

A Graph-Theoretic Framework for Randomness Optimization in First-Order Masked Circuits

Dilip Kumar S. V.[✉], Benedikt Gierlichs[✉], Ingrid Verbauwhede[✉]

COSIC, ESAT, KU Leuven, Leuven, Belgium

dilipv46@gmail.com, {benedikt.gierlichs, ingrid.verbauwhede}@esat.kuleuven.be

Abstract—We present a generic, automatable framework to reduce the demand for fresh randomness in first-order masked circuits while preserving security in the glitch-extended probing model. The method analyzes the flow of randomness through a circuit to establish security rules based on the glitch-extended probing model. These rules are then encoded as an interference graph, transforming the optimization challenge into a graph coloring problem, which is solved efficiently with a DSATUR heuristic. Crucially, the optimization only rewires randomness inputs without altering core logic, ensuring seamless integration into standard EDA flows and applicability to various gadgets like *DOM-indep* (Domain-Oriented Masking) and HPC (Hardware Private Circuits). On 32-bit adder architectures, the framework substantially reduces randomness requirements by 79–90%; for instance, the Kogge–Stone adder’s requirement of 259 unique random inputs is reduced to 27. All optimized designs were evaluated using PROLEAD, with the leakage results indicating compliance with first-order glitch-extended probing security.

Index Terms—Side-Channel Analysis, Hardware Masking, Glitch-Extended Probing Model, Security-Aware Optimization

I. INTRODUCTION

Side-channel attacks (SCAs) [1], [2] pose a persistent threat to the security of cryptographic hardware implementations. Masking [3] is a state-of-the-art countermeasure that provides provable security by splitting sensitive variables into randomized shares and restricting computations to incomplete sets, ensuring that no intermediate correlates with the secret. Beyond the classical probing model [4], the glitch-extended probing model [5] captures physical effects more accurately and is now the standard reference for secure hardware implementations. With these models established, current research focuses on optimizing key hardware metrics such as area, latency, and randomness cost.

Two main approaches to hardware masking exist: algorithm-level and gate-level. Algorithm-level schemes such as Threshold Implementations (TI) [6] achieve first-order security with minimal randomness by enforcing algebraic constraints, but extending them to arbitrary functions is difficult [7]. Gate-level masking, exemplified by Domain-Oriented Masking (DOM) [8], replaces standard AND/XOR gates with masked versions, allowing generic protection of any circuit and straightforward integration into EDA design flows.

This work was supported by the Flemish Government through the Cybersecurity Research Programme (CRPF, VOEWICS02) and by the European Commission through the Horizon 2020 ERC Advanced Grant “Belfort” (Grant Agreement 101020005 695305). The authors thank Quinten Norga for helpful discussions and the anonymous reviewers for constructive feedback.

Despite its flexibility, the efficiency of gate-level masking is often constrained by the high cost of randomness, which is a primary implementation bottleneck. The most straightforward strategy is to assign unique fresh random bits to every masked AND gate, which guarantees statistical independence between intermediate signals and simplifies formal security proofs. However, this strategy is prohibitively expensive in practice, requiring substantial on-chip randomness generation from high-throughput PRNGs, which in turn increases area overhead. Consequently, optimizing randomness usage is crucial and challenging, since careless reuse may introduce statistical correlations that break security.

While several circuit-level methods reduce randomness [9]–[11], they are often tailored to specific circuits, offering substantial savings but requiring careful, design-specific tuning. Other works address safe randomness reuse either by designing composable gadgets, as in the proposal by Knichel and Moradi [12], to secure any first-order circuit with only six fresh random bits (albeit at the cost of significant area and latency), or by clustering masked gadgets to enable reuse within each cluster for higher-order masking [13].

This work introduces a generic and automatable methodology to reduce the demand for fresh randomness in first-order masked circuits without requiring structural redesign. The method applies to combinational circuits, typically expressed in terms of 2-input AND and XOR primitives, and guarantees preservation of security in first-order glitch-extended probing model. Optimization is achieved by simply reassigning the wiring of randomness inputs while leaving the core logic unmodified, which ensures compatibility with standard design flows.

The remainder of the paper is organized as follows. Section II motivates and formalizes the problem. Section III introduces the methodology for tracking randomness through a circuit and formulating the corresponding security constraints, while Section IV resolves these constraints using graph coloring [14]. Section V reports results on 32-bit adders and the Boyar–Peralta AES S-box [15], and Section VI reports the security evaluation using PROLEAD [16].

II. MOTIVATION AND PROBLEM STATEMENT

A. Core Observation

The motivation for this work arises from a basic yet crucial property of the *DOM-indep* masked AND gate [8], illustrated

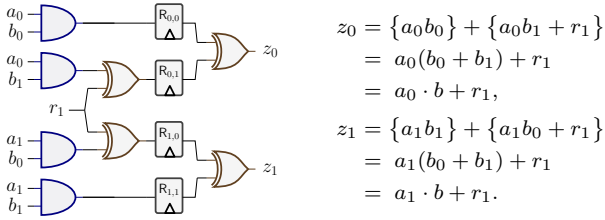


Fig. 1: First-order DOM-*indep* [8].

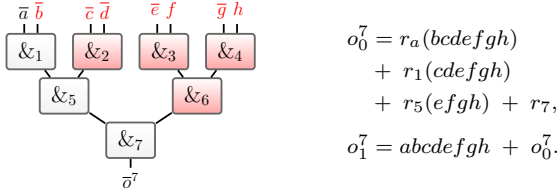


Fig. 2: Multiplication tree and expressions for eight-input AND.

in Fig. 1. Consider $\bar{z} = \text{DOM-indep}(\bar{a}, \bar{b})$, where \bar{a} and \bar{b} are two-share encodings of bits a and b with masks r_a and r_b : $a_0 = r_a$, $a_1 = a + r_a$, $b_0 = r_b$, $b_1 = b + r_b$. The gate forms four partial products, two blinded with a fresh mask r_1 , each stored in a register $\{\cdot\}$. Throughout this work, $+$ denotes XOR and \cdot denotes AND. As shown in the simplification in Fig. 1, $b_0 + b_1 = r_b + (b + r_b) = b$, so the outputs depend only on b while the mask r_b is pruned, meaning randomness from the second input does not propagate to the output [17].

The pruning effect is amplified in larger compositions. In the eight-input AND tree of Fig. 2, inputs \bar{a} to \bar{h} are combined by DOM-*indep* gates $\&E_1$ to $\&E_7$ with fresh bits r_1 to r_7 , and intermediate outputs are \bar{o}^t . We take the left child as the first input and the right child as the second input. At each node, the relation $b_0 + b_1 = b$ ensures that the second input’s randomness is pruned, so only the first input’s mask and the node’s fresh bit propagate upward. Concretely, $\bar{o}^1 = \text{DOM-indep}(\bar{a}, \bar{b})$ depends on $\{r_a, r_1\}$ (prunes r_b); $\bar{o}^5 = \text{DOM-indep}(\bar{o}^1, \bar{o}^2)$ depends on $\{r_a, r_1, r_5\}$ (prunes $\{r_c, r_d, r_2\}$); and $\bar{o}^7 = \text{DOM-indep}(\bar{o}^5, \bar{o}^6)$ depends on $\{r_a, r_1, r_5, r_7\}$ (prunes $\{r_e, r_f, r_g, r_h, r_3, r_4, r_6\}$). Along the left spine $\&E_1 \rightarrow \&E_5 \rightarrow \&E_7$, the fresh bits r_1, r_5, r_7 propagate together with the mask of a (r_a), while the input masks r_b to r_h and the fresh bits r_2, r_3, r_4, r_6 are eliminated in the red subtrees or at their parents. The final equations are in Fig. 2. This opens the possibility of safely reusing the eliminated random bits elsewhere in the circuit.

The pruning phenomenon extends beyond DOM-*indep* and also occurs in composable gadgets such as HPC [18], as will be discussed in Section V-B.

B. Problem Statement

We formalize randomness optimization in first-order, gate-level masked circuits as an allocation problem. At the specification level, each masked AND gate is assigned a unique *symbolic random variable* (e.g., r_1, r_2, \dots), reflecting the unoptimized assumption that every gate requires an independent fresh bit. In hardware, these symbolic variables are instantiated by *physical random bits* generated by a randomness source (e.g., a PRNG) and distributed to the gates. The objective is to define a mapping

$\text{phys} : \mathcal{R}_{\text{sym}} \rightarrow \mathcal{R}_{\text{phys}}$ from symbolic variables to a minimal set of physical random bits. This allows multiple symbolic variables (e.g., r_i and r_j) to share the same physical bit, i.e., $\text{phys}(r_i) = \text{phys}(r_j)$, when such reuse is proven safe, thereby minimizing randomness cost while maintaining first-order security under the glitch-extended probing model.

Our methodology addresses this allocation in three stages. First, *Propagation Analysis* traces each symbolic random from its origin through the circuit and records all intermediate signals it influences. Second, *Constraint Formulation* establishes security rules that forbid symbolic randoms that could be observed together under the glitch-extended probing model from sharing the same physical bit. Third, *Constraint-Driven Optimization* casts these rules as a graph coloring problem and solves it to obtain a secure mapping that minimizes the number of physical random bits required.

The first two steps, tracking randomness propagation and formulating interaction constraints, are described in Section III, which concludes with a small illustrative example. The third step, the optimization strategy derived from these constraints, is presented in Section IV.

III. TRACKING RANDOMNESS AND DEFINING CONSTRAINTS

A. Share Annotation and Propagation Analysis

To enable a systematic analysis of arbitrary first-order masked circuits built from 2-input AND and XOR gates, we represent each AND gate with its masked counterpart using the DOM-*indep* gadget and model XOR as a share-wise linear operation. Pipeline registers are inserted to equalize logic depth at gate inputs; functionally, these registers simply duplicate signals without altering their values.

For each signal share, we associate three disjoint sets, $\mathcal{S}(\cdot)$ for unshared inputs, $\mathcal{M}(\cdot)$ for input masks, and $\mathcal{R}(\cdot)$ for gate-randomness, which together form a complete bookkeeping mechanism where membership in each set is determined by the share’s functional dependence on variables of that type. For example, in Fig. 2, the share o_0^7 has $\mathcal{S}(o_0^7) = \{b, c, d, e, f, g, h\}$, $\mathcal{M}(o_0^7) = \{r_a\}$, and $\mathcal{R}(o_0^7) = \{r_1, r_5, r_7\}$. In this work, our focus is on the \mathcal{R} sets, within which we seek to safely reuse bits to reduce the total randomness cost, while still tracking \mathcal{S} and \mathcal{M} to account for all data dependencies in the circuit. The following rules define how these sets are initialized and updated for each circuit element.

1) *Initialization*: For a primary input a with mask r_a , the shares are $a_0 = r_a$ and $a_1 = a + r_a$. The corresponding sets are $\mathcal{S}(a_0) = \emptyset$, $\mathcal{M}(a_0) = \{r_a\}$, $\mathcal{R}(a_0) = \emptyset$, and $\mathcal{S}(a_1) = \{a\}$, $\mathcal{M}(a_1) = \{r_a\}$, $\mathcal{R}(a_1) = \emptyset$.

2) *Masked XOR Gate*: For a masked XOR gate $z = x + y$ computed sharewise as $z_i = x_i + y_i$, the sets are $\mathcal{S}(z_i) = \mathcal{S}(x_i) \cup \mathcal{S}(y_i)$, $\mathcal{M}(z_i) = \mathcal{M}(x_i) \cup \mathcal{M}(y_i)$, $\mathcal{R}(z_i) = \mathcal{R}(x_i) \cup \mathcal{R}(y_i)$. We take the union without attempting symbol-level cancellations, as this conservative over-approximation produces stronger constraints. This might reduce optimization opportunities but ensures no reuse is allowed where it could

compromise security. This is an intentional trade-off, prioritizing the preservation of security over aggressive optimization.

3) *Masked AND Gate*: For a masked AND gate $\&_t$ implementing $z = x \& y$ using the DOM-*indep* gadget, the output share sets update as $\mathcal{S}(z_i) = \mathcal{S}(x_i) \cup \mathcal{S}(y_0) \cup \mathcal{S}(y_1)$, $\mathcal{M}(z_i) = \mathcal{M}(x_i)$, $\mathcal{R}(z_i) = \mathcal{R}(x_i) \cup \{r_t\}$, where r_t is the fresh random bit introduced by the gate. This matches the behaviour in Fig. 2, where only the first input's masks and gate-randoms propagate, while all randomness from the second input is pruned.

Internally, the gadget contains four registers, see Fig. 1, labeled $\&_t.R_{i,j}$ with $i, j \in \{0, 1\}$ denoting the share of the first and second input, respectively. For each register, $\mathcal{S}(\&_t.R_{i,j}) = \mathcal{S}(x_i) \cup \mathcal{S}(y_j)$, $\mathcal{M}(\&_t.R_{i,j}) = \mathcal{M}(x_i) \cup \mathcal{M}(y_j)$, $\mathcal{R}(\&_t.R_{i,j}) = \mathcal{R}(x_i) \cup \mathcal{R}(y_j) \cup \delta_{i \neq j} \{r_t\}$, where $\delta_{i \neq j}$ is 1 if $i \neq j$ and 0 otherwise, meaning the extra $\{r_t\}$ term occurs only for $\&_t.R_{0,1}$ and $\&_t.R_{1,0}$.

4) *Pipeline Registers*: A pipeline register storing share x_i is denoted $P_{x,i}$, where $i \in \{0, 1\}$ is the share index. Since the register output is an exact copy of its input, the tracked sets remain unchanged: $\mathcal{S}(P_{x,i}) = \mathcal{S}(x_i)$, $\mathcal{M}(P_{x,i}) = \mathcal{M}(x_i)$, $\mathcal{R}(P_{x,i}) = \mathcal{R}(x_i)$.

B. Constraints for Randomness Reuse

Glitches from unequal path delays can cause secret shares to recombine, breaking masking security. The standard probing model [19] is insufficient for hardware, so we adopt the glitch-extended probing model of Faust et al. [5]. We formalize probes via the *glitch extension*.

Definition 1 (Glitch Extension [5], [20]). *A probe \mathcal{P} is extended to its glitch extension $\mathcal{P}^E = \text{Extend}(\mathcal{P})$, the set of all signals in its transitive fan-in up to the preceding timing boundary (register or primary input). It is defined recursively as*

$$\mathcal{P}^E = \begin{cases} \bigcup_{u \in \text{fan-in}(\mathcal{P})} u^E, & \text{if } \mathcal{P} \text{ observes the output of} \\ & \text{a combinational gate,} \\ \{\mathcal{P}\}, & \text{if } \mathcal{P} \text{ observes the output of} \\ & \text{a register or a circuit input.} \end{cases}$$

By Def. 1, a probe's extension includes all contributing registers from the preceding pipeline stage. Applying this model to the circuit components of Section III-A:

1) *Masked XOR Gate*: For $z_i = x_i + y_i$: $\text{Extend}(z_i) = \text{Extend}(x_i) \cup \text{Extend}(y_i)$.

2) *Masked AND Gate*: The DOM-*indep* gate presents two probing scenarios. A probe on the final output z_i observes the XOR of two internal registers; its extension is therefore $\mathcal{P}^E = \{\&_t.R_{i,0}, \&_t.R_{i,1}\}$, making them jointly observable. Alternatively, a probe on the first-stage logic for register $\&_t.R_{i,j}$ has an extension that includes the sources of all its inputs—the shares x_i and y_j , and the fresh random bit r_t when applicable. The resulting extension set is $\mathcal{P}^E = \text{Extend}(x_i) \cup \text{Extend}(y_j) \cup \delta_{i \neq j} \{\text{Src}_{r_t}\}$, where Src_{r_t} denotes the source of r_t .

3) *Pipeline Registers*: A probe on the output of a pipeline register $P_{x,i}$ is the base case: $\mathcal{P}^E = \{P_{x,i}\}$.

Definition 2 (Conflict set). *For a probe \mathcal{P} , the conflict set $\mathcal{C}_{\mathcal{P}}$ is the union of gate-randomness over all registers in \mathcal{P}^E :*

$$\mathcal{C}_{\mathcal{P}} = \bigcup_{\text{Reg} \in \mathcal{P}^E} \mathcal{R}(\text{Reg}).$$

Enumerating $\mathcal{C}_{\mathcal{P}}$ for all probes is unnecessary. By Def. 1, the extension of any internal probe is contained in that of the block outputs. Thus, it suffices to analyze only next-stage register inputs, which subsume all internal probes and capture every reuse conflict.

To formalize the security argument, we define the observation available to an attacker. For a probe \mathcal{P} , the observation $\text{Obs}(\mathcal{P})$ is the ordered tuple of outputs of all registers in \mathcal{P}^E . It is expressed as the output of a deterministic function $G_{\mathcal{P}}$ applied to three inputs: the secret-dependent signals $\mathcal{S}_{\mathcal{P}} = \bigcup_{\text{Reg} \in \mathcal{P}^E} \mathcal{S}(\text{Reg})$, the masks $\mathcal{M}_{\mathcal{P}} = \bigcup_{\text{Reg} \in \mathcal{P}^E} \mathcal{M}(\text{Reg})$, and the randoms $(r)_{r \in \mathcal{C}_{\mathcal{P}}}$ from the conflict set. Thus, $\text{Obs}(\mathcal{P}) = G_{\mathcal{P}}(\mathcal{S}_{\mathcal{P}}, \mathcal{M}_{\mathcal{P}}, (r)_{r \in \mathcal{C}_{\mathcal{P}}})$.

Definition 3 (Local No-Reuse Constraint). *Let $\text{phys}(r)$ be the physical random bit assigned to a symbolic random r . For any probe \mathcal{P} , if $r_i, r_j \in \mathcal{C}_{\mathcal{P}}$ with $i \neq j$, then $\text{phys}(r_i) \neq \text{phys}(r_j)$.*

Proposition 1 (Preservation of First-Order Security). *Let an unoptimized circuit be secure in the first-order glitch-extended probing model. If a randomness assignment satisfies the Local No-Reuse Constraint (Def. 3), the optimized circuit remains secure in the same model.*

Proof. For any probe \mathcal{P} , security requires that the distribution of $\text{Obs}(\mathcal{P})$ be independent of $\mathcal{S}_{\mathcal{P}}$. In the unoptimized circuit, each $r \in \mathcal{C}_{\mathcal{P}}$ is realized by a fresh random bit, so $\mathbf{u}_{\text{fresh}} = (u_r)_{r \in \mathcal{C}_{\mathcal{P}}}$ is a vector of independent uniforms and $\text{Obs}_{\text{fresh}}(\mathcal{P}) = G_{\mathcal{P}}(\mathcal{S}_{\mathcal{P}}, \mathcal{M}_{\mathcal{P}}, \mathbf{u}_{\text{fresh}})$. In the optimized circuit, the random input is $\mathbf{u}_{\text{reuse}} = (\text{phys}(r))_{r \in \mathcal{C}_{\mathcal{P}}}$. By Def. 3, $\text{phys}(\cdot)$ is injective on $\mathcal{C}_{\mathcal{P}}$, ensuring that no random bit is reused within $\mathcal{C}_{\mathcal{P}}$; consequently, $\mathbf{u}_{\text{reuse}}$ consists of independent uniforms and has the same distribution as $\mathbf{u}_{\text{fresh}}$. Hence, $\text{Obs}_{\text{reuse}}(\mathcal{P}) = G_{\mathcal{P}}(\mathcal{S}_{\mathcal{P}}, \mathcal{M}_{\mathcal{P}}, \mathbf{u}_{\text{reuse}}) \stackrel{d}{=} \text{Obs}_{\text{fresh}}(\mathcal{P})$. Since the observation distribution is identical, independence from $\mathcal{S}_{\mathcal{P}}$ is preserved, and the optimized circuit remains secure in the first-order glitch-extended probing model. \square

C. An Illustrative Example

To make the propagation rules and security constraints concrete, we illustrate our method on a 16-input multiplier. The circuit is a combinational tree of 15 DOM-*indep* AND gates ($\&_t$), as shown in Fig. 3. Each gate introduces a fresh random bit r_t , and the only registers are those internal to each gate. This example serves to illustrate the methodology; complete case studies are presented in Section V.

1) *Labeling Internal Registers with Gate Randomness*: First, we determine the gate-randomness set, $\mathcal{R}(\cdot)$, for each gate's internal registers by applying the propagation and pruning rules from Section III-A. For the first-layer gates ($\&_1$ to $\&_8$), the

V. CASE STUDIES

We evaluate our methodology on circuits with contrasting structures: 32-bit adders, which achieve large reductions in randomness, and the AES S-box, where the gains are more modest. This contrast illustrates the methodology in designs with different structural properties.

A. 32-bit Adders

We begin with 32-bit adders, a fundamental building block in post-quantum cryptographic hardware, particularly for mask conversions such as Arithmetic-to-Boolean (A2B) and Boolean-to-Arithmetic (B2A) [23]. We use unprotected 32-bit netlists consisting solely of 2-input AND and XOR gates, obtained from a public artifact [24], [25], and consider four standard topologies: Brent–Kung [26], Sklansky [27], Kogge–Stone [28], and Ripple–Carry. Each AND gate is replaced with a first-order DOM-*indep* gadget, each XOR with a masked XOR gate, and pipeline registers are inserted as required. The resulting masked netlists are then processed with our three-step methodology. Table III summarizes the interference-graph properties. Although the vertex counts are large, reflecting circuit complexity, the graphs exhibit relatively small chromatic numbers and clique sizes, already suggesting significant optimization potential.

This observation is confirmed by the implementation results. Our framework reduces the demand for fresh randomness by 79%–90% compared to a naive assignment where each masked AND consumes a distinct random bit. As shown in Table IV, the parallel-prefix adders (Kogge–Stone, Sklansky, Brent–Kung) require up to 259 symbolic random bits ($|V|$) but only 24–27 physical random bits ($\chi(G)$) after coloring.

This high efficiency is a direct and predictable consequence of the inherent architectural properties of parallel-prefix adders. At each level of the prefix tree, the core logic merges *propagate* (P) and *generate* (G) signals from smaller bit-intervals using the fundamental recurrence $G_{\text{new}} \leftarrow G_{\text{upper}} + (P_{\text{upper}} \cdot G_{\text{lower}})$. This regular, hierarchical structure systematically limits the creation of conflicts in three key ways. First, the recurrence is implemented with constant low fan-in (one AND and one XOR gate), which strictly confines a probe’s glitch extension and keeps conflict sets small. Second, these computations run in parallel on disjoint bit-intervals, a separation that allows the same random bits to be safely reused across the adder’s width. Finally, the staged fan-out follows a strict hierarchy of bit intervals: an intermediate value such as G_{lower} from one interval feeds only the computations of G_{new} within its parent interval at the next level. This restriction prevents long-range connections that would otherwise create a dense interference graph. Collectively, these properties guarantee a sparse graph with a fundamentally low chromatic number, a characteristic clearly illustrated by the clustered patterns in Figs. 4a to 4c.

The ripple-carry adder has a strictly serial structure, where each carry depends only on the preceding one. This confines a probe’s glitch extension to a narrow scope, so at most three randomness symbols co-occur, yielding $\chi(G) = \omega(G) = 3$.

Table IV summarizes the implementation results for the four 32-bit adder topologies, reporting area, randomness consump-

TABLE III: Interference-graph properties for 32-bit adders. Reported are the number of vertices $|V|$, density $\rho(G)$, chromatic number $\chi(G)$, maximum clique size $\omega(G)$, average degree d_{avg} , and maximum degree d_{max} .

Design	$ V $	$\rho(G)$	$\chi(G)$	$\omega(G)$	d_{avg}	d_{max}
Brent–Kung	115	23%	24	19	26.5	69
Kogge–Stone	259	17%	27	26	44.3	72
Sklansky	161	21%	26	19	33.9	125
Ripple–Carry	31	13%	3	3	3.8	4

TABLE IV: Implementation metrics for 32-bit adders, before and after optimization.

Design	Area (kGE)	Randomness			Latency (cycles)
		Before ($ V $)	After ($\chi(G)$)	Reduction (%)	
Brent–Kung	9.99	115	24	79.1%	9
Kogge–Stone	14.25	259	27	89.6%	5
Sklansky	11.06	161	26	83.9%	6
Ripple–Carry	13.32	31	3	90.3%	31

tion, and latency. All designs were described in Verilog and synthesized with the NanGate 45 nm Open Cell Library using Synopsys DC Compiler v2021.06. Area, given in gate equivalents (GE), covers the complete implementation, including the masked datapath and the randomness-delivery pipeline. This pipeline adds about 5–11% overhead depending on the topology and is essential to prevent transient leakage: it forwards random bits through register stages that mirror the adder’s logic depth, ensuring the masks align with the data in every cycle. In the results, the total randomness equals the chromatic number $\chi(G)$, since each color maps to one physical random bit. Latency, measured in cycles, matches the prefix depth, with one level completed per cycle in the pipelined design. For all adder designs, the DSATUR algorithm completed in under 300 ms on a laptop with an Intel Core i7-8550U CPU (4 cores, 8 threads, 1.80 GHz) and 16 GB RAM, demonstrating that the method is efficient in practice. The Verilog source code, graph data (including conflict sets and color assignments), and security verification logs are available online.¹

Extending the analysis to compositional notions is left for future work. As a possible direction, a refresh stage could be added at the adder outputs, where each of the 32 bits is randomized and registered before use in subsequent logic.

B. AES S-box

To illustrate the methodology on a circuit with contrasting characteristics, we analyze the AES S-box. The design of optimized masked S-boxes is well studied in hardware masking, and our aim here is not to propose the most efficient variant but to provide a contrasting case to the adders in Section V-A. Whereas the adders produced sparse graphs and large randomness savings, the S-box is optimized for high diffusion, leading to a dense interference graph with more limited, yet still meaningful, opportunities for optimization.

¹github.com/KULeuven-COSIC/randomOPT

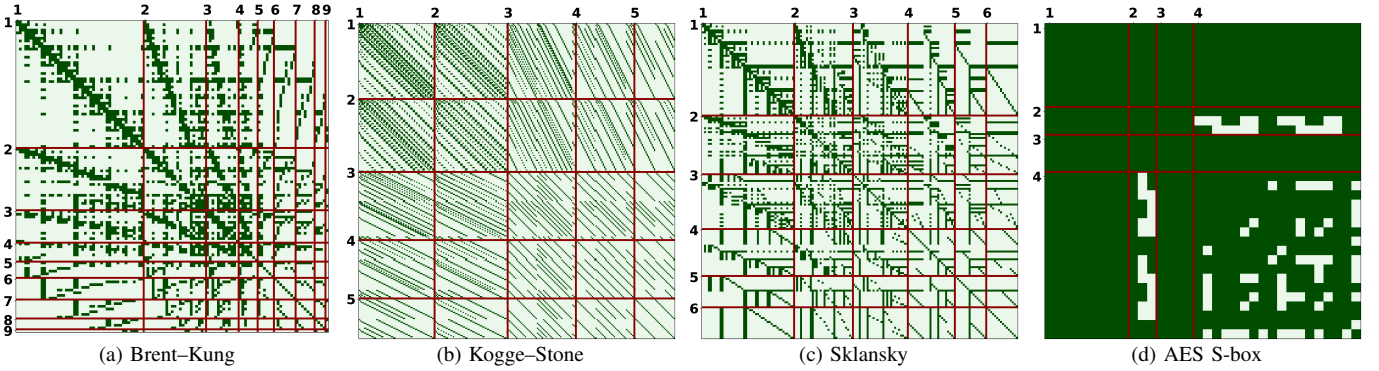


Fig. 4: Adjacency matrices of graphs $G = (V, E)$ for adders and AES S-box. The x - and y -axes list the vertices $v_i \in V$, ordered first by logic level (the prefix depth) and then by bit index; ticks indicate level boundaries. Entry (i, j) is 1 if $(v_i, v_j) \in E$. Pale green denotes no conflict (no edge), dark green denotes a conflict (edge). Level separators are in dark red.

We use the Boyar–Peralta AES S-box representation [15], which contains 34 two-input AND gates alongside XORs. Because several of these ANDs have dependent inputs, violating the assumption of the *DOM-indep* gadget, we replace them with the stronger but less efficient first-order HPC3 gadgets [20], which support dependent inputs. As shown in Eq. (1), $\bar{z} = \text{HPC3}(\bar{a}, \bar{b})$ preserves the crucial one-input randomness compression property, but each gadget consumes a pair of fresh random bits, $\{r', r''\}$, corresponding to a single vertex in our graph. This confirms that our methodology applies beyond DOM-based constructions and naturally extends to stronger gadgets such as HPC3, and the same principle, by design, holds for other gadgets of the HPC family [18], [29], [30].

$$\begin{aligned}
 z_0 &= \{a_0 b_0\} + \{a_0\} \cdot \{b_1 + r'\} + \{(-a_0)r' + r''\} \\
 &= a_0(b_0 + b_1) + (a_0 + \neg a_0)r' + r'' = a_0 \cdot b + r' + r'', \\
 z_1 &= \{a_1 b_1\} + \{a_1\} \cdot \{b_0 + r'\} + \{(-a_1)r' + r''\} \\
 &= a_1(b_1 + b_0) + (a_1 + \neg a_1)r' + r'' = a_1 \cdot b + r' + r''.
 \end{aligned} \tag{1}$$

The resulting graph is extremely dense ($\rho(G) \approx 91\%$), with an average vertex degree of 30.2 out of 33. This density stems from the structure of the S-box netlist, which features extensive fan-out and frequent recombination of intermediate values through XOR layers. In contrast to the localized dependencies of the adders, this design produces conflict sets that span multiple logic depths, resulting in the dense, high-clique graph shown in Fig. 4d. As a consequence, the maximum clique size equals the chromatic number ($\omega(G) = \chi(G) = 26$), which determines the randomness requirement. The optimized implementation consumes 52 fresh random bits ($2 \cdot \chi(G)$), reduced from an initial 68 bits ($2 \cdot |V|$), a meaningful reduction although the result guarantees only glitch-extended probing security rather than PINI composability [31]. The circuit occupies 4.34 kGE and has a latency of 4 cycles.

VI. SECURITY EVALUATION

We performed a simulation-based evaluation on all optimized designs in this work to test for first-order glitch-extended probing security. The evaluation covered the four 32-bit adder topologies in Sections V-A, the illustrative example in Section III-C, and the Boyar–Peralta AES S-box in Section V-B, all after applying the proposed randomness optimization. This

evaluation was performed using PROLEAD [16], a simulation-based leakage detection framework that tests statistical independence of internal nodes under the robust probing model. This tool was chosen for its scalability, as exhaustive formal verification is computationally infeasible for large circuits like the 32-bit adders. For each design, we generated the corresponding masked netlist and conducted fixed-versus-random G -tests in compact mode with default parameters over 100 million simulated traces. All designs passed without any detectable leakage, indicating compliance with the targeted security notion.

VII. CONCLUSION

We presented an automatable framework that reduces the randomness complexity of first-order masked hardware while preserving security in the glitch-extended probing model. The method leverages structural properties of masked AND gadgets, independent of their realization (e.g., DOM or HPC), and uses precise randomness-flow analysis to identify safe reuse opportunities. Modeling the resulting constraints as a graph-coloring problem turns the optimization into a well-understood combinatorial task solvable with standard heuristics.

Our case studies on 32-bit adders and the AES S-box show that the potential for randomness reduction is largely determined by circuit structure: regular adders yield sparse graphs and high savings, while diffusive designs such as the S-box form dense graphs with limited reuse.

Extending our methodology to higher orders appears non-trivial, as multiple probes enlarge conflict sets and complicate the coloring process. The current approach also conservatively unions all randomness within a probe’s glitch extension and models the constraints as cliques, which can be overly restrictive; identifying safe relaxations of this assumption is interesting future work. Furthermore, the security constraints are derived under the glitch-extended probing model; expanding the analysis to incorporate other established models (e.g., [32]) would further broaden applicability. Finally, this work has focused on gate-level gadgets (DOM and HPC) that exhibit randomness pruning. A challenging, open question is whether this graph-theoretic optimization can be adapted to other masking techniques, such as Threshold Implementations [6].

REFERENCES

- [1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.
- [2] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Smart Card Programming and Security*, I. Attali and T. Jensen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 200–210.
- [3] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 398–412.
- [4] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 463–481.
- [5] S. Faust, V. Grosso, S. Merino Del Pozo, C. Paglialonga, and F.-X. Standaert, "Composable masking schemes in the presence of physical defaults & the robust probing model," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, p. 89–120, Aug. 2018. [Online]. Available: <https://tches.iacr.org/index.php/TCHES/article/view/7270>
- [6] S. Nikova, C. Rechberger, and V. Rijmen, *Threshold Implementations Against Side-Channel Attacks and Glitches*. Springer Berlin Heidelberg, 2006, pp. 529–545. [Online]. Available: https://doi.org/10.1007/11935308_38
- [7] B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, and G. Stütz, "Threshold implementations of all 3 × 3 and 4 × 4 s-boxes," in *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, ser. Lecture Notes in Computer Science, E. Prouff and P. Schaumont, Eds., vol. 7428. Springer, 2012, pp. 76–91. [Online]. Available: https://doi.org/10.1007/978-3-642-33027-8_5
- [8] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking," in *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, ACM. ACM, oct 24 2016, pp. 3–3. [Online]. Available: <https://doi.org/10.1145/2996366.2996426>
- [9] A. Rezaei Shahmirzadi and A. Moradi, "Second-order sca security with almost no fresh randomness," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 708–755, jul 9 2021. [Online]. Available: <https://doi.org/10.46586/tches.v2021.i3.708-755>
- [10] —, "Re-consolidating first-order masking schemes," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 305–342, dec 3 2020. [Online]. Available: <https://doi.org/10.46586/tches.v2021.i1.305-342>
- [11] J. Daemen, "Changing of the guards: a simple and efficient method for achieving uniformity in threshold sharing," *Cryptology ePrint Archive*, Paper 2016/1061, 2016. [Online]. Available: <https://eprint.iacr.org/2016/1061>
- [12] D. Knichel and A. Moradi, "Composable gadgets with reused fresh masks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 114–140, jun 8 2022. [Online]. Available: <https://doi.org/10.46586/tches.v2022.i3.114-140>
- [13] J. Feldtkeller, D. Knichel, P. Sasdrich, A. Moradi, and T. Güneysu, "Randomness optimization for gadget compositions in higher-order masking," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 188–227, aug 31 2022. [Online]. Available: <https://doi.org/10.46586/tches.v2022.i4.188-227>
- [14] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990.
- [15] J. Boyar and R. Peralta, "A small depth-16 circuit for the aes s-box," in *Information Security and Privacy Research*, D. Gritzalis, S. Furnell, and M. Theoharidou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 287–298.
- [16] N. Müller and A. Moradi, "Prolead: A probing-based hardware leakage detection tool," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 4, p. 311–348, Aug. 2022. [Online]. Available: <https://tches.iacr.org/index.php/TCHES/article/view/9822>
- [17] L. De Meyer, O. Reparaz, and B. Bilgin, "Multiplicative masking for aes in hardware," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, p. 431–468, Aug. 2018. [Online]. Available: <https://tches.iacr.org/index.php/TCHES/article/view/7282>
- [18] G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert, "Hardware private circuits: From trivial composition to full verification," *IEEE Transactions on Computers*, vol. 70, no. 10, pp. 1677–1690, 2021.
- [19] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 463–481.
- [20] D. Knichel and A. Moradi, "Low-latency hardware private circuits," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ACM. ACM, nov 7 2022, pp. 1799–1812. [Online]. Available: <https://doi.org/10.1145/3548606.3559362>
- [21] R. Diestel, *Graph Theory*, 5th ed. Springer Publishing Company, Incorporated, 2017.
- [22] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, p. 251–256, Apr. 1979. [Online]. Available: <https://doi.org/10.1145/359094.359101>
- [23] L. Goubin, "A sound method for switching between boolean and arithmetic masking," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 3–15.
- [24] G. Cassiers, B. Gigerl, S. Mangard, C. Momin, and R. Nagpal, "Compress: Generate small and fast masked pipelined circuits," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 3, p. 500–529, Jul. 2024, artifact available at <https://artifacts.iacr.org/tches/2024/a23/>. [Online]. Available: <https://tches.iacr.org/index.php/TCHES/article/view/11686>
- [25] G. Cassiers and C. Momin, "Compress: Generate small and fast masked pipelined circuits," <https://github.com/cassiersg/compress>, Oct. 2024, source code repository, commit 01afd26.
- [26] R. P. Brent and H. T. Kung, "Fast algorithms for manipulating formal power series," *Journal of the ACM (JACM)*, vol. 25, no. 4, pp. 581–595, 1978.
- [27] J. Sklansky, "Conditional-sum addition logic," *IRE Transactions on Electronic Computers*, vol. EC-9, no. 2, pp. 226–231, 1960.
- [28] 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.
- [29] V. Hadžić and R. Bloem, "Efficient and composable masked aes s-box designs using optimized inverters," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2025, no. 1, p. 656–683, Dec. 2024. [Online]. Available: <https://tches.iacr.org/index.php/TCHES/article/view/11942>
- [30] G. Cassiers, F.-X. Standaert, and C. Verhamme, "Low-latency masked gadgets robust against physical defaults with application to ascon," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 3, p. 603–633, Jul. 2024. [Online]. Available: <https://tches.iacr.org/index.php/TCHES/article/view/11689>
- [31] 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.
- [32] N. Müller and A. Moradi, "Robust but relaxed probing model," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 4, pp. 451–482, Sep. 2024. [Online]. Available: <https://doi.org/10.46586/tches.v2024.i4.451-482>