Hindawi
IET Information Security
Volume 2024, Article ID 5544999, 16 pages
https://doi.org/10.1049/2024/5544999

Research Article

Improved Masking Multiplication with PRGs and Its Application to Arithmetic Addition

Bohan Wang,[1] **Qian Sui**,[1] **Fanjie Ji**,[1] **Chun Guo**,[1,2,3] **and Weijia Wang**[1,2,4]

[1]*School of Cyber Science and Technology, Shandong University, Qingdao 266237, Shandong, China*
[2]*Quan Cheng Laboratory, Jinan 250103, Shandong, China*
[3]*Shandong Research Institute of Industrial Technology, Jinan 250102, Shandong, China*
[4]*Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao 266237, Shandong, China*

Correspondence should be addressed to Weijia Wang; wjwang@sdu.edu.cn

Received 17 July 2023; Revised 11 November 2023; Accepted 18 November 2023; Published 18 January 2024

Academic Editor: Qichun Wang

At Eurocrypt 2020, Coron et al. proposed a masking technique allowing the use of random numbers from pseudo-random generators (PRGs) to largely reduce the use of expansive true-random generators (TRNGs). For security against d probes, they describe a construction using $2d$ PRGs, each of which is fed with at most $2d$ random variables in a finite field, resulting in a randomness requirement of $\tilde{O}(d^2)$. In this paper, we improve the technique on multiple frontiers. On the theoretical level, we push the limits of the randomness requirement by providing an improved masking multiplication using only d PRGs, each of which is fed with d random variables, saving more than half random bits. On the practical level, considering that the masking of arithmetic addition usually requires more randomness (than multiplication), we apply the technique to the algorithm proposed at FSE 2015 that is a very efficient scheme performing arithmetic addition modulo 2^w. It significantly reduces the randomness cost of masked arithmetic addition, and further advocates the advantage of masking with PRGs. Furthermore, we apply our masking scheme to the SPECK, XTEA, and SPARKLE, and provide the first (to the best of our knowledge) higher order masked implementations for the ciphers using ARX structure.

1. Introduction

Side-channel attack (SCA) [1, 2] is a kind of attack exploiting physical leakage (eg., timing information, power consumption, or electromagnetic leaks) of the cryptographic implementations. Masking is a popular countermeasure against SCA, whose concept is to randomly divide every variable (say, x) into $d+1$ shares $x_1, x_2 \ldots x_{d+1}$ such that the joint distribution of any d shares is independent of x. This is known as the d-probing (aka., d-private) security, and d is called the security order. Notably, for the popular Boolean masking, we have $x = x_1 \oplus x_2 \oplus \cdots \oplus x_{d+1}$ with $\oplus$ the addition over $\mathbb{F}_2$ (aka., bitwise XOR). Besides, it has been proved that d-probing security can ensure that the information exploited from any adversary decreases exponentially with d [3]. Mainstream masking schemes use a gate-by-gate approach that transforms each elemental operation (eg., addition and multiplication over $\mathbb{F}_{2^w}$) into its masked correspondence called gadget, surrounding which flourishing literature emerges in the last years.

One of the most groundbreaking works toward designing masking schemes is the work of Barthe et al. [4]. Instead of proving the security of full implementation at once, this work introduces the composable security notions called noninterference/strong noninterference (NI/SNI). The composable security notions allow proving the security of smaller gadgets in terms of composability with other masked circuits. Later, Cassiers and Standaert [5] proposed a new composable security notion called probing–isolating noninference (PINI), enabling a more straightforward composition of gadgets. That is, gadgets fulfilling this PINI notion can be freely composed with each other without interfering with their SCA resistance.

Coron et al. [6] proposed a special technique called locality of randomness subset, allowing the usage of multiple PRGs to reduce the randomness cost by setting proper randomness subsets of each gadget. According to it, if all gadgets are SNI-R/PINI-R defined in [6], we can securely use d-wise PRGs [7] to generate the random bit for the gadgets and keep an equivalent security in the probing model, even if the worst case where the adversary can get the variables in a PRG with one probe happens. Then, we can reuse the random seeds of d-wise PRGs in different gadgets based on the locality of the subsets, which significantly reduces the randomness cost. In [6], the ISWAND [8] has been proved as SNI-R with 1-local use of $d(d+1)/2$ subsets. Furthermore, two better SNI-R AND algorithms are given in [6] with d 1-local use subsets and d 2-local use subsets.

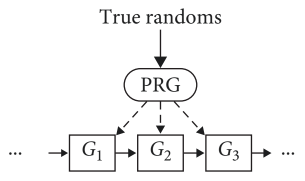
When a cryptographic algorithm involves arithmetic addition operations (eg., the add-rotate-xor (ARX)-based block ciphers such as XTEA [9] and SPECK [10], hash functions SHA-1 and SHA-2, and NIST lightweight cryptography finalist SPARKLE [11, 12]), transforming elemental operations becomes intricate—because of the higher algebra degree of the arithmetic addition operations. At CHES 2001, Goubin [13] described a very elegant algorithm for converting between shares $x_1, \ldots, x_{d+1}$ and shares $A_1, \ldots, A_{d+1}$ such that:

$$x_1 \oplus \ldots \oplus x_{d+1} = A_1 + \ldots + A_{d+1}, \quad (1)$$

with $+$ the arithmetic addition. Afterward, there has been a series of literature focusing on designing better-converting algorithms [13–19]. At FSE 2015, Coron et al. [20] described an improved algorithm performing arithmetic addition modulo 2^w with complexity $\mathcal{O}(d^2 \log w)$ that integrated conversion of both directions. Although Coron's algorithm is a very efficient scheme, (there indeed exist some other approaches only focusing on the conversion (of one direction) from Boolean to arithmetic masking with somewhat better complexities [18, 21, 22]. Despite their prospective applications in many scenarios such as masked postquantum cryptography [23, 24], it is intricate to applying them to arithmetic addition, which requires the conversions of both directions) it is not provably secure in any composed security notions and thus is risky to be used for larger composed computation. Then there is a high-order arithmetic addition algorithm proposed at [25], which is based on [20] but only satisfies NI security.

1.1. Our Contributions. Following [6], our main contribution is to propose a new security notion allowing multiple PRGs, and a more efficient masked d-order AND algorithm with 1-local use of $2d$ randomness subsets based on [5] which satisfies the new notion. Besides, we consolidate the work on masked arithmetic addition by improving the existing work of Coron et al. [20]. Our contributions can be summarized as follows.

1.1.1. A New Security Notion Allowing the Use of Multiple PRGs. We extend the composable security notion called PINI to allow more efficient (than the work in [6]) the use of multiple PRGs. This brings a new notion called PINI-extension (PINI-E). We also describe the deduction from security in PINI-E to the security in the probing model. We introduce the usage of PRGs for PINI-E gadgets at Figure 1.



	State-of-the-art work [12]	Our work
Composable security	PINI-R*	PINI-E
Ture random cost	$6d^2$	$2d^2$

*The state-of-the-art work is SNI-R security, and its implementation in [12] uses double-SNI construction to ensure the PINI-R security. We choose to compare with the double-SNI version for a similar composability.

FIGURE 1: The comparison of previous works and ours, and the usage of multiple PRGs in PINI-E gadgets.

1.1.2. A New Algorithm for Bitwise Multiplication. We propose a new d-order AND algorithm with $d(d+1)$ random bits and PINI-E security, where we apply the PINI trick proposed at [5]. Besides, we can keep its 1-local use for $2d$ randomness subsets. We show the comparison of the works proposed at [6] and ours in the locality and randomness subsets in Table 1.

1.1.3. Application to Arithmetic Addition. Based on the methodology from Coron et al. [20], we provide an algorithm for higher order masked arithmetic addition, and describe applications of our countermeasure to the SPECK, XTEA, and SPARKLE. We implement masked round functions on the ARM Cortex M3 architecture at the assembly level and report the performance results. Notably, to the best of our knowledge, they are the first implementation results of higher order masking for the ciphers using the ARX structure.

1.2. Organization. In the rest of this paper, we present notations and backgrounds in Section 2. And, we describe the new AND algorithms and give the necessary proofs in Section 3. Section 4 presents the arithmetic algorithm, including its description, related proofs and randomness cost. The implementations of the arithmetic algorithm are in Section 5. Finally, we conclude our work in Section 6.

2. Preliminaries

2.1. Notations. Let $\mathbb{F}_{2^w}$ be a field with characteristic two. Let $\oplus$ be the field addition over $\mathbb{F}_2$ (aka., bitwise XOR), and $\cdot$ be bitwise AND operation. We denote a set of variables by $(x_i)_{1\leqslant i\leqslant n} \overset{\text{def}}{=} \{x_1 \ldots x_n\}$, and particularly, if $n = d+1$, we denote the set of variables by $x_\star \overset{\text{def}}{=} \{x_1 \ldots x_{d+1}\}$. In addition, we use $x_{|I|} \overset{\text{def}}{=} \{x_i | i \in I\}$ to denote a set of variables whose indices are contained in I and denote the size of indices set I by $|I|$. Let $+$ be addition modulo 2^w. For any $a_\star \in \mathbb{F}_{2^w}^{d+1}$, let $\oplus a_\star \overset{\text{def}}{=} a_1 \oplus a_2 \oplus \cdots \oplus a_{d+1}$. Let $[d] \overset{\text{def}}{=} [1, d] \cap \mathbb{Z}$. Let $a_\star \oplus b_\star \overset{\text{def}}{=} a_i \oplus b_i$ for $i \in [d+1]$.

In a matrix A, we define s_{ij}^A as the element at the i-th row and j-th column. For $n \times n$ matrices A and B, let $A \times B \overset{\text{def}}{=} C$

TABLE 1: The comparison among PINI, PINI-R, and PINI-E.

	Security	Composability	Probe for randomness	Security requirement*
PINI	d-Private	Trivial	Wire	⋆
PINI-R	d-Private	Trivial	Subset	⋆ ⋆ ⋆
PINI-E	d-Private	Limited	Subset	⋆ ⋆

*The more stars there are, the stronger the requirement is. More precisely, a PINI-E gadget is PINI but not the other way. Furthermore, a PINI-R gadget is PINI-E where the subset number of PINI-E is n, *which is not the other way as well*. The difference of the diffculty for simulation between PINI-R and PINI-E comes from the requirement whether $c_{|\,|R}$ should be simulated or not.

where $s_{ij}^{C} \overset{\text{def}}{=} (s_{ij}^{A}, s_{ij}^{B})$ for $i, j \in [n]$. Let S_i^k be a sequence $\{s_{i1}, s_{i2}, \ldots, s_{ik}\}$ in a $n \times n$ matrix for $i, k \in [n]$. And, let $S_i^{k'}/S_i^k \overset{\text{def}}{=} \{s_{i,k+1}, s_{i,k+2}, \ldots, s_{ik'}\}$ for $k < k'$.

2.2. Private Circuits. In this part, we describe some definitions regarding the private circuit proposed in [8]. A circuit is a directed acyclic graph with gates as vertices and wires as edges, respectively, where every wire carries a variable in $\mathbb{F}_{2^w}$, and each gate represents an elementary calculation over $\mathbb{F}_{2^w}$. We recall the definition of private circuit proposed at [26] below.

Definition 1 (private circuit [26]). A private circuit for $f: \mathbb{F}_{2^w}^n \to \mathbb{F}_{2^w}^m$ is defined by a triple $(\mathsf{I}, \mathsf{C}, \mathsf{O})$, where

(1) $\mathsf{I}: \mathbb{F}_{2^w}^n \to (\mathbb{F}_{2^w}^n)^{d+1}$ is a randomized circuit called input encoder. It maps each input to $d+1$ independent shares.

(2) C is a randomized circuit with $n \times (d+1)$ inputs and $m \times (d+1)$ outputs over $\mathbb{F}_{2^w}$.

(3) $\mathsf{O}: (\mathbb{F}_{2^w}^m)^{d+1} \to \mathbb{F}_{2^w}^m$ is a circuit called decoder. It maps the outputs ($d+1$ shares) of C to the original outputs of the private circuit.

Moreover, a private circuit is called a d-private (or d-probing secure) circuit if it satisfies the requirements below:

(1) Correctness: for any input $x \in \mathbb{F}_{2^w}^n$, $\mathsf{O}(\mathsf{C}(\mathsf{I}(x))) = f(x)$;

$$\begin{matrix} 0 & r_{12} & r_{13} \\ r_{12} & 0 & r_{23} \\ r_{13} & r_{23} & 0 \end{matrix}$$

FIGURE 2: The randomness matrix of a second-order ISWAND gadget. It reflects the bold part of the above example.

(2) Privacy: for any $x, x' \in \mathbb{F}_{2^w}^n$ and any set $\mathscr{P}$ of at most d wires in C, the distributions of $\mathsf{C}_{\mathscr{P}}(\mathsf{I}(x))$ and $\mathsf{C}_{\mathscr{P}}(\mathsf{I}(x'))$ are identical, where $\mathsf{C}_{\mathscr{P}}(\mathsf{I}(x))$ refers to the values of variables in $\mathscr{P}$ with input x.

Although the definition of private circuit nicely provides protection against the SCAs, proving a large circuit (such as the AES) to be d-private is nontrivial since the possible tuples of the d wires grow exponentially with the circuit size. To cope with such an issue, Ishai et al. [8] proposed a gate-by-gate approach to transform each gate separately into the masked correspondence circuit called gadget and compose the gadgets to achieve the private circuit. A gadget is a circuit with shares as inputs and outputs.

The first d-probing secure bitwise AND gadget (that implements the bitwise AND operation over $\mathbb{F}_{2^w}$ in the masked domain) was proposed by Ishai et al. [8] at CRYPTO 2003 named ISWAND, which we give an example for $d = 2$ in the following:

$$\begin{array}{lll} c_1 \leftarrow a_1 \cdot b_1 & \oplus \mathbf{r_{12}} & \oplus \mathbf{r_{13}} \\ c_2 \leftarrow (a_1 \cdot b_2 \oplus (a_2 \cdot b_1 \oplus \mathbf{r_{12}})) & \oplus a_2 \cdot b_2 & \oplus \mathbf{r_{23}} \\ c_3 \leftarrow (a_1 \cdot b_3 \oplus (a_3 \cdot b_1 \oplus \mathbf{r_{13}})) & \oplus (a_2 \cdot b_3 \oplus (a_3 \cdot b_2 \oplus \mathbf{r_{23}})) & \oplus a_3 \cdot b_3 \end{array}. \quad (2)$$

Meanwhile, we give a randomness matrix in Figure 2 to express the construction of its randomness, which will appear in Section 3 again.

We can verify that the sum (over $\mathbb{F}_{2^w}$) of all c_is is the bitwise AND of a and b. Note that the order of the calculation is strict. For instance, at line 8, $r_{ij} \oplus a_j \cdot b_i$ is calculated before XORing $a_i \cdot b_j$.

2.3. Composable Security Notions and Extensions. Note that one has to insert many refreshing gadgets to compose d-private gadgets securely, significantly increasing the randomness cost. Barthe et al. [4] proposed the concept of composable security notions NI/SNI that enable the composition without refreshing gadgets. Below, we recall the definitions of NI/SNI proposed at [4].

Definition 2 (NI/SNI [4]). Let G be a gadget taking $a_\star, b_\star$ as inputs and returning $c_\star$. The gadget G is NI (resp., SNI) secure if and only if for any set of t intermediate variables and any subset $\mathcal{O}$ of output indices such that $t_0 = t + |\mathcal{O}| \leqslant d$, there exists sets I and J of input indices with $|I| \leqslant t_0$ and $|J| \leqslant t_0$ (resp., $|I| \leqslant t$ and $|J| \leqslant t$), such that the t intermediate variables and the output variables $c_{|\mathcal{O}}$ can be perfectly simulated from $a_{|I}$ and $b_{|J}$.

Besides, there is an updated definition called SNI-R proposed at [6], which is used in the situation where a randomness subset in gadget G can be got with a single probe.

Definition 3 (SNI-R [6]). Let G be a gadget with input shares $a_\star$ and $b_\star$, output shares $c_\star$. Let $(\rho_i)_{1\leqslant i\leqslant d+1}$ be subsets of the randoms used by G. The gadget is SNI-R if and only if for any set of t intermediate variables, any subset $\mathcal{O}$ of output indices and any subset $R\subset[n]$, such that $t+|\mathcal{O}|+|R|\leqslant d$. Then the t intermediate variables, the output variables $c_{|\mathcal{O}\cup R}$, and all ρ_i for $i\in R$ can be perfectly simulated from the knowledge of $a_{|I\cup R}$ and $b_{|J\cup R}$ with $|I|\leqslant t$ and $|J|\leqslant t$.

However, the security of the trivial composition of several private circuits is not evident. More precisely, even the SNI circuits can not keep its security with trivial composition. To mitigate this issue, Cassiers and Standaert [5] proposed a new composable security notion called PINI, by which we can concentrate on the proof of every single gadget and the global security can be directly deduced. We recall it in the following.

Definition 4. (PINI [5], adapted (the original PINI security is defined for arbitrary number of inputs, we provide a fan-in 2 version in our paper)). Let G be a gadget with input shares $a_\star, b_\star$ and output shares $c_\star$. The gadget G is PINI if for any $t_1\in\mathbb{N}$, any set of t_1 intermediate variables and any subset $\mathcal{O}$ of output indices, there exists a subset $I\subset[1,d+1]$ of input indices with $|I|\leqslant t_1$ such that the t_1 intermediate variables and the output shares $c_{|\mathcal{O}}$ can be perfectly simulated from the input shares $a_{|I\cup\mathcal{O}}$ and $b_{|I\cup\mathcal{O}}$.

Meanwhile, Cassiers and Standaert [5] provided a gadget construction called double-SNI which can turn SNI gadgets into PINI one.

Definition 5 (double-SNI [5]). Let G be an SNI gadget taking as input $a_\star, b_\star$ and output $c_\star$. Let R be an SNI gadget taking as input $x_\star$ and output $y_\star$. The composite gadget G' taking as input $x_\star, b_\star$, and output $c_\star$ with $G'(x_\star, b_\star)=G(R(x_\star), b_\star)$ is PINI.

To reduce the randomness cost of a large circuit, there is an adapted definition called PINI-R proposed at [6] which also assumes the adversary can get a randomness subset with a single probe.

Definition 6 (PINI-R [6]). Let G be a gadget with input shares $a_\star, b_\star$ and output shares $c_\star$. Let $(\rho_i)_{1\leqslant i\leqslant d+1}$ be subsets of the randoms used by G. The gadget G is PINI-R if for any $t_1\in\mathbb{N}$, any set of t_1 intermediate variables, any subset $\mathcal{O}$ of output indices and any subset $R\subset[n]$, there exists a subset $I\subset[d+1]$ of input indices with $|I|\leqslant t_1$ such that the t_1 intermediate variables, the output shares $c_{|\mathcal{O}\cup R}$, and the randoms ρ_i for $i\in R$ can be perfectly simulated from the input shares $a_{|I\cup\mathcal{O}\cup R}$ and $b_{|I\cup\mathcal{O}\cup R}$.

2.4. Masking with Randomness from PRGs. In this part, we recall some definitions for gadgets using randomness generated from PRGs and the corresponding PRGs.

2.4.1. Locality of Randomness and Its Application. First of all, we introduce the locality of randomness subset proposed at [6] used to describe the reuse extent of the randoms. It decides the PRGs used for the subset.

Definition 7 (ℓ-local randomness subset [6], adapted). Let G be a gadget and ρ be a randomness subset used by G. We say that ρ is ℓ-local use if any intermediate variable of G is related with at most ℓ elements of ρ.

With the definition of locality, we propose a weaker security definition than PINI-R which can also keep the composability and d-private with the same extended probing model as PINI-R. We define it as PINI-E (shorted for PINI-Extension) and provide the definition in the following.

Definition 8 $(\mathrm{PINI}-\mathrm{E})$. Let G be a gadget with input shares $a_\star, b_\star$ and output shares $c_\star$. Let $(\rho_i)_{1\leqslant i\leqslant m}$ be subsets of the randoms used by G with $m\in\mathbb{N}$, and each ρ_i is ℓ_i local use. The gadget G is PINI-E if for any set of t intermediate variables, any subset $\mathcal{O}$ of output indices and any set of t_r randomness subsets ρ_i with $t+t_r+|\mathcal{O}|\leqslant d$, there exist subsets I, $R\subset[d+1]$ with $|I|\leqslant t$ and $|R|\leqslant t_r$ such that the t intermediate variables, the t_r subsets ρ_i and the output shares $c_{\mathcal{O}}$ can be perfectly simulated from the input shares $a_{|I\cup\mathcal{O}\cup R}$ and $b_{|I\cup\mathcal{O}\cup R}$.

Obviously, PINI-E is an extension of PINI which allows to probe a subset of randomness with a single probe. And compared with PINI-R, the PINI-E security does not need to simulate $c_{|R}$, therefore PINI-E algorithm is easier to construct. But intuitively, its number of randomness subsets is not bounded as PINI-R. We introduce the d-private security and composability of PINI-E in the following and provide the proofs at Appendix A as supproting information.

Theorem 1 (security of $\mathrm{PINI}-\mathrm{E}$). *Let G be a gadget with input shares $a_\star, b_\star$ and output shares $c_\star$. Let $(\rho_i)_{i\in[m]}$ be a partition of the randomness used by G. If G is* PINI-E *with randoms $(\rho_i)_{i\in[m]}$, then G is d-private secure in an extended model of security where the adversary can get each ρ_i with a single probe.*

Theorem 2 (composability of $\mathrm{PINI}-\mathrm{E}$). *Let $(G_i)_{1\leqslant i\leqslant k}$ be* PINI-E *implementations of f with randomness subsets $(\rho_j)_{1\leqslant j\leqslant m}$. The composite gadget made of G_i is* PINI-E *with randomness subsets $(\rho_j)_{1\leqslant j\leqslant m}$.*

We stress that the G_i in Theorem 2 are the implementations of the same f. Meanwhile, we provide a proposition about the composition of PINI-E gadgets implementing different algorithms $(f_i)_{1\leqslant i\leqslant k}$ which we also prove at Appendix A as supproting information.

```
Input: shares $a_\star \in \mathbb{F}_{2^w}^{d+1}$
Output: shares $b_\star \in \mathbb{F}_{2^w}^{d+1}$
It ensures $\oplus b_\star = \oplus a_\star$
1: $b_{d+1} = a_{d+1}$
2: for $i = 1$ to $d$ do
3:     $s_i \leftarrow \{0, 1\}^w$
4:     $b_i \leftarrow s_i$
5:     $b_{d+1} \leftarrow b_{d+1} \oplus (a_i \oplus s_i)$
6: end for
```

Algorithm 1: LR: locality refreshing [7].

Proposition 1. *Let $(G_i)_{1\leqslant i\leqslant k}$ be* PINI-E *implementations of $(f_i)_{i\in[k]}$ with randomness $(\rho_k^i)_{k\in[n_i]}$. The composite gadget made of G_i is* PINI-E *with the same randomness subsets.*

Proposition 1 shows why PINI-E is weaker than PINI-R since the composition of PINI-R gadgets keeps the number of randomness subsets regardless of the circuit size. We mention that the composability of PINI-E is theoretically limited for the situation where there is more than one kind of gadget used to replace the same gates in the unprotected circuit, and all these gadgets use the same PRG (e.g., two kinds of multiplication gadgets are used in one circuit, and both of them use the same PRG), which barely happens in reality. In Table 1, we compare the security of PINI, PINI-R, and PINI-E. The remaining part is the construction of the masked implementation with locality property. We recall the mask refreshing named locality refreshing (LR) from the study of Ishai et al. [7] to keep a small locality for each gadget in Algorithm 1.

Lemma 1. *The* LR *gadget is* PINI-E *with $\rho_i \overset{\text{def}}{=} \{s_i : i \in [d]\}$.*

The proof of Lemma 1 is equivalent to prove PINI which has been proposed at [6], because the division of ρ_i in Lemma 1 is exactly the single random s_i.

Theorem 3 (locality composition with randomness subset [6], adapted). *Let $(G_i)_{1\leqslant i\leqslant k}$ be a set of 2-input gadgets with randomness subsets $(\rho_j^i)_{1\leqslant j\leqslant m}$, each of which makes an ℓ_j-local use. Consider the gadgets G_i' where the inputs and output of each G_i' is locality refreshed with randoms $s_t^{(a,i)}$, $s_t^{(b,i)}$ and $s_t^{(c,i)}$ for $1 \leqslant t \leqslant d$. Any composite gadget made of G_i' makes an ℓ_j-local use of randomness $\bigcup_{i\in[k]}\rho_j^i$, and for all $1 \leqslant t \leqslant d$, it makes a 1-local use of the randoms in $\{s_t^{(a,i)}, s_t^{(b,i)}, s_t^{(c,i)} : i \in [k]\}$.*

2.4.2. Application of Multiple PRGs. We recall the definition of r-wise independent PRG, which can be much more efficient than traditional PRGs.

Definition 9. (r-wise independent PRG [7] (we adapt the elements in each subset to those in $\mathbb{F}_{2^w}$, while the original definition was in $\mathbb{F}_2$ in [7])). A function G: $\mathbb{F}_{2^w}^n \rightarrow \mathbb{F}_{2^w}^m$ is an r-wise independent PRG if any subset of its r outputs is independently and uniformly distributed when the input is uniformly distributed.

Here, we describe two r-wise PRGs called $\mathscr{R}_1$ and $\mathscr{R}_2$ proposed at [6]. The parameter r of $\mathscr{R}_1$ can be set as any positive integer while that of $\mathscr{R}_2$ is fixed as three. However, the running efficiency of $\mathscr{R}_2$ is much higher than that of $\mathscr{R}_1$.

We define $\mathscr{R}_1 : \mathbb{F}_{2^w}^r \rightarrow \mathbb{F}_{2^w}^{2^w}$ as follows:

$$\mathscr{R}_1(\mathbf{a}) = (h_\mathbf{a}(0), \ldots, h_\mathbf{a}(2^w - 1)) \ , \tag{3}$$

where $\mathbf{a} = (a_0, \ldots, a_{r-1}) \in \mathbb{F}_{2^w}^r$ and:

$$h_\mathbf{a}(x) = \sum_{i=0}^{r-1} a_i x^i \ . \tag{4}$$

$\mathscr{R}_1$ is an r-wise PRG because there is a bijection between the r coefficients of $h_\mathbf{a}(x)$ and its evaluation at r distinct points x_i [6]. For instance, $\mathscr{R}_1$ can output at most $w \cdot 2^w$ bits of randomness when given wr bit seeds over $\mathbb{F}_{2^w}$.

We define another PRG $\mathscr{R}_2 : \mathbb{F}_2^{2n} \rightarrow \mathbb{F}_2^{n^2}$ as follows:

$$\mathscr{R}_2(x_1, \ldots, x_n, y_1, \ldots, y_n) = (x_i \oplus y_i)_{1\leqslant i,j\leqslant n} \ . \tag{5}$$

This PRG is based on the expander graph used in [7]. It can generate n^2 randoms by $2n$ bit seeds. It is much more lightweight (with only XOR operations) than $\mathscr{R}_1$. In [6], it is proved as a 3-wise PRG, recalled Lemma 2.

Lemma 2 (see [6]). *The randomized function $\mathscr{R}_2$ is a 3-wise independent PRG.*

Then, we introduce the security of masking with multiple PRGs in Theorem 4 proposed at [6], where we can keep ℓ-local gadgets secure when multiple PRGs are used to generate the random elements. This reduces the randomness cost efficiently.

Theorem 4 (security with multiple PRGs [6], adapted). *Suppose C is a d-private implementation of f with encoder I and decoder O, where the circuit $\mathsf{C}(\widehat{\omega}, \rho_1, \ldots \rho_k)$ uses for each $1 \leqslant i \leqslant k$, n random elements ρ_i and makes an ℓ-local use of ρ_i, $\widehat{\omega}$ are the inputs of C, and the adversary can obtain ρ_i with a single probe. Let $G : \mathbb{F}_{2^w}^{n_r} \rightarrow \mathbb{F}_{2^w}^n$ be a linear ℓd-wise independent PRG. Then, the circuit C' denoted by $\mathsf{C}'(\widehat{\omega}, \rho_1', \ldots \rho_k') = \mathsf{C}(\widehat{\omega}, G(\rho_1'), \ldots G(\rho_k'))$ is a d-private implementation of f with encoder I and decoder O, which uses $k \cdot n_r$ random elements.*

2.5. Coron's Work on Masked Arithmetic Addition. Coron et al. [20] introduce a new algorithm to convert from

```
Input: shares $a_\star \in \mathbb{F}_{2^w}^{d+1}$ and $b_\star \in \mathbb{F}_{2^w}^{d+1}$
Output: shares $c_\star \in \mathbb{F}_{2^w}^{d+1}$
It ensures $\oplus c_\star = \oplus a_\star \cdot \oplus b_\star$
1: $c_{d+1} \leftarrow a_{d+1} \cdot b_{d+1}$
2: for $i = 1$ to $d$ do
3:     $c_i \leftarrow a_i \cdot b_i$
4:     for $j = 1$ to $d+1$ do
5:         if $j < d+2-i$ then
6:             $t_{i,d+2-j} \leftarrow \{0,1\}^w$
7:             $c_i \leftarrow c_i \oplus t_{i,d+2-j}$
8:         end if
9:         if $i < j$ then
10:            $r_{ij} \leftarrow \{0,1\}^w$
11:            $c_i \leftarrow \mathrm{PIRT}(a_i, b_j, r_{ij}, c_i)$
12:        end if
13:    end for
14: end for
15: for $i = 2$ to $d$ do
16:    for $j = 1$ to $d-1$ do
17:        if $j > d+2-i$ then
18:            $c_i \leftarrow c_i \oplus t_{j,d+1-i}$
19:        end if
20:        if $i > j$ then
21:            $c_i \leftarrow \mathrm{PIRT}(a_{i+1}, b_j, r_{j,i+1}, c_i)$
22:        end if
23:    end for
24: end for
25: for $j = 1$ to $d$ do
26:    $c_{d+1} \leftarrow c_{d+1} \oplus t_{j,d+1-j}$
27:    $c_{d+1} \leftarrow \mathrm{PIRT}(a_{j+1}, b_j, r_{j,j+1}, c_{d+1})$
28: end for
```

ALGORITHM 2: LatinAND: the new PINI-E AND algorithm.

arithmetic masking to Boolean masking, which is introduced in Theorem 5. This algorithm uses Kogge-Stone [27] carry look-ahead algorithm proposed at to replace the classical ripple-carry adder, which reduces the complexity from $\mathcal{O}(w)$ (in a previous work [15]) to $\mathcal{O}(\log w)$.

Theorem 5 (see [20]). *Let $x, y \in \mathbb{F}_{2^w}$, $\ell = \lceil \log_2(w-1) \rceil$. Define the sequence of w-bit variables $P^{(i)}$ and $Q^{(i)}$, with $P^{(0)} = x \oplus y$ and $Q^{(0)} = x \cdot y$, and:*

$$\begin{cases} P^{(k)} = P^{(k-1)} \cdot \left(P^{(k-1)} \ll 2^{k-1}\right) \\ Q^{(k)} = \left(P^{(k-1)} \cdot \left(Q^{(k-1)} \ll 2^{k-1}\right)\right) \oplus Q^{(k-1)} \end{cases}, \qquad (6)$$

for $1 \leqslant k \leqslant \ell$. Then $x + y = x \oplus y \oplus (2Q^{(\ell)})$.

In Section 4, we propose a new algorithm that expands the security order from 1 to any d based on Theorem 5.

```
Input: input shares $a_i, b_j$, random $r_{i'j'}$ and an intermediate variable $c$
Output: the intermediate variable $c'$
It ensures $c' = c \oplus a_i b_j \oplus r$
1: $s_{ij} \leftarrow b_j \oplus r_{i'j'}$
2: $p_{ij}^0 \leftarrow (a_i \oplus 1^w) \cdot r_{i'j'}$
3: $p_{ij}^1 \leftarrow a_i \cdot s_{ij}$
4: $c' \leftarrow c \oplus p_{ij}^0 \oplus p_{ij}^1$
```

ALGORITHM 3: PIRT: part of a PINI algorithm [5], adapted.

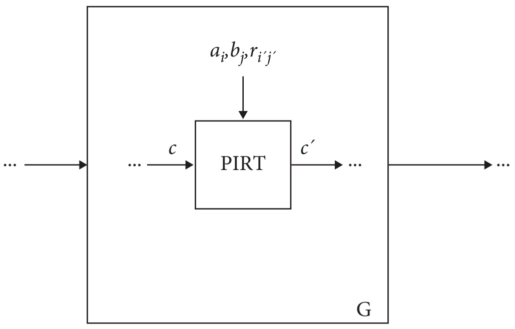


FIGURE 3: The example of how PIRT works in a gadget G. The indexes i', j' are used to mention they are independent with i, j.

Besides, we have proved its PINI-E security and locality in Appendix B as supproting information. We give a new AND algorithm in Section 3 with 1-local use of its $\mathcal{O}(d)$ randomness subsets with odd d and use it in the new arithmetic algorithm.

3. The New Masked AND Gadget

In this section, we introduce a new masked AND algorithm with lower locality, as well as its PINI-E security and corresponding proof.

3.1. The Description of the New Algorithm. We describe our new algorithm with odd d in Algorithm 2, which is provable secure in PINI-E with 1-local use of $2d$ randomness subsets. We will prove its PINI-E security and locality in the next section. Algorithm 3 provides a PINI trick proposed at [5] keeping LatinAND PINI (and PINI-E). Note that the inputs and ouput of PIRT (short for PINI-PART) are explained at Figure 3.

Intuitionally, PINI security does not allow the leakage of more than one input indices with one probe, and the PINI trick (i.e., Algorithm 3) avoids these leakages in the multiplication gadgets by changing the operation order of multiplying secret (i.e., a_i and b_j in Algorithm 3) and adding randoms (i.e., $r_{i'j'}$ in Algorithm 3). In comparison, ISWAND calculates $a_i \cdot b_j + r_{ij}$ directly and thus it is not PINI.

The intermediate step intuitionally defines a partial order among the intermediate variables. We use this definition in

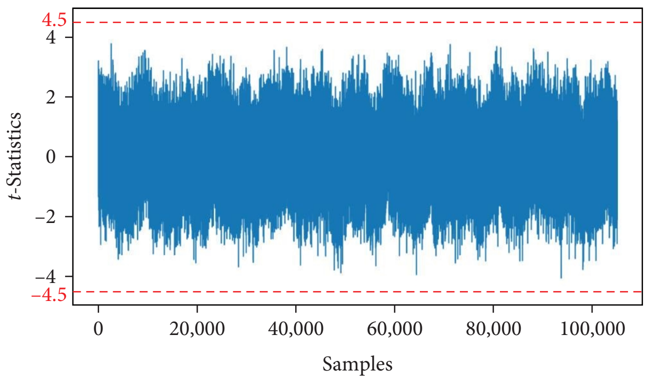


FIGURE 4: LatinAND, PRGs.

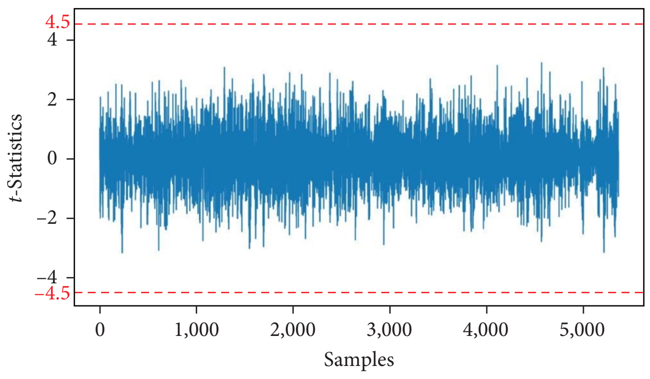


FIGURE 5: AND gadget proposed in [28], TRNGs.

the PINI-E proof of LatinAND which is given at Appendix B as supproting information.

Definition 10. (Intermediate step). Let a and c be the intermediate variables of gadget G. We define a as the intermediate step of c if some b exists for $a \oplus b = c$ or $a \cdot b = c$.

Theorem 6. LatinAND *is* PINI-E *with randomness subsets* ρ_k^L *and* ρ_k^R *for* $k \in [d]$.

3.2. The Randomness Reuse of LatinAND. We mention that LatinAND is 1-local use of $2d$ randomness subsets in Theorem 7. And we can build a gadget G with LatinAND which always keeps its 1-local use of randomness subsets by Theorem 3. And, if G satisfies the d-probing security in Theorem 4, we can use $2d$ d-wise PRGs to generate all r_{ij} and t_{ij} in LatinAND.

Theorem 7. LatinAND *is 1-local use of* ρ_k^L *and* ρ_k^R *for* $k \in [d]$.

The proof of Theorem 7 is obvious, because all randoms in each ρ_k^L or ρ_k^R appear only once in each c_i. This is exactly the definition of the locality of randomness subset.

3.3. Discussion for the Randomness of LatinAND. We have proven in the prevoius subsections that PINI-E gadget is d-probing secure with PRG-generated randoms and almost trivial composability, and the PRGs are required to be ℓd-wise if the randoms are ℓ-local use. Also, we provide the construction of r-wise PRGs with arbitrary r. Moreover, we have proven that LatinAND is PINI-E. Thus, the randomness of LatinAND is theoretically indistinguishable from a d-probing secure AND gadget with TRNG-generated randoms if the PRGs of LatinAND are d-wise.

To validate the impact of the randomness on the practical security, we run LatinAND and another multiplication gadget proposed in [28] on a ChipWhisperer STM32F4 UFO target board and collect its power traces with Picoscope 5244D at sampling rate of 125 MS/s. Besides, we perform a Welch's T-test with $10,000$ executions, whose randoms are generated by PRGs (LatinAND) and TRNGs (AND gadget proposed in [28]), respectively, to compare the randomness of the PRG implementation and the TRNG ones. Figure 4 depicts the T-test results for LatinAND, and we provide in Figure 5, the result for the other gadget with the randomness from TRNGs.

4. Application to Arithmetic Addition

In this section, we implement LatinAND gadget in an arithmetic addition algorithm proposed at [25], which is costly in randomness for previous multiple gadgets.

Our description is structured by means of top-down. All gadgets presented in this subsection are PINI-E, and we defer the security proofs to Appendix B as supproting information. First of all, we describe the algorithm SecADD to perform addition operations directly on the masked shares, which is similar to the algorithm proposed in [25] but we add some construction in our algorithm so that it can use multiple PRGs. More precisely, we receive the shares $a_\star$ and $b_\star$ satisfying $a = \oplus a_\star$ and $b = \oplus b_\star$ as inputs, and the goal is to compute $c_\star$ satisfying $\oplus c_\star = a + b$. Note that our new algorithm is based on the concept of [20] and adapted for higher security orders. We describe it in Algorithm 4.

In the rest of this subsection, we will explain the construction of the ingredients GoQ$_i$ and GoP$_i$. Both of them are additionally with locality property for the use of r-wise PRGs, so that the randomness cost can be reduced.

First, we propose the gadgets to calculate $P^{(i)}$ and $Q^{(i)}$ in Theorem 5. We will introduce GoP$_k$ gadget first, which is used to generate P_k proposed at Equation (6) because the inputs of GoP$_k$ are the outputs of GoP$_{k-1}$, furthermore, they do not need any intermediate variables from the generation of Q_k for $0 \leqslant k \leqslant \ell - 1$ with $\ell \overset{\text{def}}{=} \lceil \log_2 (w-1) \rceil$. Algorithm 5 is the description of GoP$_k$.

Then it comes to $Q^{(k)}$, which is shown in Algorithm 6. We use GoQ$_k$ gadget to get all $Q^{(k)}$ proposed at Equation (6). But the inputs of GoQ$_k$ are the outputs of GoP$_{k-1}$ and GoQ$_{k-1}$, so we must get $P^{(k-1)}$ and $Q^{(k-1)}$ first to calculate $Q^{(k)}$, and this is why we introduce it as the latter one.

```
Input: shares $a_\star \in \mathbb{F}_{2^w}^{d+1}$ and $b_\star \in \mathbb{F}_{2^w}^{d+1}$
Output: shares $c_\star \in \mathbb{F}_{2^w}^{d+1}$
It ensures $\oplus c_\star = \oplus a_\star + \oplus b_\star$
1: $Q_\star \leftarrow \mathrm{LR}(\mathrm{LatinAND}(\mathrm{LR}(a\star); \mathrm{LR}(b_\star)))$
2: $P_\star \leftarrow \mathrm{LR}(\mathrm{LR}(a_\star) \oplus \mathrm{LR}(b_\star))$
3: $\ell = \lceil \log_2 (w-1) \rceil$
4: for $k = 1$ to $\ell - 1$ do
5:     $Q_\star \leftarrow \mathrm{GoQ}_k(P_\star; Q_\star)$     ▷ $Q^{(k)} \leftarrow (P^{(k-1)} \cdot (Q^{(k-1)} \ll 2^{k-1})) \oplus Q^{(k-1)}$
6:     $P_\star \leftarrow \mathrm{GoP}_k(P_\star)$     ▷ $P^{(k)} \leftarrow P^{(k-1)} \cdot (P^{(k-1)} \ll 2^{k-1})$
7: end for
8: $Q_\star \leftarrow \mathrm{GoQ}_\ell(P_\star; Q_\star)$
9: for $i = 1$ to $d + 1$ do
10:     $Q_i \leftarrow Q_i \ll 1$
11: end for
12: $x_\star \leftarrow \mathrm{LR}(\mathrm{LR}(a_\star) \oplus \mathrm{LR}(b_\star))$
13: $c_\star \leftarrow \mathrm{LR}(\mathrm{LR}(Q_\star) \oplus \mathrm{LR}(x_\star))$
```

ALGORITHM 4: SecADD: masked addition.

```
Input: shares $a_\star \in \mathbb{F}_{2^w}^{d+1}$
Output: shares $c_\star \in \mathbb{F}_{2^w}^{d+1}$
1: for $i = 1$ to $d + 1$ do
2:     $b_i \leftarrow a_i \ll 2^{k-1}$
3: end for
4: $c_\star \leftarrow \mathrm{LR}(\mathrm{LatinAND}(\mathrm{LR}(a\star); \mathrm{LR}(b\star)))$
```

ALGORITHM 5: GoP$_k$: generation of $P^{(k)} = P^{(k-1)} \cdot (P^{(k-1)} \ll 2^{k-1})$.

```
Input: shares $a_\star \in \mathbb{F}_{2^w}^{d+1}$ and $b_\star \in \mathbb{F}_{2^w}^{d+1}$
Output: shares $c_\star \in \mathbb{F}_{2^w}^{d+1}$
1: for $i = 1$ to $d + 1$ do
2:     $x_i \leftarrow b_i \ll 2^{k-1}$
3: end for
4: $y_\star \leftarrow \mathrm{LR}(\mathrm{LatinAND}(\mathrm{LR}(x_\star); \mathrm{LR}(a_\star)))$
5: $c_\star \leftarrow \mathrm{LR}(\mathrm{LR}(y_\star) \oplus \mathrm{LR}(b_\star))$
```

ALGORITHM 6: GoQ$_k$: generation of $Q^{(k)} = (P^{(k-1)} \cdot (Q^{(k-1)} \ll 2^{k-1})) \oplus Q^{(k-1)}$.

Meanwhile, we provide the evaluation of the randomness cost for Algorithm 4 in Appendix C.

5. Masked Implementations of SPARKLE, XTEA, and SPECK

In this section, to evaluate the performance of SecADD, we apply our scheme to SPARKLE, XTEA, and SPECK ciphers. SPARKLE [11, 12] is a family of cryptographic permutations shortlisted for the finalists NIST lightweight cryptography standardization. We choose SPARKLE256 for the evaluation. XTEA block cipher was introduced in [9], which is designed to correct weaknesses in TEA. And SPECK is a family of lightweight block ciphers publicly released by the National Security Agency (NSA) [10], which is optimized for performance in software implementations.

SPARKLE, XTEA, and SPECK are all based on the ARX design with arithmetic addition, rotation, and XOR operations, where the masked arithmetic addition perfectly fits SecADD. We can use SecXOR for the masking of bitwise XOR operations, which is PINI-E. For masking of shifting operations, we directly use the trivial implementation where each share is operated separately. For example, the masked rotate left shifting by n can be implemented by $c_i = a_i \lll n, i \in [1, d+1]$ with a_i the input share and c_i the output one, which is secure in PINI-E. We use independent random bits/seeds for different SecADD. SecADD is PINI-E according to the composability of PINI-E. By Proposition 1, the masked SPARKLE, XTEA, and SPECK are all PINI-E.

We implement masked SPARKLE XTEA and SPECK based on ARM Cortex M3 architecture at assembly level, for illustrative purposes and timing comparisons. We show the costs in

TABLE 2: Running kilocycles/random bits of masked SPARKLE with different security orders.

Security order	SecADD with $\mathscr{R}_1$	SecADD with $\mathscr{R}_2$
$d = 1$	6,67/2,560	791/22,976
$d = 2$	2,654/11,264	933/44,544
$d = 3$	6,626/23,040	1,115/69,120
$d = 5$	21,814/64,000	—

Table 3: Running kilocycles/random bits of masked XTEA with different security orders.

Security order	SecADD with $\mathscr{R}_1$	SecADD with $\mathscr{R}_2$
$d = 1$	1,015/1,120	1,203/10,052
$d = 2$	4,041/4,928	1,418/19,488
$d = 3$	10,092/10,080	1,694/30,240
$d = 5$	33,232/28,000	—

Table 4: Running kilocycles/random bits of masked SPECK with different security orders.

Security order	SecADD with $\mathscr{R}_1$	SecADD with $\mathscr{R}_2$
$d = 1$	8/160	10/1,436
$d = 2$	32/704	11/2,784
$d = 3$	79/1,440	13/4,320
$d = 5$	260/4,000	—

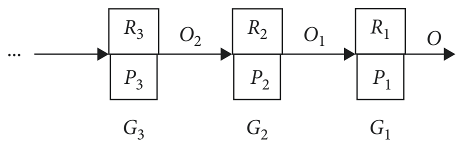


Figure 6: The composition of PINI-E gadgets. Let P_i be the set of probed intermediate variables, and let R_i be the randomness subsets. $\mathcal{O}_i$ are the probed outputs of G_{i+1}. Specially, $\mathcal{O}$ is the probed output set of the composite gadget.

Tables 2–4, with the number of required true random number bits. We present the implementations using SecADD with both $\mathscr{R}_1$ and $\mathscr{R}_2$ introduced in Section 2.

6. Conclusion

We proposed a new security definition named PINI-E to release the requirements in PINI-R proposed in [6], where both of them support the randoms generated by multiple PRGs. Furthermore, we provide a high-order PINI-E multiplication gadget (i.e., Algorithm 2) with a two-thirds reduction of true random cost compared with the state-of-the-art work proposed in [6]. Then we apply the new multiplication gadget into the Boolean-to-Boolean arithmetic addition algorithm (i.e., Algorithm 4), and use it in the implementations of SPARKLE, XTEA, and SPECK based on ARM Cortex M3, which are the first implementations of higher order masking for the ciphers using the ARX structure.

Appendix

A. Composability and Security of PINI-E

A.1. Proof of Composability

Proof. Consider the composite gadget like Figure 6, we define P_i as the probed intermediate variables of G_i, where $|P_i| = t_i$. And we denote by R_i the indice sets of probed randomness subsets ρ_j for each G_i. Furthermore, the indice set of probed output are defined as $\mathcal{O}_i$ for G_{i+1}. For G_1 which is the last gadget of the composite gadget, its probed output set is $\mathcal{O}$. Meanwhile, we have:

$$\sum|P_i| + \sum|R_i| + \sum|\mathcal{O}_i| + |\mathcal{O}| \leqslant d\,. \tag{A.1}$$

First we consider G_1. According to PINI-E, the indice set of its inputs $I_1 \cup \mathcal{O} \cup R_1$ can simulate all probes in G_1, where $|I_1| \leqslant |P_1|$. Then we consider G_2. Since the outputs of G_2 are the inputs of G_1, the indice set for the simulation of G_1 is equivalent to the probed output of G_2. Therefore the probed output indice set of G_2 becomes $\mathcal{O}_1 \cup I_1 \cup \mathcal{O} \cup R_1$. Meanwhile, according to Theorem 4, the indice set of randoms for G_2 should be $R_1 \cup R_2$. So, the indice set of input for G_2 to simulate all the probes is as follows:

$$\begin{aligned} I_2 \cup \mathcal{O}' \cup R' &= I_2 \cup (\mathcal{O}_1 \cup I_1 \cup \mathcal{O} \cup R_1) \cup (R_1 \cup R_2) \\ &= (I_1 \cup I_2) \cup (\mathcal{O} \cup \mathcal{O}_1) \cup (R_1 \cup R_2)\,. \end{aligned} \tag{A.2}$$

With this proof method, the indice set of input for the first G_i of the composite gadget is $\bigcup I_i \cup \bigcup \mathcal{O}_i \cup \mathcal{O} \cup \bigcup R_i$. And we have:

$$\begin{aligned} &\left| \bigcup_{i\in[k]} I_i \cup \bigcup_{i\in[k-1]} \mathcal{O}_i \cup \mathcal{O} \cup \bigcup_{i\in[k]} R_i \right| \\ &\leqslant \left| \bigcup_{i\in[k]} I_i \right| + \left| \bigcup_{i\in[k-1]} \mathcal{O}_i \right| + |\mathcal{O}| + \left| \bigcup_{i\in[k]} R_i \right| \,, \\ &\leqslant \sum_{i\in[k]} |P_i| + \sum_{i\in[k]} |R_i| + \sum_{i\in[k-1]} |\mathcal{O}_i| + |\mathcal{O}| \\ &\leqslant d \end{aligned} \tag{A.3}$$

which means the composite gadget is also PINI-E with $(\rho_j)_{j\in[m]}$. □

A.2. Proof of Security

Proof. WLOG, let $t + |R| + |\mathcal{O}| \leqslant d$. The t intermediate variables and $|R|$ probed randomness subsets can be simulated by input shares with indices $I \cup R$, and the probed outputs can be simulated by the inputs with indices $\mathcal{O}$. Consider $|I| \leqslant t$, we have

$$|I \cup \mathcal{O} \cup R| \leqslant |I| + |\mathcal{O}| + |R| \leqslant t + |R| + |\mathcal{O}| \leqslant d\,. \tag{A.4}$$

Therefore the adversary learns nothing from the inputs. □

A.3. Proof of Proposition 1

Proof. We suppose the input indice set for G_i is I_i and the probed randomness subset is R_i. Consider the last gadget G_1 where its ouput $\mathcal{O}$ is the output of the whole composition, we

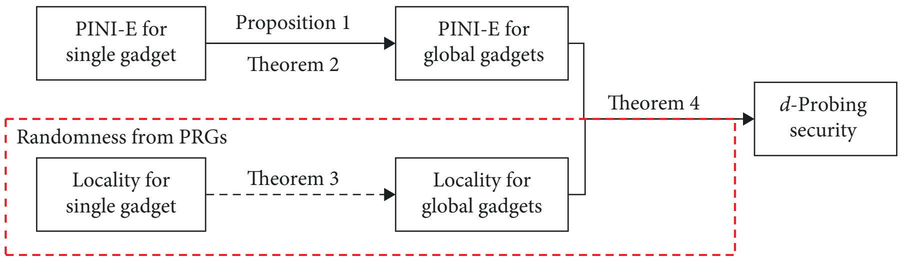


Figure 7: One can focus on PINI-E security and locality property of each single gadget, then the whole algorithm's probing security can be deducted by the proof sketch.

use $I_1 \cup R_1 \cup \mathcal{O}$ to simulate all its probed variables. For G_2 where its output $\mathcal{O}_1$ is one of the inputs of G_1, its indice set for simulation should be:

$$I_2 \cup R_2 \cup (\mathcal{O}_1 \cup I_1 \cup R_1 \cup \mathcal{O}) = I_2 \cup (R_2 \cup \mathcal{O}_1 \cup I_1 \cup R_1) \cup \mathcal{O}. \tag{A.5}$$

If G_1 and G_2 is the implementation of the same f_i, they are PINI-E according to Theorem 2. And if they are different implementations, we have

$$\begin{aligned} &|I_2 \cup R_2 \cup (\mathcal{O}_1 \cup I_1 \cup R_1 \cup \mathcal{O})| \\ &\leqslant |I_2 \cup R_2 \cup \mathcal{O}_1| + |I_1 \cup R_1 \cup \mathcal{O}| \\ &\leqslant (t_2 + |R_2|) + (t_1 + |R_1|) \\ &\leqslant d \end{aligned}, \tag{A.6}$$

where t_2 (resp., t_1) is the number of probes in G_2 (resp., t_1) without its probed output. Therefore, the simulation for the whole composition needs no more than $\sum_{i\in[k]}(t_i + |R_i|) \leqslant d$ input indices, and the indice set is $I_k \cup (\bigcup_{i\in[k-1]}(R_i \cup I_i \cup \mathcal{O}_i) \cup R_k) \cup \mathcal{O}$ which satisfies PINI-E. As a result, the composition is PINI-E. □

A.4. Application of PINI-E. Consider the properties of PINI-E, if $(G_i)_{i\in[n]}$ are proved as PINI-E, their composition will satisfy Theorem 4. Moreover, all ℓ-local randomness subsets ρ_j in Theorem 2 can be generated by a ℓd-wise PRG.

Moreover, we provide the whole procedure of how to use multiple PRGs in PINI-E gadgets and keep them d-private in the following.

How to use multiple PRGs in PINI-E gadgets:

(1) We assume gadget G is the composition of gadgets $(G_{ij})_{i\in[n]}$ where G is the implementation of f and G_{ij} are those of f_i, and each G_{ij} is PINI-E with randomness subsets $(\rho_k^i)_{k\in[q_i]}$, each of which is ℓ_k^i local use. The subscript k is used to distinguish the different randomness subsets in G_{ij}. We mention that the subscript j is used to count how many times the f_i is implemented in f.

(2) According to Theorem 3, we add three LR gadgets to the two inputs and one output of each G_{ij}, each of which owns the randomness subsets $\rho_{s,i}^{(k)} \overset{\text{def}}{=} \{s_i^{(k,p)}, p \in [n \cdot m_i]\}, k \in [3]$ and $i \in [d]$ with 1-local use. So that each randomness subset ρ_k^i keeps their ℓ_k^i-local use. We define the composition of G_{ij} and LRs as G'.

(3) According to Theorem 2, the composition of G_{ij} with the same i is PINI-E with the randomness subsets $(\rho_k^i)_{k\in[q_i]}$. We define these compositions as G_i for each i. And the LRs are also PINI-E with randomness subsets $\rho_{s,i}^{(k)}$.

(4) According to Proposition 1, gadget G' is PINI-E with randomness subsets $(\rho_k^i)_{k\in[q_i]}$ which keeps ℓ_k^i-local use, and LR gadgets with 1-local use randomness subsets $\rho_{s,i}^{(i)}$. Therefore according to Theorem 4, G' is still d-private with $3d + \sum_{i\in[n]} q_i$ PRGs among which the $\ell_k^i d$-wise one is used to generate randoms for the ℓ_k^i-local subset, and the other $3d$ PRGs are used to generate randoms for the LR gadgets.

According to the illustration in Sections 2.3 and 2.4, we summarize a proof sketch on the probing security of gadgets' composition in Figure 7.

B. Proofs for LatinAND

B.1. The Security of LatinAND. First we introduce the construction of the matrix L_d. We give a $(d+1) \times d$ matrix as Figure 8. Its first row is $\{1, 2, \ldots, d\}$, and in other rows, the order of sequence is the cyclic shift of its last row except the last row whose first $\frac{d+1}{2}$ elements are $2j-1$ for the j-th element and the rest elements are $2j-d-1$. Then we add a sequence $\{0\}^{d+1}$ as the first column of L_d. We give the construction of L_5 as an instance in Figure 8.

Then let M_d^r be the randomness matrix of r in LatinAND with order d, and we define the mapping $\phi : L_d \to M_d \times L_d$ where $\phi(s_{ij}^{(L)}) \overset{\text{def}}{=} (s_{ij}^{(M)}, s_{ij}^{(L)})$ with $s_{ij}^{(L)} \in L_d$ and $s_{ij}^{(M)} \in M_d^r$. For a d-order LatinAND, we define:

$$\rho_{s_{ij}^{(L)}}^{L} \overset{\text{def}}{=} \left\{ s_{ij}^{(M)} : \phi\left(s_{ij}^{(L)}\right) = \left(s_{ij}^{(M)}, s_{ij}^{(L)}\right) \right\}, \tag{B.1}$$

with $s_{ij}^{(L)} \in [d]$, which are the randomness subsets of randoms r. The randomness matrix of randoms t, called M_d^t, is the mirror symmetry of M_d^r. And the construction of the

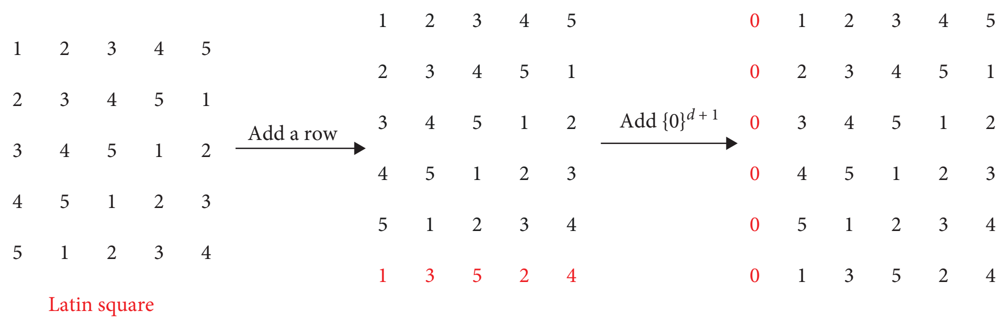


Figure 8: Constructing an L_d from an latin square with $d=5$.

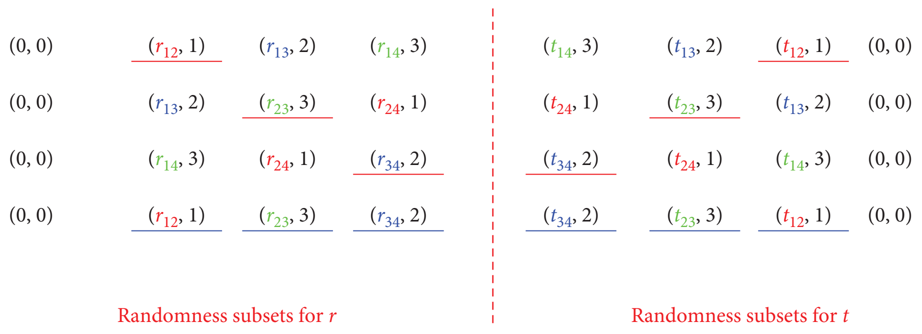


Figure 9: Examples of the randomness subsets for Algorithm 2 with $d=3$. Each element in the same ρ_k^L or ρ_k^R is the same color. Intuitionally in M_d^r, the randoms corresponding to the "initial" latin square of L_d are axial symmetry to the diagonal of latin square, and the randoms at the last row are exactly the randoms at the diagonal. Meanwhile M_d^t is M_d^r's mirror symmetry.

randomness subsets of t is also the mirror symmetry of ρ_k^L for $k\in[d]$, called ρ_k^R. We give an example of M_d^r and M_d^t in Figure 9. Let $M_d \overset{\text{def}}{=} M_d^r + M_d^t$ be the randomness matrix of LatinAND.

Finally, we define N_d as the matrix mixed M_d with the inputs a_i, b_j. More precisely, let:

$$s_{ij}^{(N)} \overset{\text{def}}{=} s_{ij}^{(M)} \oplus a_i b_j\,. \tag{B.2}$$

For example, $s_{12}^{(M)} = r_{12} \oplus t_{13}$ and correspondingly $s_{12}^{(N)} = a_1b_2 \oplus r_{12} \oplus t_{13}$. And we define $|s_{ij}^{(N)}|_I$ as the indice set of the corresponding a_ib_j of $s_{ij}^{(N)}$.

We provide Lemma 3 for the proof of PINI-E and prove it at Appendix C as supproting information.

Lemma B.1. *In M_d^r, there are at most 2 randoms $r_{i_1j_1}\in S_i^k$ and $r_{i_2j_2}\in S_{i'}^{k'}$ satisfying $r_{i_1j_1}=r_{i_2j_2}$ for $r_{ik}, r_{i'k'}\in\rho_j^L$.*

Proposition B.1. *Lemma 3 also works when we replace $S_i^k, S_{i'}^{k'}$ with S_i^{d+1}/S_i^k and $S_{i'}^{d+1}/S_{i'}^{k'}$. More precisely, the randoms pair exists for S_i^{d+1}/S_i^k and $S_{i'}^{d+1}/S_{i'}^{k'}$ iff the randoms pair in Lemma 3 does not exist.*

Lemma 3 and Proposition 2 show that every random is used only twice in the different outputs.

B.2. Proof of Lemma 3

Proof. According to the construction of M_d^r, there always exists $r_{i_1j_1}$ and $r_{i_2j_2}$ satisfying Lemma 3 between S_i^{d+1} and $S_{i'}^{d+1}$. So Lemma 3 is proved. □

B.3. Proof of Proposition 2

Proof. Mention that $S_i^k \cup (S_i^{d+1}/S_i^k) = S_i^{d+1}$ and $S_{i'}^{k'} \cup (S_{i'}^{d+1}/S_{i'}^{k'}) = S_{i'}^{d+1}$, the proof is the same as Lemma 3. □

B.4. Proof of Theorem 6

Proof. There are two steps in our proof, the proof of PINI and the extension to PINI-E.

First we prove the PINI security. Let I be the indice set of inputs. WLOG, we only consider the randoms r, because the other randoms do not weaken the security.

(1) According to the construction of M_d, each pair of a_ib_j and a_jb_i for $i\neq j$ is protected by the same random r_{ij}. As a result, if the random r_{ij} in the probed variables is simulated, we put the corresponding indice i, j into I.

(2) Then we consider the situation where the randoms r of probed variable p are simulated by more than one probe, for example, r_1, r_2 in the probed variable p_1 are simulated by variables p_2 and p_3 because each probe can simulate at most one random of the other probe according to Lemma 3 and Proposition 2. Consider Algorithm 2, the input indice of each

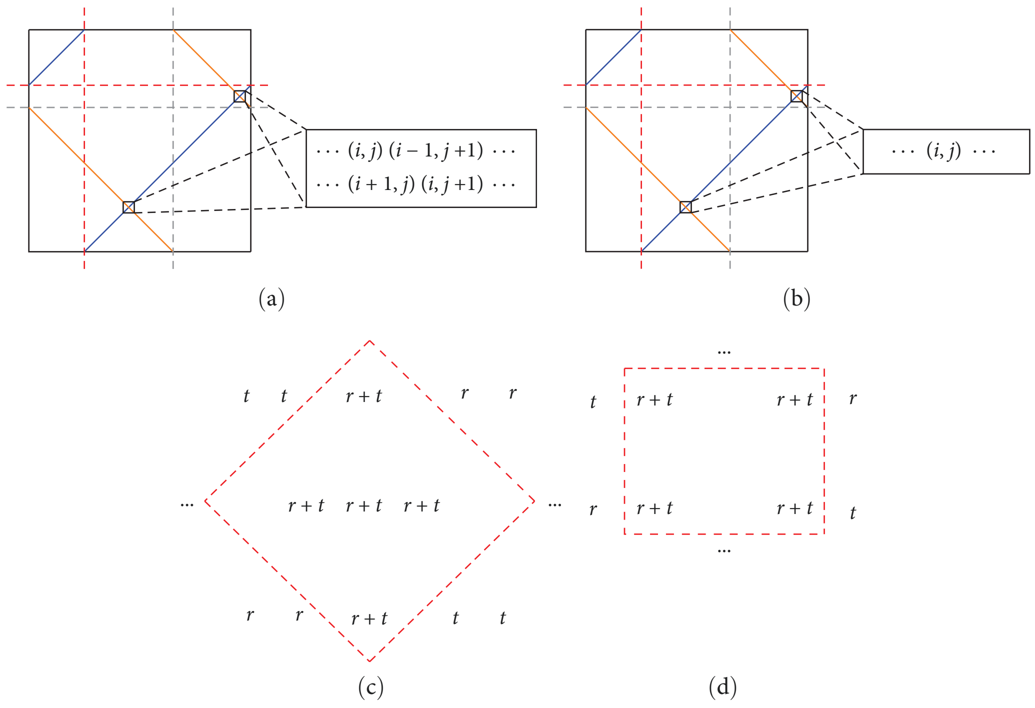


Figure 10: Subparts (a, b) are the two cases for probing randomness subsets ρ_k^L and ρ_k^R at c_i for $i \in [d]$, where the blue line corresponds to the probed ρ_k^L and the orange one corresponds to ρ_k^R. And the (i, j) is the index of (ρ_i^L, ρ_j^R) in M_d. Subpart (a) refers to the situation where the probed ρ_i^L and ρ_j^R do not intersect at M_d, while subpart (b) does. Subparts (c, d) are two cases of the intersections of more than one probes to ρ_i^L and ρ_i^R, where we assume there are three probes for both ρ_i^L and ρ_i^R. The r (resp., t) refers to the probed randoms at ρ^L (resp., ρ^R), and $r+t$ refers to that both randoms r and t are probed, defined as bare. The red squares in subparts (c, d) are used to stress the bare variables.

intermediate step of c_i for $i \in [d]$ are continuous. More precisely, for the adjacent elements $r_{i_1 j_1}$ and $r_{i_2 j_2}$ in S_i^k of M_d^r, there must be $i_1 = i_2$ or $j_1 = j_2$, which means the input indices corresponding to the randoms are also continuous. Thus in this case, each additional probe only adds 1 more indice into I.

(3) For the intermediate steps of c_{d+1}, the adjacent elements of S_i^k also own the same indice. And thanks to PIRT, the simulation of c' in PIRT needs all randoms contained in its intermediate steps, which is similar to case 2. Consider s_{ij} and p_{ij}^0 must satisfy PINI, there is only p_{ij}^1 left for the proof of PINI. Since $p_{ij}^1 = a_i b_j \oplus a_i r_{i'j'}$, there are at least 2 probes to simulate $a_i r_{i'j'}$. Therefore the intermediate steps of c_{d+1} also satisfy PINI, the PINI security of LatinAND is deduced.

The proof also works when we only consider the randoms t, the proof is the same as that of r so we omit it. Then we prove the PINI-E of LatinAND. We provide Figure 10 to describe the distribution of proved randoms at M_d when a ρ_i^L and a ρ_j^R are probed.

First, we prove that all intermediate variables must satisfy PINI-E except those at the "intersection" as Figure 10. In other words, an intermediate variables will not break PINI-E unless all its randoms r and t are contained in the probed randomness subsets. Consider the proof of PINI, if the randoms r for some intermediate variable are probed, it still satisfies PINI security because the PINI proof also works with the randoms t. And the situation of probing t is the same. We mention that the only difference between PINI-E and PINI is the probes of randomness subsets, so the intermediate variables mentioned above also satisfy PINI-E.

As a result, we only consider those intermediate variables whose randoms are simulated with their randomness subsets, which is called bare in the rest of the proof.

(1) We prove that there are at most two bare $s_{ij}^{(N)}$ when there are one probe for ρ_k^L and $\rho_{k'}^R$, respectively, with $k, k' \in [d]$. First, we consider the i-th row for $i \in [d]$. In this case, the proof is equal to prove there are at most two intersections in Figures 10(a) and 10(b). Mention that the included angle of either the blue line or the orange line in Figure 10 and the edges of M_d are $\frac{\pi}{4}$, we know that the blue line is perpendicular to the orange one. We assume, there are more than two intersections of these lines, i.e., there are three or four intersections. In this case there must be two intersections at the extreme points of one of the dotted lines, WLOG, we assume they are at the extreme points of the vertical line. However, according to M_d^r, if $s_{1j}^{(M^r)} \in \rho_k^L$, $s_{d,j+1}^{(M^r)} \in \rho_k^L$. And for M_d^t, $s_{1j}^{(M^t)} \in \rho_k^R$ refers to $s_{d,j-1}^{(M^t)} \in \rho_k^R$. Hence, there must not be two intersections at the extreme points of the dotted line, which means there are no more than two intersections. So, we prove the proposition. WLOG, in the rest of the proof we assume there are two bare $a_i b_j$ for each two

probes of ρ_k^L and $\rho_{k'}^R$ with $k, k' \in [d]$ (generally, if $s_{1j}^{(M^r)}$, $s_{dj'}^{(M^r)} \in \rho_k^L$ and $s_{1k}^{(M^t)}, s_{dk'}^{(M^t)} \in \rho_{k'}^R$, we have $|j-k| = |j'-k'| - 2$, which comes from the construction of L_d. Therefore we only consider $j = k$).

(2) Then we show that indice set $|s_{ij}^{(N)}|_I \cup |s_{ij'}^{(N)}|_I$ for $1 < i \leqslant d$ and $1 < j, j' \leqslant i$ satisfies $||s_{ij}^{(N)}|_I \cup |s_{ij'}^{(N)}|_I| = 3$, which can be got from the construction of N_d. And it also works for $j, j' > i$. More precisely, for a fixed i, there is always an input indice $i+1$ in the $s_{ij}^{(N)}$ with $1 < j, j' \leqslant i$, and i for $j, j' > i$. Specially, $|s_{i,i}^{(N)}|_I \cap |s_{i,i+1}^{(N)}|_I = \{i+1\}$. Meanwhile, there are $|s_{ij}^{(N)}|_I \cap |s_{i'j}^{(N)}|_I = \{j\}$ for $i, i' < j$ and $|s_{ij}^{(N)}|_I \cap |s_{i'j}^{(N)}|_I = \{j-1\}$ for $i, i' \geqslant j$. Therefore, each probe for adjacent $s_{ij}^{(N)}$ can provide at most one more indice.

(3) Then we prove there are at least two probes to get $S_i^k / S_{i'}^k$ at N_d. Mention that there is no $a_i b_j$ appearing directly in the intermediate variables at PIRT, the only way to get $s_{ij}^{(N)}$ is to probe both S_j^i and S_{j-1}^i at N_d. As a result, getting m $S_i^k / S_{i'}^k$ needs at least $2m$ probes. Moreover, consider the distribution of $a_i b_j$ in N_d, which we discuss at last case, the most efficient probe method for the adversary is to probe the continuous sequence $S_i^k / S_{i'}^k$ instead of $s_{ij}^{(N)}$ with discrete j, thus we omit other situations in the rest of the proof.

(4) According to case 2 and case 3 above, we consider the probes containing the adjacent ρ_m^L and ρ_n^R for $m, n \in [d]$ and the corresponding intermediate variables, the "adjacent" means the subsets are adjacent at M_d^r and M_d^t and intuitionally the adjacent subsets can also describe as the orange and blue lines in Figure 10 with larger thickness. Figures 10(a) and 10(b) show two different situations of the intersections of ρ_m^L and ρ_n^R. In the situation of Figure 10(a), there are no $s_{ij}^{(N)}$ are bare, so we only consider the situation of Figure 10(b) according to case 1. And the case of Figure 10(b) can be divided into two different situations as Figures 10(c) and 10(d). The "shapes" (enclosed by the red lines at Figures 10(c) and 10(d)) of the bare $s_{ij}^{(N)}$ may be square, hexagon or octagon, which depends on the choice and the number of probes. All the shapes can be contained at a square with side length ℓ where $\ell \overset{\text{def}}{=} \frac{\ell_1 + \ell_2}{2}$ and ℓ_1 (resp., ℓ_2) is the number of probes of ρ_i^L (resp., ρ_i^R). We assume the number of probes at ρ_i^L and ρ_i^R is equivalent, and we put the explanations about the propositions and assumption above into Appendix C as supproting information. In the rest of the proof, we assume the shape of the bare variables is the ℓ-length square proposed at Figure 11.

(5) According to case 2, the adjacent $s_{ij}^{(N)}$ at the same row or the same column have at least one same indice if the adjacent elements satisfy $j \leqslant i$ or $j > i$. Therefore, if any $S_i^k / S_{i'}^k$ for N_d is probed, there are at least 2κ more probes for the randomness subsets needed to make the probe bare with $\kappa \overset{\text{def}}{=} |S_i^k / S_{i'}^k|$. And consider that there are at most $\kappa + 1$ indices contained at $S_i^k / S_{i'}^k$, its simulation satisfies PINI-E. Mention that $\kappa < \frac{d}{2}$, we consider other probes contained by the ℓ-length square mentioned at case 4. Note that, there are two intersections for the probes of randomness subsets, we assume there are $\alpha \leqslant \frac{d}{6}$ probes for each of ρ_i^L and ρ_i^R, and probe all sequences contained by the two squares, which are $2 \times \alpha + 2 \times (2 \times \alpha) = 6\alpha \leqslant d$ probes totally according to the discussion at case 3. First, we consider the square with the probe simulated before, each other sequence in this square provides at most two more input indices, one of the additional indice comes from the situation with $i, i' \leqslant j$ mentioned at case 2 and the other possible indice comes from $i, i' > j$. Therefore the probes in this square provide at most $(\alpha + 1) + 2 \times (\alpha - 1) = 3\alpha - 1 < \frac{d}{2}$, and the indices for the two squares are less than d. Hence, we prove the PINI-E security for c_i and their intermediate steps with $i \in [d]$.

(6) The PINI-E security for c_{d+1} and its intermediate steps is trivial. Since the adjacent $s_{d+1,j}^{(M^r)}$ and $s_{d+1,j}^{(M^t)}$ are different with any other elements at other rows, the probed $s_{d+1,j}^{(M)}$ are not adjacent when we probe the adjacent $s_{ij}^{(M)}$ with $i \in [d]$, which means there are twice probes needed to probe the bare $s_{d+1,j}^{(N)}$. Therefore the PINI-E security also works for c_{d+1} and its intermediate steps, and we deduce Theorem 6. □

Remark B.1. In this part we give a retrospect of the proof. First, we prove LatinAND is PINI with either ρ_i^L or ρ_i^R. Then in the proof of PINI-E, we reduce the scope of potential "unsecure" intermediate variables and finally prove that all variables are PINI-E. More precisely,

(1) In case 1 we provide the distribution of the bare $s_{ij}^{(N)}$ with the single probe of both ρ_i^L and ρ_i^R. Consider the PINI security of LatinAND with either ρ_i^L or ρ_i^R for $i \in [d]$, the intermediate variables which are not bare must satisfy PINI, and thus satisfy PINI-E. As a result, we only consider these bare $s_{ij}^{(N)}$.

(2) In case 2 we analyze the indices of the elements at N_d at the same row or column.

(3) In case 3 we provide the relation between the number of probes and the constructions of probed sequences at N_d. Consider the indice distribution of N_d discussed at case 2, we determine the most efficient probing method to get most indices.

(4) In case 4 we extend the conclusion at case 1 from the single probe of both ρ_i^L and ρ_i^R to several probes. Also, we discuss the "shape" of the bare variables

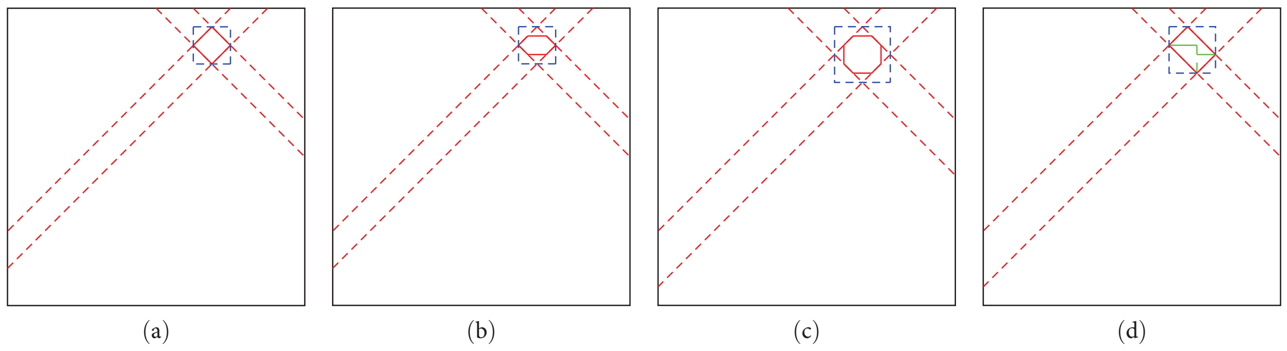


Figure 11: Subparts (a–d) are the different shapes of bare $s_{ij}^{(N)}$ and their corresponding extended squares in the proof of Theorem 6. The enclosed parts of the dotted lines refer to the $s_{ij}^{(N)}$ which randoms r or t are probed, and the enclosed parts of the full lines are the bare $s_{ij}^{(N)}$. Meanwhile, the blue squares refer to the extended squares.

and extend it into a square in N_d which is easier to prove security. The details of why the "shape" is exactly what we claim and how the size of the extended square comes are put at Appendix C as supproting information. According to case 1, we only need to prove the security of the $s_{ij}^{(N)}$ contained in the extended squares.

(5) In case 5 we prove the PINI-E security of $s_{ij}^{(N)}$ and their intermediate steps with the conclusion in case 2 and 3 for $i \in [d]$.

(6) In case 6 we prove the PINI-E of the rest intermediate variables (i.e., c_{d+1} and its intermediate steps).

B.5. Explanations about the Proof of Theorem 6

B.5.1. The Enclosed Part at Figures 10(c) and 10(d). We provide the different shapes at Figure 11, in which the enclosed parts of the dashed lines refer to the probed randomness subsets and those with full lines refer to the bare variables at N_d. The Figure 11(a) is the situation where there is an element of N_d at each vertex of the dashed square exactly, therefore the full line square is also a square. The Figure 11(b) is the situation where there is no element at the top and bottom vertexes of the dashed square, thus the remained shape is hexagon, note that if the top vertex is not element, the bottom one is neither because of the symmetry of the construction. Figure 11(c) is the situation where there is no element at all 4 vertexes of the dashed square, therefore the remained shape is octagon.

B.5.2. The Figure of the Enclosed Part and the Scaled Square. Figure 11(d) shows the situation where the probe number of ρ_i^L and ρ_i^R is unequal. With the green full lines at Figure 11(d), we know that the red rectangle can be contained by a square with side length $\frac{\ell_1+\ell_2}{2}$ (this conclusion comes from elementary geometry and we omit the detailed proof), where ℓ_1 and ℓ_2 is the probe number of ρ_i^L and ρ_i^R and we assume $\ell_1 > \ell_2$. The blue squares at Figure 11 are the scaled ones at the proof of Theorem 6, easy to see that its side length is $\frac{\ell_1+\ell_2}{2}$ and it contains all bare $s_{ij}^{(N)}$.

C. Evaluation of the Randomness Cost for SecADD

In this part, we will calculate the cost of randomness in SecADD. We consider the operations are over $\mathbb{F}_{2^w}$ and let $\ell \overset{\text{def}}{=} \lceil \log_2 (w-1) \rceil$.

Now we calculate the cost in SecADD with multiple PRGs. According to Theorem 4, we use a set of d-wise PRGs to generate randoms used by LR gadgets as they make a 1-local use of each $\rho_{s,i}^{(k)}$ for $k \in [3]$, and a set of d-wise PRGs to generate r_{ij} and t_{ij} in LatinAND gadget because they make a 1-local use of each ρ_k^L and ρ_k^R. In the following, we will separately discuss the number of PRGs and randoms with either $\mathscr{R}_1$ or $\mathscr{R}_2$.

When using $\mathscr{R}_1$, we calculate the number of PRGs and randoms in different situations. First, we consider PRG$^{(r)}$ which is used to generate the randoms in ρ_k^L and ρ_k^R. According to the maximum distance separable (MDS) conjecture [29], we have the following inequality, where there are 2ℓ LatinAND gadgets used in a SecADD:

$$2\ell \cdot \frac{d+1}{2} \leqslant 2^w, \tag{C.1}$$

from which we have $d \leqslant \frac{2^w}{\ell} - 1$. In our implementation, we set $w = 8$ and $\ell = 5$ (the input length of SecADD are set as 32. When randoms are needed, we use 4 outputs of the 8-bit PRGs to generate a 32-bit random. In other words, there are 4 PRGs needed for a random with different seeds). It means that we can use $4d$ d-wise PRGs to generate all randoms in ρ_k^L or ρ_k^R for any $d \leqslant \lfloor \frac{2^w}{\ell} - 1 \rfloor = 50$. Then we calculate PRG$_k^{(s)}$ for $\rho_{s,i}^{(k)}$ with some $k \in [3]$, we have:

$$3\ell + 3 \leqslant 2^w\,. \tag{C.2}$$

From the value of d and ℓ given above, we know that it is satisfied. Therefore, we need $4d$ d-wise PRGs to generate the randoms in $\rho_{s,i}^{(k)}$ for $k \in [3]$. According to all the calculations

Table 5: The random bits used in SecADD with the application of R_1, R_2, and the situation without PRGs with $d \in [3]$ for R_2 and $d \in [50]$ for R_1.

	The randomness cost		
	R_1	R_2	No PRGs
$\mathrm{PRG}^{(r)}$	$64d^2$	$64d\lceil 2\sqrt{5(d+1)}\rceil$	$320d(d+1)$
$\mathrm{PRG}_i^{(s)}$ for all $i \in [3]$	$96d^2$	$864d$	$1,728d$
Total cost	$160d^2$	$64d\lceil 2\sqrt{5(d+1)}\rceil + 864d$	$320d^2 + 1,728d$

above, we know that we need $2 \cdot 32d^2 + 3 \cdot 32d^2 = 160d^2$ bits of randomness for the whole SecADD algorithm.

Then we consider the case using $\mathscr{R}_2$, and we will also calculate PRGs and randoms, respectively. As the output of $\mathscr{R}_2$ is 3-wise independent, according to Theorem 4, we always need d-wise PRGs, and thus the security order d is no more than 3. Let x_r and x_s be the numbers of randoms needed in $\mathscr{R}_2$ for ρ_k^L or ρ_k^R and $\rho_{s,i}^{(k)}$ for some $k \in [3]$. First, we consider $\mathrm{PRG}^{(r)}$, we can get the following inequality by the definition of $\mathscr{R}_2$:

$$2\ell \cdot \frac{(d+1)}{2} \leqslant \left(\frac{x_r}{2}\right)^2, \tag{C.3}$$

therefore $x_r \geqslant 2\sqrt{\ell(d+1)} = 2\sqrt{5(d+1)}$. Similarly, for $\mathrm{PRG}_k^{(s)}$ we have:

$$3\ell + 3 \leqslant \left(\frac{x_s}{2}\right)^2, \tag{C.4}$$

thus we have $x_s \geqslant 2\sqrt{3\ell+3} = 6\sqrt{2}$. Then we know that we need $8d$ $\mathscr{R}_2$ with $w\lceil 2\sqrt{5(d+1)}\rceil = 8\lceil 2\sqrt{5(d+1)}\rceil$ bit seeds to generate all ρ_k^L and ρ_k^R in LatinAND, and $12d$ $\mathscr{R}_2$ with $w\lceil 6\sqrt{2}\rceil = 72$ bit seeds to generate all randoms in LR. We compare the randomness cost of R_1, R_2 and situation without PRGs in Table 5.

Then, we discuss the case when a set of PRGs are used by mutliple SecADD. We only consider the use of PRG R_1, and the maximum number of SecADD can be calculated by $n = \lfloor \frac{2^w}{2\ell \cdot \lceil \frac{d}{2} \rceil} \rfloor$. It means that there are 2ℓ LatinAND in each SecADD and each randomness subset of LatinAND contains at most $\lceil d/2 \rceil$ elements, and thus $2\ell \cdot \lceil d/2 \rceil$ elements are contained by a ρ_k^L in a SecADD algorithm. And, 2^w refers to the number of output variables of a PRG. Hence, $\lfloor \frac{2^w}{2\ell \cdot \lceil \frac{d}{2} \rceil} \rfloor$ is the maximum number of SecADD for one set of R_1. We set $d = 1$, $w = 8$, and $\ell = 5$ which is quite a practical relevant setting. Then, we have $n = 25$. Considering that, in Table 5, one SecADD using R_1 and no PRGs requires $160d^2$ and $320d^2 + 1,728d$ random bits, respectively. Therefore, the randomness cost can be reduced by a factor of up to $25(320d^2 + 1,728d)/160d^2 = 320$.

Data Availability

The source code data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Bohan Wang and Weijia Wang contributed in the writing, investigation, and methodology. Qian Sui contributed in the software and methodology. Fanjie Ji contributed in the software and validation. Chun Guo and Weijia Wang contributed in the resources and funding. Weijia Wang contributed in the conceptualization and validation. Chun Guo contributed in the investigation.

Acknowledgments

This work was financially supported by the National Key Research and Development Program of China (No. 2021YFA1000600), the Program of Qilu Young Scholars (Grant Nos. 61580089963177 and 61580082063088) of Shandong University, the Program of Taishan Young Scholars of the Shandong Province, the National Natural Science Foundation of China (Grant Nos. 62032014, 62002202, and 62002204), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025), the Department of Science & Technology of Shandong Province (SYS202201), and the Quan Cheng Laboratory (QCLZD202306).

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO '96*, N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113, Springer, Berlin, Heidelberg, 1996.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO' 99*, M. J. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, Springer, Berlin, Heidelberg, 1999.

[3] A. Duc, S. Dziembowski, and S. Faust, "Unifying leakage models: from probing attacks to noisy leakage," in *Advances in Cryptology—EUROCRYPT 2014*, P. Q. Nguyen and E. Oswald, Eds., vol. 8441 of *Lecture Notes in Computer Science*, pp. 423–440, Springer, Berlin, Heidelberg, 2014.

[4] G. Barthe, S. Belaïd, F. Dupressoir et al., "Strong non-interference and type-directed higher-order masking," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds., pp. 116–129, ACM, Vienna, Austria, 2016.

[5] G. Cassiers and F.-X. Standaert, "Trivially and efficiently composing masked gadgets with probe isolating non-interference," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2542–2555, 2020.

[6] J.-S. Coron, A. Greuet, and R. Zeitoun, “Side-channel masking with pseudo-random generator,” in *Advances in Cryptology—EUROCRYPT 2020*, A. Canteaut and Y. Ishai, Eds., vol. 12107 of *Lecture Notes in Computer Science*, pp. 342–375, Springer, Cham, 2020.

[7] Y. Ishai, E. Kushilevitz, X. Li et al., “Robust pseudorandom generators,” in *Automata, Languages, and Programming. ICALP 2013*, F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, Eds., vol. 7965 of *Lecture Notes in Computer Science*, pp. 576–588, Springer, Berlin, Heidelberg, 2013.

[8] Y. Ishai, A. Sahai, and D. A. Wagner, “Private circuits: securing hardware against probing attacks,” in *Advances in Cryptology—CRYPTO 2003*, D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, pp. 463–481, Springer, Berlin, Heidelberg, 2003.

[9] R. M. Needham and D. J. Wheeler, *Tea Extensions*, Report, Cambridge University, 1997.

[10] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK lightweight block ciphers,” in *Proceedings of the 52nd Annual Design Automation Conference*, pp. 1–6, ACM, San Francisco, CA, USA, 2015.

[11] C. Beierle, A. Biryukov, L. C. dos Santos et al., “Alzette: a 64-bit ARX-box,” in *Advances in Cryptology—CRYPTO 2020*, D. Micciancio and T. Ristenpart, Eds., vol. 12172 of *Lecture Notes in Computer Science*, pp. 419–448, Springer, Cham, 2020.

[12] C. Beierle, A. Biryukov, L. C. dos Santos et al., “Lightweight AEAD and hashing using the sparkle permutation family,” *IACR Transactions on Symmetric Cryptology*, vol. 2020, no. S1, pp. 208–261, 2020.

[13] L. Goubin, “Cryptographic Hardware and Embedded Systems —CHES 2001,” *A sound method for switching between Boolean and arithmetic masking*, vol. 2162, Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, 2001.

[14] M. Van Beirendonck, J.-P. D’Anvers, and I. Verbauwhede, “Analysis and comparison of table-based arithmetic to Boolean masking,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, pp. 275–297, 2021.

[15] J.-S. Coron, J. Großschädl, and P. K. Vadnala, “Secure conversion between Boolean and arithmetic masking of any order,” in *Cryptographic Hardware and Embedded Systems—CHES 2014*, L. Batina and M. Robshaw, Eds., vol. 8731 of *Lecture Notes in Computer Science*, pp. 188–205, Springer, Berlin, Heidelberg, 2014.

[16] J.-S. Coron and A. Tchulkine, “A new algorithm for switching from arithmetic to Boolean masking,” in *Cryptographic Hardware and Embedded Systems—CHES 2003*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds., vol. 2779 of *Lecture Notes in Computer Science*, pp. 89–97, Springer, Berlin, Heidelberg, 2003.

[17] B. Debraize, “Efficient and provably secure methods for switching from arithmetic to Boolean masking,” in *Cryptographic Hardware and Embedded Systems—CHES 2012*, E. Prouff and P. Schaumont, Eds., vol. 7428 of *Lecture Notes in Computer Science*, pp. 107–121, Springer, Berlin, Heidelberg, 2012.

[18] M. Hutter and M. Tunstall, “Constant-time higher-order Boolean-to-arithmetic masking,” *Journal of Cryptographic Engineering*, vol. 9, no. 2, pp. 173–184, 2019.

[19] M. Karroumi, B. Richard, and M. Joye, “Addition with blinded operands,” in *Constructive Side-Channel Analysis and Secure Design. COSADE 2014*, E. Prouff, Ed., vol. 8622 of *Lecture Notes in Computer Science*, pp. 41–55, Springer, Cham, 2014.

[20] J.-S. Coron, J. Großschädl, M. Tibouchi, and P. K. Vadnala, “Conversion from arithmetic to Boolean masking with logarithmic complexity,” in *Fast Software Encryption*, G. Leander, Ed., vol. 9054 of *Lecture Notes in Computer Science*, pp. 130–149, Springer, Berlin, Heidelberg, 2015.

[21] L. Bettale, J.-S. Coron, and R. Zeitoun, “Improved high-order conversion from Boolean to arithmetic masking,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 2, pp. 22–45, 2018.

[22] J.-S. Coron, “High-order conversion from Boolean to arithmetic masking,” in *Cryptographic Hardware and Embedded Systems—CHES 2017*, W. Fischer and N. Homma, Eds., vol. 10529 of *Lecture Notes in Computer Science*, pp. 93–114, Springer, Cham, 2017.

[23] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. Van Vredendaal, “Masking kyber: first- and higher-order implementations,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 4, pp. 173–214, 2021.

[24] V. Migliore, B. Gérard, M. Tibouchi, and P.-A. Fouque, “Masking dilithium,” in *Applied Cryptography and Network Security*, R. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, Eds., vol. 11464 of *Lecture Notes in Computer Science*, pp. 344–362, Springer, Cham, 2019.

[25] G. Barthe, S. Belaïd, T. Espitau et al., “Masking the GLP lattice-based signature scheme at any order,” in *Advances in Cryptology—EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds., vol. 10821 of *Lecture Notes in Computer Science*, pp. 354–384, Springer, Cham, 2018.

[26] S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud, “Randomness Complexity of Private Circuits for Multiplication,” in *Advances in Cryptology—EUROCRYPT 2016*, M. Fischlin and J. S. Coron, Eds., vol. 9666 of *Lecture Notes in Computer Science*, pp. 616–648, Springer, Berlin, Heidelberg, 2016.

[27] P. M. Kogge and H. S. Stone, “A parallel algorithm for the efficient solution of a general class of recurrence equations,” *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, 1973.

[28] W. Wang, F. Ji, J. Zhang, and Y. Yu, “Efficient private circuits with precomputation,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 2, pp. 286–309, 2023.

[29] B. Segre, “Curve razionali normali ek-archi negli spazi finiti,” *Annali Di Matematica Pura ed Applicata*, vol. 39, no. 1, pp. 357–379, 1955.

[30] T. Fritzmann, M. Van Beirendonck, D. B. Roy et al., “Masked accelerators and instruction set extensions for post-quantum cryptography,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 1, pp. 414–460, 2022.