$ of admissible random mask values.

IPM random shares of zero $\mathsf{IPM}(0)$ is defined as follows, where r_i is uniform random in $\mathbb{F}_{2^k}$:

$$\mathsf{IPM}(0) := \left(\sum_{i=1}^{n-1} L_i r_i,\ r_1, r_2, \ldots, r_{n-1} \right).$$

Each $\mathsf{IPM}^w(0)$ is defined analogously, where s_i is uniform random in $\mathbb{S}_w$:

$$\mathsf{IPM}^w(0) := \left(\sum_{i=1}^{n-1} L_i s_i,\ s_1, s_2, \ldots, s_{n-1} \right).$$

Following [GD25a], $\mathsf{IPM}_{\mathsf{free}}(0)$ is the XOR of t independent copies of $\mathsf{IPM}(0)$, and $\mathsf{IPM}_{\mathsf{free}}{}^w(0)$ is the XOR of t independent copies of $\mathsf{IPM}^w(0)$.

Finally, we define $\mathsf{FMM}(x, y) : \mathbb{F}_{2^k} \times \mathbb{F}_{2^k} \to \mathbb{F}_2^{W \times k}$ as the function that maps two field elements to their bit-AND monomial matrix in $\mathbf{C}'$ form. By construction, the row-wise sum of $\mathsf{FMM}(x, y)$ equals the bit representation of xy.

Our IPM multiplication follows the same three-gadget structure as [GD25a]: a preprocessing gadget (Gadget-1) generates all necessary mask matrices, a field multiplication gadget (Gadget-2) computes a single masked field multiplication, and a compression gadget (Gadget-3) calls Gadget-2 for all n^2 cross products and compresses the results. Compared to [GD25a], the differences are: (i) Gadget-1 uses W rows instead of $\frac{k(k+1)}{2}$, and (ii) the nW out of $(n+1)W$ refresh instances use $\mathsf{IPM}_{\mathsf{free}}{}^w(0)$ instead of $\mathsf{IPM}_{\mathsf{free}}(0)$.

Both $\mathsf{IPM}_{\mathsf{free}}(0)$ and $\mathsf{IPM}_{\mathsf{free}}{}^w(0)$ are IPM sharings of zero, so the correctness argument of [GD25a] applies without modification to our gadgets.

4.3 Example

We illustrate Gadget-1, Gadget-2, and Gadget-3 through a 2-share IPM multiplication over $\mathbb{F}_2[X]/(X^3 + X + 1)$. We first apply Row Packing and Reduction in Rows to this field.

Row packing. The baseline multiplication matrix $\mathbf{C} \in \mathbb{F}_2^{6 \times 3}$ and its row-packed version $\mathbf{C}' \in \mathbb{F}_2^{5 \times 3}$ are:

$$\mathbf{C} = \begin{bmatrix} x_0[0]y_0[2] & x_0[0]y_0[1] & x_0[0]y_0[0] \\ x_0[1]y_0[1] & x_0[1]y_0[0] & 0 \\ x_0[2]y_0[0] & 0 & 0 \\ 0 & x_0[1]y_0[2] & x_0[1]y_0[2] \\ 0 & x_0[2]y_0[1] & x_0[2]y_0[1] \\ x_0[2]y_0[2] & x_0[2]y_0[2] & 0 \end{bmatrix}, \quad \mathbf{C}' = \begin{bmatrix} x_0[0]y_0[2] & x_0[0]y_0[1] & x_0[0]y_0[0] \\ x_0[1]y_0[1] & x_0[1]y_0[0] & 0 \\ x_0[2]y_0[0] & x_0[1]y_0[2] & x_0[1]y_0[2] \\ 0 & x_0[2]y_0[1] & x_0[2]y_0[1] \\ x_0[2]y_0[2] & x_0[2]y_0[2] & 0 \end{bmatrix}$$

Here $\mathbf{C}$ has $\frac{k(k+1)}{2} = 6$ rows, while $\mathbf{C}'$ is packed to $W = 5$ rows.

Reduction in Rows. For security order $t = 3$, the uniform random bit matrix $\mathbf{S}$ (baseline) and the randomness-reduced matrix $\mathbf{S}'$ are:

Algorithm 3 (Gadget-1) - Mask Matrices Generation

Input: -
Output: $\mathbf{S} \in (\mathbb{F}_2^{W\times k})^{n\times n}$
▷ Compute $\bar{s}^{w,i} = (s_0^{w,i}, ..., s_{n-1}^{w,i})$ n-share IPMs of zero
for $i = 0$ to $n-1$ **do**
  for $w = 0$ to $W-1$ **do**
    $\bar{s}^{w,i} = (s_0^{w,i}, ..., s_{n-1}^{w,i}) \leftarrow \mathsf{IPM_{free}}^{w}(0) \in \mathbb{F}_{2^k}^n$
  end for
end for
▷ Compute $\bar{r}^{w} = (r_0^{w}, ..., r_{n-1}^{w})$ n-share IPMs of zero
for $w = 0$ to $W-1$ **do**
  $\bar{r}^{w} = (r_0^{w}, ..., r_{n-1}^{w}) \leftarrow \mathsf{IPM_{free}}(0) \in \mathbb{F}_{2^k}^n$
end for
▷ Compute partial mask matrices $\mathbf{S}_{i,j} \in \mathbb{F}_2^{W\times k}$
for $i = 0$ to $n-1$ **do**
  for $w = 0$ to $W-1$ **do**
    $\mathbf{R}_i[w, *] \leftarrow r_i^w$
    for $j = 0$ to $n-1$ **do**
      $\mathbf{S}_{i,j}[w, *] \leftarrow s_j^{w,i}$
    end for
  end for
  $\mathbf{S}_{i,0} \leftarrow \mathbf{S}_{i,0} \oplus \mathbf{R}_i$
end for
return $\mathbf{S} = \begin{bmatrix} \mathbf{S}_{0,0} & \cdots & \mathbf{S}_{0,n-1} \\ \vdots & \ddots & \vdots \\ \mathbf{S}_{n-1,0} & \cdots & \mathbf{S}_{n-1,n-1} \end{bmatrix}$

$$\mathbf{S} = \begin{bmatrix} s_0[2] & s_0[1] & s_0[0] \\ s_1[2] & s_1[1] & s_1[0] \\ s_2[2] & s_2[1] & s_2[0] \\ s_3[2] & s_3[1] & s_3[0] \\ s_4[2] & s_4[1] & s_4[0] \end{bmatrix}, \qquad \mathbf{S}' = \begin{bmatrix} s_0[2] & s_0[1] & s_0[0] \\ s_1[1] & s_1[0] & 0 \\ s_2[1] & s_2[0] & s_2[0] \\ 0 & s_3[0] & s_3[0] \\ s_4[0] & s_4[0] & 0 \end{bmatrix}$$

The baseline $\mathbf{S}$ requires 15 random bits ($Wk = 5 \times 3$), whereas $\mathbf{S}'$ requires only 9 bits ($k^2 = 3^2$), reducing randomness from $Wk = 15$ to $k^2 = 9$ bits per refresh instance.

Gadget-1. Gadget-1 generates the mask matrices. For each row w, it samples from $\mathbb{S}_w$ according to the structure of $\mathbf{S}'$: row 0 uses 3 independent uniform random bits; row 1 uses a single uniform random bit replicated across its two nonzero positions; and so on. Gadget-1 generates $\bar{s}^{w,j}$ instances using $\mathsf{IPM_{free}}^{w}(0)$ and $\bar{r}^{w}$ instances using $\mathsf{IPM_{free}}(0)$, combining them to produce $\mathbf{S} \in (\mathbb{F}_2^{5\times 3})^{2\times 2}$.

Gadget-2. Gadget-2 applies $\mathsf{FMM}(x_i, y_j)$ to obtain $\mathbf{C}'$ and XORs it with the corresponding $\mathbf{S}_{i,j}$ block. Since x_0y_0 and x_0y_1 use a fully populated $\mathbf{S}$ (no optimization applied), all rows are refreshed uniformly. For x_1y_j ($i \neq 0$), the Reduction in Rows applies, so zero entries in $\mathbf{C}'$ are left unrefreshed and duplicate terms share random bits. The following matrices illustrate the masked $\tilde{\mathbf{C}}$ for x_0y_0 and x_1y_0 respectively:

Algorithm 4 (Gadget-2) - Field Multiplication with Mask

Input: $x, y \in \mathbb{F}_{2^k}, \mathbf{S} \in \mathbb{F}_2^{W \times k}$
Output: $z = xy + s \in \mathbb{F}_{2^k}$ where $s = \sum_{w=0}^{W-1} \mathbf{S}[w, *]$
▷ Compute field multiplication matrix $\mathbf{C} \in \mathbb{F}_2^{W \times k}$
1: $\mathbf{C} \leftarrow \mathsf{FMM}(x, y)$ ▷ $xy = \sum_{w=0}^{W-1} \mathbf{C}[w, *]$
2: $\tilde{\mathbf{C}} \leftarrow \mathbf{C} \oplus \mathbf{S}$
▷ Compute z
3: $z \leftarrow \sum_{w=0}^{W-1} \tilde{\mathbf{C}}[w, *]$
4: **return** z

Algorithm 5 (Gadget-3) - IPM Field Multiplication

Input: $\bar{x}, \bar{y} \in \mathbb{F}_{2^k}^n$, $\mathbf{S} \in (\mathbb{F}_2^{W \times k})^{n \times n}$ such that $x = \sum_{i=0}^{n-1} L_i x_i,\ y = \sum_{i=0}^{n-1} L_i y_i$
Output: $\bar{z} \in \mathbb{F}_{2^k}^n$ such that $xy = \sum_{i=0}^{n-1} L_i z_i \in \mathbb{F}_{2^k}$
▷ Compute $z_{i,j} = x_i y_j + s_{i,j}$
1: **for** $i = 0$ to $n - 1$ **do**
2: **for** $j = 0$ to $n - 1$ **do**
3: $z_{i,j} \leftarrow$ Gadget-2$(x_i, y_j, \mathbf{S}_{i,j})$
4: **end for**
5: **end for**
▷ Compress to z_i
6: $z_i \leftarrow \sum_{j=0}^{n-1} L_j z_{i,j}$
7: **return** z

$$\tilde{\mathbf{C}}_{x_0 y_0} = \begin{bmatrix} x_0[0]y_0[2] \oplus \mathbf{S}[0,0] & x_0[0]y_0[1] \oplus \mathbf{S}[0,1] & x_0[0]y_0[0] \oplus \mathbf{S}[0,2] \\ x_0[1]y_0[1] \oplus \mathbf{S}[1,0] & x_0[1]y_0[0] \oplus \mathbf{S}[1,1] & \mathbf{S}[1,2] \\ x_0[2]y_0[0] \oplus \mathbf{S}[2,0] & x_0[1]y_0[2] \oplus \mathbf{S}[2,1] & x_0[1]y_0[2] \oplus \mathbf{S}[2,2] \\ \mathbf{S}[3,0] & x_0[2]y_0[1] \oplus \mathbf{S}[3,1] & x_0[2]y_0[1] \oplus \mathbf{S}[3,2] \\ x_0[2]y_0[2] \oplus \mathbf{S}[4,0] & x_0[2]y_0[2] \oplus \mathbf{S}[4,1] & \mathbf{S}[4,2] \end{bmatrix}$$

$$\tilde{\mathbf{C}}_{x_1 y_0} = \begin{bmatrix} x_1[0]y_0[2] \oplus \mathbf{S}[0,0] & x_1[0]y_0[1] \oplus \mathbf{S}[0,1] & x_1[0]y_0[0] \oplus \mathbf{S}[0,2] \\ x_1[1]y_0[1] \oplus \mathbf{S}[1,0] & x_1[1]y_0[0] \oplus \mathbf{S}[1,1] & 0 \\ x_1[2]y_0[0] \oplus \mathbf{S}[2,0] & x_1[1]y_0[2] \oplus \mathbf{S}[2,1] & x_1[1]y_0[2] \oplus \mathbf{S}[2,1] \\ 0 & x_1[2]y_0[1] \oplus \mathbf{S}[3,1] & x_1[2]y_0[1] \oplus \mathbf{S}[3,1] \\ x_1[2]y_0[2] \oplus \mathbf{S}[4,0] & x_1[2]y_0[2] \oplus \mathbf{S}[4,0] & 0 \end{bmatrix}$$

In the $x_1 y_0$ case, duplicate AND terms share the same random mask and zero positions are left unmasked, reducing the XOR count in the row-summation step.

Gadget-3. Gadget-3 simply calls Gadget-2 for all four cross products $z_{0,0}, z_{0,1}, z_{1,0}, z_{1,1}$ and computes $z_0 = z_{0,0} \oplus L_1 z_{0,1}$ and $z_1 = z_{1,0} \oplus L_1 z_{1,1}$.

4.4 Security Proofs

The bit t-SNI proof for our optimized IPM multiplication follows directly from Theorem 2 of [GD25a]. We verify that the four key properties used in that proof are preserved:

1. For each i, the multiplication matrix for $(x_i y_0, x_i y_1, \ldots, x_i y_{n-1})$ is refreshed with independent $\mathsf{IPM}(0)$ sharings per row.

2. The multiplication matrix for $(x_0 y_0, x_1 y_0, \ldots, x_{n-1} y_0)$ is refreshed with independent $\mathsf{IPM}(0)$ sharings per row.

3. Every distinct bit-AND term is refreshed with an independent random bit.

4. All $\mathsf{IPM}(0)$ instances satisfy the $(t-1)$-free condition (i.e., they are $\mathsf{IPM}_{\mathsf{free}}(0)$).

Row Packing reduces the number of rows from $\frac{k(k+1)}{2}$ to W but preserves per-row independence, leaving properties (1) and (2) intact. The refresh structure and the free condition are unaffected. Reduction in Rows modifies the per-row random mask matrix but does not affect per-row independence; it is applied only to the nW instances not covering $x_i y_0$, so property (2) is preserved.

For property (3), observe that the Reduction in Rows assigns random mask bits based on the distinct bit-AND monomials appearing in each row: positions containing the same monomial $x_i[a]y_j[b]$ receive the same mask bit, positions containing zero receive no mask, and positions containing distinct monomials receive independent mask bits. Consequently, after XOR with the mask matrix, each distinct bit-AND monomial is refreshed with its own independent random bit, while identical monomials sharing the same mask cancel consistently under row summation, preserving correctness. Thus property (3) is maintained.

Property (4) is maintained since $\mathsf{IPM}_{\mathsf{free}}{}^{w}(0)$ is constructed as the XOR of t independent $\mathsf{IPM}^{w}(0)$ instances and thus satisfies the free condition by the same argument as $\mathsf{IPM}_{\mathsf{free}}(0)$. Since all four properties hold, Theorem 2 of [GD25a] applies and our optimized multiplication gadget satisfies bit t-SNI.

4.5 Complexity Comparison

We compare our optimized gadgets against the ISW multiplication for BM [ISW03] and the IPM multiplication gadgets of Gaspoz and Dhooghe [GD25a], in terms of bit-level AND gates, XOR gates, and fresh random bits for an n-share multiplication over $\mathbb{F}_{2^k}$. Following the evaluation methodology of [GD25a], we separate the preprocessing cost (Gadget-1) from the online multiplication cost (Gadget-3), as Gadget-1 can in principle be precomputed or amortized across invocations.

To keep the asymptotic formulas concise, we adopt the following assumptions (consistent with [GD25a]): an $\mathbb{F}_{2^k}$ multiplication costs k^2 AND gates and k^2 XOR gates, and multiplication by a constant L_i costs k^2 XOR gates (a loose bound; in practice a $k \times k$ binary matrix-vector product uses at most $k(k-1)$ XOR gates).

Under these assumptions, we re-evaluate the complexity of the [GD25a] gadgets from a bitslice implementation perspective, and derive the complexity of our proposed gadgets accordingly in Table 2. Note that while ISW has lower cost for the same share count, IPM multiplication achieves a strictly higher security order for the same number of shares, which must be taken into account when comparing at the same bit security level.

All multiplications require n^2 cross products, each involving two k-bit elements whose AND gate count is n^2k^2 in all cases. For XOR gates, our Gadget-3 and [GD25a]'s Gadget-3 differ only in replacing $\frac{k(k+1)}{2}$ with W; the actual implementation further reduces XOR count for $j > 0$ cross products, since zero entries in $\mathbf{C}'$ skip the corresponding row-sum XOR and duplicate terms reuse the same random value. For Gadget-1, the row count reduction from $\frac{k(k+1)}{2}$ to W, combined with the use of $\mathsf{IPM}_{\mathsf{free}}{}^{w}(0)$, jointly reduce the random bit count. Since random bit consumption is independent of field reduction and $\bar{L}$ parameters, the values in Table 2 match the exact implementation costs.

Table 3 reports concrete gate counts for instantiations achieving the same bit security order, using the field $\mathbb{F}_2[X]/(X^4 + X + 1)$ (where Row Packing achieves $W = 7$) and IPM parameters $L_0 = 1$, $L_1 = X^2 + X$, $L_2 = X^2 + X + 1$.

At $t = 2$, Our Gadget-3 achieves lower AND and XOR counts than ISW, and Our Gadget-1 consumes half the random bits of [GD25a]'s Gadget-1, with a proportional reduction in XOR gates. At $t = 5$, the gap between ISW and our Gadget-3 widens substantially—our Gadget-3 requires less than half the total gate count of ISW—and Our

Table 2: Comparison in the number of AND, XOR gates, and fresh random bits of ISW multiplication, Gadget-3, Gadget-1, Our Gadget-3, and Our Gadget-1

Method	#ANDs	#XORs	#Random bits
ISW mul (BM)	n^2k^2	$n^2k^2 + 2nk(n-1)$	$\frac{1}{2}nk(n-1)$
Gadget-3 [GD25a]	n^2k^2	$n^2(k^2+k)k + n^2k^2$	–
Our Gadget-3	n^2k^2	$n^2(2W)k + n^2k^2$	–
Gadget-1 [GD25a]	–	$\frac{1}{2}k(k+1)(tk^2(n^2-1) + n^2kt)$	$\frac{1}{2}t(n^2-1)k^2(k+1)$
Our Gadget-1	–	$W(tk^2(n^2-1) + n^2kt)$	$tk(n-1)(kn+W)$

Gadget-1 reduces random bits and XOR gates by more than half compared to [GD25a]. Despite this significant overhead reduction, Gadget-1 remains the bottleneck of the IPM multiplication; further optimization is left for future work.

Table 3: Implementation cost in the number of AND, XOR gates, and fresh random bits of ISW multiplication, Gadget-3, Gadget-1, Our Gadget-3, and Our Gadget-1 in $\mathbb{F}_2[X]/(X^4+X+1)$

Security order	Gadget	#ANDs	#XORs	#Random bits
$t=2$	ISW mul (BM)	144	183	12
	Gadget-3 [GD25a]	64	250	–
	Our Gadget-3	64	178	–
	Gadget-1 [GD25a]	–	620	240
	Our Gadget-1	–	322	120
$t=5$	ISW mul (BM)	576	780	60
	Gadget-3 [GD25a]	144	576	–
	Our Gadget-3	144	396	–
	Gadget-1 [GD25a]	–	4840	1600
	Our Gadget-1	–	2305	760

5 Hybrid IPM-BM and Its Application to AES

This section describes the hybrid IPM-BM masking scheme and its application to AES.

We begin by articulating the complementary advantages of IPM and BM. IPM is particularly well-suited to multiplications, whose cost scales quadratically in the number of shares, because it achieves a target security order with fewer shares than BM. BM, on the other hand, benefits from structural simplicity: since each bit of the secret is shared independently, operations such as bit shifts and bit swaps can be applied share-wise at no extra cost, trivially satisfying PINI without consuming additional randomness. This stands in contrast to IPM and CM, where cross-bit dependencies in the sharing mean that such operations require explicit randomness. For example, the CM-ADD and CM-SWAP gadgets proposed in [GD25b] consume random bits to handle XOR and SWAP operations between bits at different indices, whereas in BM these operations reduce to share-wise computation. This structural limitation poses a significant barrier to deploying IPM or CM in ciphers such as AES, which perform numerous Boolean linear operations per round. We therefore propose a *hybrid IPM-BM* scheme that combines the strengths of both: multiplications, with their quadratic share-count cost, are performed in the IPM with a reduced share count, while all remaining Boolean linear operations are performed in BM at no additional randomness cost. The bridge between the two domains is provided by the MIMO-SNI conversion gadgets of Section 3.

Field selection and operation classification. To apply the hybrid IPM-BM to AES, we first select the field over which IPM operates and classify operations by domain. We choose the 4-bit field $\mathbb{F}_{2^4}$. XOR operations between bits of the same field index are linear even in IPM (since L_i is a field element, same-index XOR corresponds to adding two sharings), so they can be handled share-wise in IPM and trivially satisfy PINI. Accordingly, IPM handles multiplications and same-index XOR operations. All remaining Boolean linear operations—including XOR between bits at different field indices, bit permutations, bit shifts, and byte-level linear maps—are performed in BM.

AES round function decomposition. AES consists of AddRoundKey, SubBytes, ShiftRows, and MixColumns. All operations except SubBytes are Boolean linear. AddRoundKey applies XOR with the round key, which involves same-field-index XOR and could in principle be performed in IPM; however, since SubBytes begins with a field isomorphism that maps from the AES field to the composite field $\mathbb{F}_{(2^4)^2}$, performing AddRoundKey in IPM would require a BM-to-IPM conversion immediately thereafter. To avoid this overhead, we assign AddRoundKey, ShiftRows, and MixColumns to BM, while SubBytes takes BM as its input and output with internal multiplications computed in IPM.

S-box structure and security proof. Figure 2 shows the hybrid masking structure of the AES S-box satisfying bit t-PINI. Following the approach of Canright [Can05], we select the field isomorphism $\phi : \mathbb{F}_{2^8} \to \mathbb{F}_{(2^4)^2}$ by exhaustively evaluating all valid mappings under polynomial and normal bases, minimizing the combined Hamming weight of the isomorphism matrix and the merged inverse-isomorphism–affine-transform matrix. The isomorphism, $N \cdot x^2$, x^2, x^4, inverse isomorphism, and affine transform functions are all Boolean linear; they are therefore implemented as trivial BM input/output functions by applying the operation share-wise across $t+1$ shares, trivially satisfying bit t-PINI. The BM-to-IPM and IPM-to-BM functions are Algorithm 1 and Algorithm 2, respectively, and satisfy bit t-MIMO-SNI. The $\oplus$ operation is configured to XOR corresponding bits of two IPM shares, trivially satisfying bit t-PINI. The Mul function is the combination of Algorithm 3 and Algorithm 5 and satisfies bit t-SNI. By applying an SNI refresh (or SNI gadget) to at least one input of Mul, and constructing all other sub-gadgets as PINI or MIMO-SNI, we guarantee SP-NIB for the full S-box circuit excluding the outermost isomorphism and inverse isomorphism by the SP-NIB definition (Definition 8). We now verify the three conditions of Proposition 3 for this partial gadget.

Condition (i): the partial gadget is SP-NIB by construction, since every internal gadget is NI or better, every multiplication input is SNI-refreshed, and the simplified computation graph contains at most one path between any pair of vertices.

Condition (ii): the back-end conversion gadgets (IPM-to-BM) produce independent BM outputs from disjoint sub-circuit branches; hence no pair of output nodes shares a common predecessor.

Condition (iii): similarly, the front-end conversion gadgets (BM-to-IPM) receive independent BM inputs, so no pair of input nodes shares a common successor within the partial gadget.

Since all three conditions are satisfied, the partial gadget is t-MIMO-SNI by Proposition 3. Consequently, the partial gadget is bit t-PINI by Proposition 2. Since the outermost isomorphism and inverse isomorphism are trivial bit t-PINI share-wise operations, the whole S-box is bit t-PINI by Proposition 1.

Since all sub-gadgets operate on individual bits as the basic unit, we exploit bitslice techniques to process all 16 bytes of the AES-128 state in parallel.

Comparison with pure code-based masking. A natural alternative to our hybrid approach is to implement the entire AES using only CM gadgets, as proposed by Gaspoz and

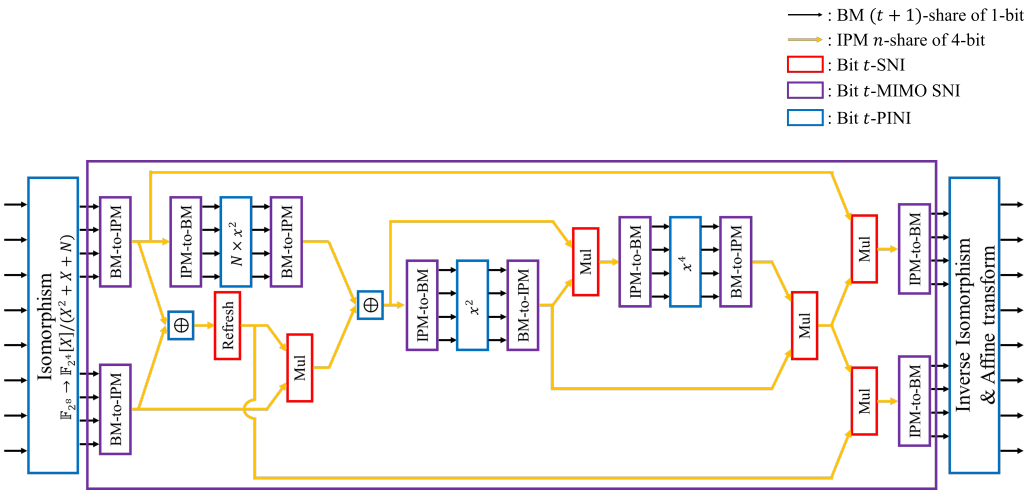


Figure 2: Bit t-PINI AES S-box design with hybrid IPM-BM

Dhooghe [GD25b]. Their CM framework provides bit t-PINI gadgets for all required operations, including CM-ADD (XOR between bits at different indices) and CM-SWAP. However, these gadgets consume fresh randomness and many XOR gates, leading to significant overhead when applied to ciphers with many Boolean linear operations such as AES. Table 4 compares the costs of the Boolean linear operations in the AES S-box under the pure CM approach of [GD25b] and our hybrid IPM-BM approach, both achieving the same bit PINI security order $t = 2$.

We compute the XOR gates and random bits of pure CM using Table 1 of [GD25b]: CM-ADD costs 67 XOR gates and 48 random bits, and each CM-SWAP costs twice as much as a CM-ADD. Accordingly, the x^2 operation over $\mathbb{F}_{2^4}$ has a simple structure consisting of 2 bit-XOR and 2 bit-SWAP, which in pure CM requires 2 CM-ADD and 2 CM-SWAP. Similarly, the $N \times x^2$ operation requires 4 CM-ADD and 2 CM-SWAP, the x^4 operation requires 4 CM-ADD, the isomorphism requires 14 CM-ADD and 3 CM-SWAP, and the inverse isomorphism combined with the affine transform requires 29 CM-ADD and 1 CM-SWAP.

In the hybrid approach, each operation is bracketed by BM-to-IPM and IPM-to-BM conversions, with the BM operation itself performed share-wise. Since our conversion gadgets operate on a 4-bit unit (matching the field $\mathbb{F}_{2^4}$ on which the IPM is defined), 4-bit operations (x^2, $N \times x^2$, x^4) require a single conversion pair, whereas 8-bit operations spanning $\mathbb{F}_{2^8}$ (the isomorphism and the combined inverse-isomorphism with the affine transform) require two conversion pairs. The BM-to-IPM conversion costs 43 XOR gates and 12 random bits per pair, and the IPM-to-BM conversion costs 31 XOR gates and 15 random bits per pair; the per-pair conversion overhead is thus 74 XOR gates and 27 random bits, doubled to 148 XOR gates and 54 random bits for 8-bit operations. Since x^2 in 3-share BM is simply a threefold repetition of the unmasked x^2, it contributes only $2 \times 3 = 6$ XOR gates. Applying the same share-wise reasoning to the other operations yields 4×3 XOR gates for $N \times x^2$, 4×3 XOR gates for x^4, 14×3 XOR gates for the isomorphism, and 29×3 XOR gates for the inverse isomorphism combined with the affine transform.

As summarized in Table 4, the hybrid approach reduces the XOR gate count by 3–9× and the random bit consumption by 7–28× across the Boolean linear operations in the S-box, with the gap widening for more complex operations such as the isomorphism and the inverse-isomorphism combined with the affine transform. Since the hybrid approach handles all Boolean linear operations share-wise in BM—requiring only BM-to-IPM and IPM-to-

Table 4: Cost comparison of second-order masked Boolean linear operations in the AES S-box: pure CM [GD25b] vs. hybrid IPM-BM (this work). Both implementations achieve bit 2-PINI. Operations spanning $\mathbb{F}_{2^8}$ (isomorphism, InvIso + Affine transform) require two pairs of 4-bit IPM-BM conversions in the hybrid approach.

Approach	Operation	#XORs	#Random bits
Pure CM [GD25b]	x^2 over $\mathbb{F}_{2^4}$	402	288
	$N \times x^2$ over $\mathbb{F}_{2^4}$	536	384
	x^4 over $\mathbb{F}_{2^4}$	268	192
	Isomorphism	1340	960
	InvIso + Affine transform	2077	1488
Hybrid IPM-BM (ours)	x^2 over $\mathbb{F}_{2^4}$	80	27
	$N \times x^2$ over $\mathbb{F}_{2^4}$	86	27
	x^4 over $\mathbb{F}_{2^4}$	86	27
	Isomorphism	190	54
	InvIso + Affine transform	235	54

BM conversion pairs to bracket each segment of linear operations—the cost advantage over pure CM grows with the number of such operations. In particular, Table 4 bounds each entry with a full conversion cycle for a fair per-operation comparison; in the actual hybrid IPM-BM AES implementation, however, AddRoundKey, ShiftRows, and MixColumns are also handled share-wise in BM, so the isomorphism located at the S-box input requires only two BM-to-IPM conversions (with no preceding IPM-to-BM), and the inverse-isomorphism combined with the affine transform located at the S-box output requires only two IPM-to-BM conversions (with no following BM-to-IPM). Consequently, the actual cost advantage is even larger than Table 4 suggests, and widens further when considering Boolean linear functions outside the S-box such as MixColumns.

6 Practical Side-Channel Leakage Evaluation

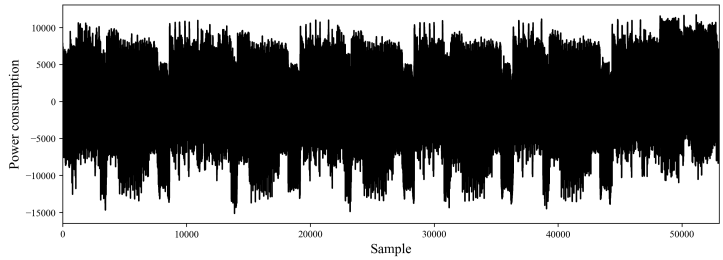
As a concrete instantiation of the hybrid IPM-BM architecture described in Section 5, we implement a second-order masked AES-128 on an ARM Cortex-M4 microcontroller and experimentally validate the side-channel security of the proposed conversion gadgets, IPM multiplication, and the overall hybrid construction.

6.1 Experimental Setup

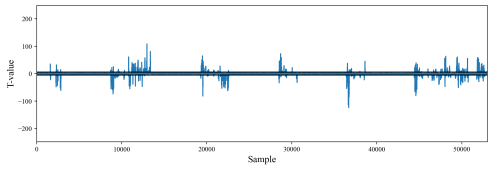
We evaluate the side-channel leakage of the first round of a second-order masked AES-128 encryption. The IPM is instantiated over $\mathbb{F}_2[X]/(X^4 + X + 1)$ with parameter $L_0 = 1$, $L_1 = X^2 + X$, yielding a 2-share IPM that achieves second-order bit-probing security (cf. Table 1). All operations are implemented using a bitslice technique so that all 16 state bytes are processed in parallel. Ueno et al. noted that the choice of pseudo-random generator is critical for practical leakage in masked implementations [UHIM24]. We select the Mersenne Twister, an improved linear feedback shift register (LFSR) satisfying high-dimensional uniform equidistribution [MN98].

Measurements are collected on a CW308T-STM32F405 target board running at 46 MHz. Power traces are acquired using a LeCroy HDO6104 oscilloscope at a sampling rate of 100 MS/s.

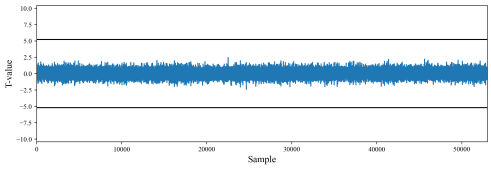
We assess leakage using the non-specific fixed-vs-random t-test methodology of TVLA [GJJR11, SM15]. Fixed and random plaintexts are interleaved uniformly at random, and the first-round power traces are partitioned into fixed and random sets; a sample-wise



(a) Mean power trace of second order IPM-BM AES round 1



(b) TVLA result when RNG is off using 100K traces (Threshold: 5.21, Maximum t-value: 124.10)



(c) TVLA result when RNG is on using 1M traces (Threshold: 5.21, Maximum t-value: 2.45)

Figure 3: First order TVLA results of second order IPM-BM AES round 1

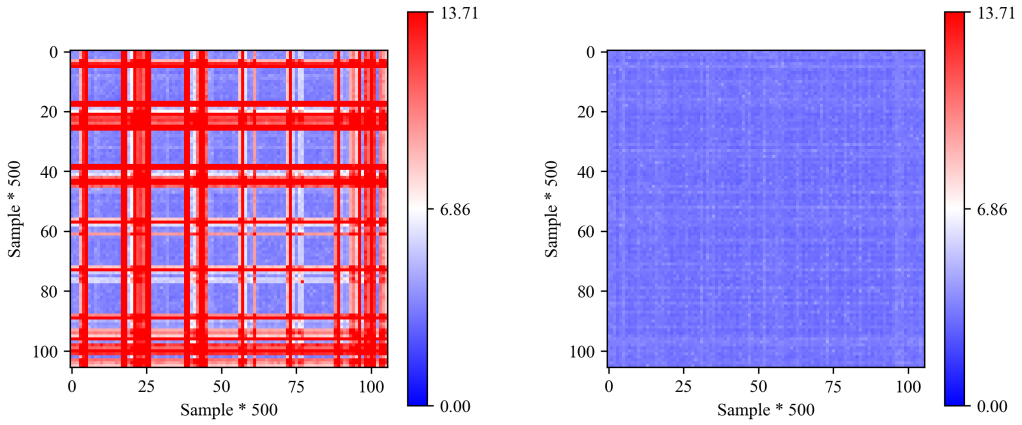
Welch's t-statistic is then computed to test whether the two populations share the same mean. Rejection of the null hypothesis indicates the presence of exploitable leakage.

While a threshold of ± 4.5 (corresponding to a significance level less than $\alpha = 0.00001$) is widely used in the literature, it is known to yield false positives at high rates when applied to traces with many sample points (described in Table 1 of [DZD+17] and Appendix A of [BGG+14]). We therefore apply a multiplicity correction and report thresholds derived from a family-wise significance level of $\alpha = 0.01$. For the first-order evaluation, the 53 000-sample trace gives a corrected threshold of 5.21. For the second-order bivariate evaluation, the $53000^2/2$ sample-pair space gives a corrected threshold of 6.86.

6.2 Results

First-order TVLA. Figure 3a shows the mean power trace of 100 first-round AES measurements. Figure 3b presents the first-order TVLA result when the random number generator is disabled (i.e., all mask values are set to zero), using 100 000 traces. Multiple sample points far exceed the corrected threshold of 5.21 (maximum t-value: 124.10), confirming that the measurement environment is sensitive to secret-dependent power variations and establishing the soundness of the evaluation setup. Figure 3c shows the result with the RNG enabled, using 1 000 000 traces. The maximum t-value across all 53 000 sample points is 2.45, well below the threshold of 5.21, providing no statistical evidence of first-order leakage.

Second-order TVLA. For the second-order bivariate evaluation, we apply the centered-product combination function to all pairs of sample points. To make the full bivariate analysis tractable, sample points are grouped into 500×500-sample blocks, and the maximum t-value within each block is plotted, yielding a 106×106 grid of block-wise maxima. Figure 4a shows the result with the RNG disabled using 100 000 traces: the maximum block-wise t-value (39.90) far exceeds the corrected threshold of 6.86, validating the sensitivity of the second-order setup. Figure 4b presents the result with the RNG enabled using 1 000 000 traces. The maximum t-value across all $53000^2/2$ sample-pair



(a) TVLA result when RNG is off using 100K traces (Threshold: 6.86, Maximum t-value: 39.90)

(b) TVLA result when RNG is on using 1M traces (Threshold: 6.86, Maximum t-value: 4.52)

Figure 4: Second order TVLA results of second order IPM-BM AES round 1

combinations is 4.52, below the corrected threshold of 6.86, providing no statistical evidence of second-order leakage.

The absence of detectable first- and second-order leakage across one million traces confirms that the proposed BM-to-IPM and IPM-to-BM conversion gadgets, the optimized IPM multiplication, and the hybrid masked AES collectively maintain their theoretical bit-probing security guarantees in a real-world implementation. These experimental results contradict the previously reported degradation of security order due to transition-based leakage in low-noise microcontrollers [BGG+14, PV17, DDT20], and demonstrate that the proposed algorithms are robustly secure even against transition-based leakage.

Implementation cost. Table 5 reports the cycle and random bit count breakdown of the hybrid second-order masked SubBytes of AES on the CW308T-STM32F405 at 46 MHz.

Table 5: Cycle and random bit count breakdown of the hybrid second-order masked SubBytes of AES-128 on ARM Cortex-M4.

Component	Cycles	Prop.	Random bits(×16)	Prop.
BM linear operations (share-wise)	858	3.9%	–	–
BM-to-IPM conversion	1620	7.4%	60	8.1%
IPM-to-BM conversion	955	4.4%	75	10.1%
Our Gadget-1 of IPM mul	13130	59.9%	600	80.8%
Our Gadget-3 of IPM mul	5270	24.0%	–	–
Others (IPM refresh, XOR)	348	1.6%	8	1.1%
Total (SubBytes)	21922	100%	743	100%

Since the cycles of components are measured by triggering target component, there may be minor measurement errors caused by the trigger mechanism or the compilation process. For comparison, a standard 3-share BM second-order masked SubBytes using ISW multiplication on the same structure of Figure 2 requires approximately 8 510 cycles. The current hybrid implementation incurs a factor of approximately 2.58× cycle overhead relative to the BM baseline, primarily due to the IPM multiplication preprocessing (Gadget-1). Gadget-1 consumes 59.9% of cycles and 80.8% of random bits for whole SubBytes.

Excluding Gadget-1, the remaining SubBytes operations require 8 792 cycles, which is comparable to the BM baseline (8 510 cycles). Furthermore, as shown in Table 3, the online phase of IPM multiplication (Gadget-3) outperforms ISW multiplication at security orders $t \geq 5$, suggesting that the hybrid approach becomes increasingly competitive at higher security orders, provided that the Gadget-1 preprocessing cost is amortized or reduced. Reducing the IPM multiplication including Gadget-1 cost is the key to closing this gap in future work. The full encryption requires 242,218 cycles in total and consumes 7,430 random bits per state byte (multiplied by 16 for the full 16-byte state).

7 Conclusion

We have addressed three open challenges in applying inner product masking to full cryptographic implementations.

First, we proposed BM-to-IPM and IPM-to-BM conversion gadgets satisfying bit t-MIMO-SNI in the bit-probing model, enabling provably secure composition between IPM and BM while preserving IPM's security order amplification. Second, we introduced two optimizations—Row Packing and Reduction in Rows—to the IPM multiplication gadget [GD25a], reducing its random bit consumption from $\frac{1}{2}t(n^2-1)k^2(k+1)$ to $tk(n-1)(kn+W)$ bits and proportionally reducing the XOR gate count, while maintaining bit t-SNI security. Third, we proposed the hybrid IPM-BM framework and demonstrated its applicability to AES: we implemented a hybrid 2-share IPM / 3-share BM masked AES-128 proven to satisfy bit 2-PINI. As summarized in Table 4, across the Boolean linear operations in the S-box, the hybrid approach reduces the XOR gate count by 3–9× and the random bit consumption by 7–28× compared to the pure CM construction of [GD25b], with the advantage widening for more complex operations such as the field isomorphism and the combined inverse-isomorphism and affine transform. We evaluated the practical second-order security on an ARM Cortex-M4 via first- and second-order TVLA over 1 000 000 traces with no detectable leakage at either order. A per-component cycle breakdown of the hybrid SubBytes revealed that the IPM multiplication preprocessing (Gadget-1) accounts for approximately 60% of the total cost and over 80% of the randomness, identifying it as the dominant bottleneck. To the best of our knowledge, this is the first end-to-end cryptographic implementation that provably preserves IPM's security order amplification in the bit-probing model.

Several directions remain for future work. First, the Gadget-1 preprocessing is the primary source of the performance gap between the hybrid construction and optimized BM-only implementations (approximately 2.58× in SubBytes cycle count). Reducing this overhead through structural improvements is the most impactful open problem for making IPM-based masking competitive in practice. Second, we expect the proposed conversion gadgets to extend naturally to arbitrary code-based masking beyond the IPM family, and that the hybrid framework can likewise be generalized to hybrid CM-BM constructions. Finally, extending the hybrid IPM-BM framework to hardware implementations, where glitch-resistant security notions are required, is a natural next step.

References

[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 Press, October 2016.

[BCC+14] Julien Bringer, Claude Carlet, Hervé Chabanne, Sylvain Guilley, and Houssem Maghrebi. Orthogonal direct sum masking: A smartcard friendly computation paradigm in a code, with builtin protection against side-channel and fault attacks. Cryptology ePrint Archive, Report 2014/665, 2014.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, Berlin, Heidelberg, August 2004.

[BFG15] Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. Inner product masking revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 486–510. Springer, Berlin, Heidelberg, April 2015.

[BFG+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 724–754. Springer, Cham, December 2017.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, Lecture Notes in Computer Science, pages 64–81. Springer, 2014.

[Can05] D. Canright. A very compact S-box for AES. In Josyula R. Rao and Berk Sunar, editors, *CHES 2005*, volume 3659 of *LNCS*, pages 441–455. Springer, Berlin, Heidelberg, August / September 2005.

[CGC+21] Wei Cheng, Sylvain Guilley, Claude Carlet, Sihem Mesnager, and Jean-Luc Danger. Optimizing inner product masking scheme by a coding theory approach. *IEEE Trans. Inf. Forensics Secur.*, 16:220–235, 2021.

[CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. Secure conversion between Boolean and arithmetic masking of any order. In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 188–205. Springer, Berlin, Heidelberg, September 2014.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Berlin, Heidelberg, August 1999.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[DDT20] Lauren De Meyer, Elke De Mulder, and Michael Tunstall. On the effect of the (micro)architecture on the development of side-channel resistant software. Cryptology ePrint Archive, Report 2020/1297, 2020.

[DZD+17] A. Adam Ding, Liwei Zhang, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. In Thomas Eisenbarth and Yannick Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano,*

Switzerland, November 13-15, 2017, Revised Selected Papers, Lecture Notes in Computer Science, pages 105–122. Springer, 2017.

[FMPR11] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 262–280. Springer, Berlin, Heidelberg, August 2011.

[GD25a] John Gaspoz and Siemen Dhooghe. Bit t-SNI secure multiplication gadget for inner product masking. *IACR TCHES*, 2025(1):104–127, 2025.

[GD25b] John Gaspoz and Siemen Dhooghe. Code-based masking: From fields to bits bitsliced higher-order masked SKINNY. *IACR TCHES*, 2025(3):212–237, 2025.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Joshua Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance. https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf, 2011. Retrieved on March 31, 2022.

[GM11] Louis Goubin and Ange Martinelli. Protecting AES with Shamir's secret sharing scheme. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 79–94. Springer, Berlin, Heidelberg, September / October 2011.

[Gou01] Louis Goubin. A sound method for switching between Boolean and arithmetic masking. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 3–15. Springer, Berlin, Heidelberg, May 2001.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 158–172. Springer, Berlin, Heidelberg, August 1999.

[GT03] Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 198–212. Springer, Berlin, Heidelberg, August 2003.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Berlin, Heidelberg, August 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Berlin, Heidelberg, August 1999.

[MN98] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[PGS+17] Romain Poussier, Qian Guo, François-Xavier Standaert, Claude Carlet, and Sylvain Guilley. Connecting and improving direct sum masking and inner product masking. In Thomas Eisenbarth and Yannick Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, Lecture Notes in Computer Science, pages 123–141. Springer, 2017.

[PR11] Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 63–78. Springer, Berlin, Heidelberg, September / October 2011.

[PV17] Kostas Papagiannopoulos and Nikita Veshchikov. Mind the gap: Towards secure 1st-order masking in software. In Sylvain Guilley, editor, *COSADE 2017*, volume 10348 of *LNCS*, pages 282–297. Springer, Cham, April 2017.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *CHES 2015*, volume 9293 of *LNCS*, pages 495–513. Springer, Berlin, Heidelberg, September 2015.

[UHIM24] Rei Ueno, Naofumi Homma, Akiko Inoue, and Kazuhiko Minematsu. SoK: Pseudorandom generation for masked cryptographic implementation. Cryptology ePrint Archive, Report 2024/1968, 2024.

[WCG+22] Qianmei Wu, Wei Cheng, Sylvain Guilley, Fan Zhang, and Wei Fu. On efficient and secure code-based masking: A pragmatic evaluation. *IACR TCHES*, 2022(3):192–222, 2022.

[WMCS20] Weijia Wang, Pierrick Méaux, Gaëtan Cassiers, and François-Xavier Standaert. Efficient and private computations with code-based masking. *IACR TCHES*, 2020(2):128–171, 2020.

[WSY+16] Weijia Wang, François-Xavier Standaert, Yu Yu, Sihang Pu, Junrong Liu, Zheng Guo, and Dawu Gu. Inner product masking for bitslice ciphers and security order amplification for linear leakages. In Kerstin Lemke-Rust and Michael Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, Lecture Notes in Computer Science, pages 174–191. Springer, 2016.

[WYS19] Weijia Wang, Yu Yu, and François-Xavier Standaert. Provable order amplification for code-based masking: How to avoid non-linear leakages due to masked operations. *IEEE Trans. Inf. Forensics Secur.*, 14(11):3069–3082, 2019.