Compositional Verification of Cryptographic Circuits against Fault Injection Attacks*

Huiyu Tan[1,2], Xi Yang[1], Fu Song[3,4] (✉), Taolue Chen[5], and Zhilin Wu[3]

[1] ShanghaiTech University, Shanghai 201210, China,
[2] Wingsemi Technology Co., Ltd., Shanghai 201203, China
[3] Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China, {songfu,wuzl}@ios.ac.cn
[4] Nanjing Institute of Software Technology, Nanjing 211135, China
[5] Birkbeck, University of London, WC1E 7HX, UK, t.chen@bbk.ac.uk

Abstract. Fault injection attack is a class of active, physical attacks against cryptographic circuits. The design and implementation of countermeasures against such attacks are intricate, error-prone and laborious, necessitating formal verification to guarantee their correctness. In this paper, we propose the first compositional verification approach for round-based hardware implementations of cryptographic algorithms. Our approach decomposes a circuit into a set of single-round sub-circuits which are verified individually by either SAT/SMT- or BDD-based tools. Our approach is implemented as an open-source tool CLEAVE, which is evaluated extensively on realistic cryptographic circuit benchmarks. The experimental results show that our approach is significantly more effective and efficient than the state-of-the-art.

1 Introduction

Cryptographic circuits are widely applied in various embedded and cyber-physical systems [5,39]. However, they are vulnerable to fault injection attacks, which disrupt the execution of cryptographic primitives via clock glitch [2], underpowering [34], voltage glitch [41], electromagnetic pulse [16], or laser beam [36]. With circuit's faulty outputs, attackers can employ statistical analysis methods to infer sensitive information, thereby threatening the security of, e.g., authentication. As a result, fault injection attacks pose a significant threat to the security of embedded and cyber-physical systems.

While countermeasures have been proposed to mitigate these attacks [1,26,35], their implementation does not necessarily guarantee security. Crucially, the fault-resistance of these countermeasures needs to be formally verified. While a plethora

* This work was funded by the Strategic Priority Research Program of CAS (XDA0320101), National Natural Science Foundation of China (62072309), CAS Project for Young Scientists in Basic Research (YSBR-040), ISCAS New Cultivation Project (ISCAS-PYFX-202201), ISCAS Fundamental Research Project (ISCAS-JCZD-202302), oversea grant from the State Key Laboratory of Novel Software Technology, Nanjing University (KFKT2023A04).

of fault-resistance analysis approaches have been proposed (cf. Section 6), the state-of-the-art formal verification approaches are non-compositional and limited in efficiency and scalability for realistic cryptographic circuits.

Contributions. In this work, we propose the first compositional verification approach for sequential circuits of cryptographic primitives with countermeasures against fault injection attacks, aiming to combat the efficiency and scalability challenges. Different from existing approaches for compositional safety and equivalence checking (e.g., [25,24,15]) which are not applicable for fault-resistance verification, our approach leverages the structural feature of round-based cryptographic circuits and decomposes the circuit into a set of single-round sub-circuits extended with, importantly, primary inputs/outputs, registers and their connections to guarantee soundness. We then verify those sub-circuits by leveraging SAT/SMT- and BDD-based approaches [37,31]. Our decomposition approach guarantees that the composition of fault-resistant single-round sub-circuits is always fault-resistant. Furthermore, we investigate various acceleration techniques that can significantly enhance verification efficiency.

We implement our approach as an open-source tool CLEAVE (**C**ompositional fau**L**t inj**E**ction **A**ttacks **VE**rifier), based on Verilog gate-level netlist. We thoroughly evaluate CLEAVE on 9 real-world cryptographic circuits (i.e., AES and LED64) equipped by both detection- and correction-based countermeasures, where the number of gates ranges from 1,020 to 34,351. The experimental results show that our approach is effective and efficient. For instance, the SAT-based compositional approach can verify most of the benchmarks (17/18) within 200 seconds and the remaining one can be done in 53 minutes; in contrast, the monolithic counterpart can only deal with 12 benchmarks within 6 hours and requires significantly more verification time. The same improvements can be observed for SMT- and BDD-based compositional approaches.

To summarize, we make the following contributions.

- We propose a novel compositional fault-resistance verification framework for cryptographic circuits and various techniques to enhance efficiency;
- We implement an open-source tool CLEAVE for Verilog gate-level netlists;
- We extensively evaluate our tool on realistic cryptographic circuits, demonstrating its effectiveness and efficiency.

Outline. Section 2 introduces preliminaries. Section 3 defines the fault-resistance verification problem. Section 4 presents our compositional verification approach. Section 5 reports experimental results; We discuss related work in Section 6 and conclude the work in Section 7. Benchmarks, the source code of CLEAVE, more experiential results and missing proofs are provided [38].

2 Preliminaries

Let $\mathbb{B} := \{0, 1\}$ and $[n] := \{1, \cdots, n\}$ for a natural number $n \geq 1$. We consider two types of logic gates: one-input gate $g : \mathbb{B} \to \mathbb{B}$ (e.g., `not`) and two-input gate $g : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ (e.g., `and`, `or`, `xor`). To model faulty gates, we define three faulty counterparts ($\overline{g}$, g_s, g_r) of each gate g with $\overline{g} = \neg g$, $g_s = 1$ and $g_r = 0$.

Definition 1. *A combinational circuit C is a tuple (V, I, O, E, g), where*

- *V is a finite set of vertices in the circuit such that each vertex $v \in V \setminus (I \cup O)$ is associated with a logic gate $\mathsf{g}(v)$ whose fan-in is the in-degree of v;*
- *$I \subseteq V$ and $O \subseteq V$ are the primary inputs and outputs, respectively;*
- *$E \subseteq (V \setminus O) \times (V \setminus I)$ is a set of edges, each of which $(v_1, v_2) \in E$ transmits the signal over $\mathbb{B}$ from v_1 to v_2, namely, one of the inputs of the logic gate $\mathsf{g}(v_2)$ is driven by the output of the logic gate $\mathsf{g}(v_1)$;*
- *and (V, E) forms a Directed Acyclic Graph (DAG).*

A combinational circuit C represents a Boolean function $[\![C]\!] : \mathbb{B}^{|I|} \to \mathbb{B}^{|O|}$ such that for any input signals $\boldsymbol{x} \in \mathbb{B}^{|I|}$, $[\![C]\!](\boldsymbol{x})$ is the output of the circuit C when fed with $\boldsymbol{x}$.

A (synchronous) sequential circuit is a combinational circuit with feedback via registers and synchronized by a global clock. It is memoryful as the registers store the internal state. In this paper, we focus on *round-based* circuit implementations of cryptographic algorithms. Conceptually, the circuit consists of several rounds, and physically each round may comprise some clock cycles. For our purpose, the sequential circuit is defined as follows.

Definition 2. *A k-clock cycle sequential circuit $\mathcal{S}[k]$ (we may simply write $\mathcal{S}$ to simplify the notation) is a tuple $(\mathcal{I}, \mathcal{O}, \mathcal{C}, \mathcal{R}, \boldsymbol{s}_0)$, where*

- *$\mathcal{I}$ and $\mathcal{O}$ comprise the primary inputs and primary outputs, respectively.*
- *$\mathcal{R} = \mathcal{R}_{in} \cup \mathcal{R}_s$ is a finite set of registers (aka memory gates), with initial signals $\boldsymbol{s}_0 \in \mathbb{B}^{|\mathcal{R}_s|}$ for state registers in $\mathcal{R}_s$. Intuitively, registers in $\mathcal{R}_{in}$ (resp. $\mathcal{R}_s$) store primary input signals (resp. results) of combinational circuits.*
- *$\mathcal{C} = \{C_1, \cdots, C_k\}$, where for each $i \in [k]$, $C_i = (V_i, I_i, O_i, E_i, \mathsf{g}_i)$ is a combinational circuit for the i-th clock cycle. Moreover, it is required that all the primary inputs $\mathcal{I}$ are connected to registers in $\mathcal{R}_{in}$ which in turn are connected to the inputs I_i to avoid glitches, and the outputs O_i are connected to the primary outputs $\mathcal{O}$ and registers in $\mathcal{R}_s$. We also extend function g_i such that $\mathsf{g}_i(r)$ is an identity function for every register $r \in \mathcal{R}$*

A *state* $\boldsymbol{s} : \mathcal{R}_s \to \mathbb{B}$ of $\mathcal{S}[k]$ is a valuation of the registers $\mathcal{R}_s$. In each clock cycle $i \in [k-1]$, given a state $\boldsymbol{s}_{i-1}$ and primary input signals $\boldsymbol{x}_i$, the next state $\boldsymbol{s}_i$ is $[\![C_i]\!](\boldsymbol{s}_{i-1}, \boldsymbol{x}_i)$ projected onto $\mathcal{R}_s$, while $[\![C_i]\!](\boldsymbol{s}_{i-1}, \boldsymbol{x}_i)$ projected onto $\mathcal{O}$ gives the primary output signals $\boldsymbol{y}_i$, written as $\boldsymbol{s}_{i-1} \xrightarrow{\boldsymbol{x}_i | \boldsymbol{y}_i} \boldsymbol{s}_i$.

Given a sequence of primary input signals $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$, a *run* ρ of the circuit $\mathcal{S}[k]$ is a sequence

$$\boldsymbol{s}_0 \xrightarrow{\boldsymbol{x}_1 | \boldsymbol{y}_1} \boldsymbol{s}_1 \xrightarrow{\boldsymbol{x}_2 | \boldsymbol{y}_2} \boldsymbol{s}_2 \xrightarrow{\boldsymbol{x}_3 | \boldsymbol{y}_3} \boldsymbol{s}_3 \longrightarrow \cdots \longrightarrow \boldsymbol{s}_{k-1} \xrightarrow{\boldsymbol{x}_k | \boldsymbol{y}_k} \boldsymbol{s}_k,$$

where $(\boldsymbol{y}_1, \cdots, \boldsymbol{y}_k)$ is the sequence of primary output signals. The circuit $\mathcal{S}[k]$ can also be seen as a Boolean function $[\![\mathcal{S}[k]]\!] : (\mathbb{B}^{|\mathcal{I}|})^k \to (\mathbb{B}^{|\mathcal{O}|})^k$ such that $[\![\mathcal{S}[k]]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ is the sequence of primary output signals for a sequence of primary input signals $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$.

We remark that our definition of sequential circuits is slightly different from the one given in [37], in which primary inputs can be connected to logic gates. We only allow primary inputs to connect to registers to avoid glitches which often introduce faults as well. Hence, our definition is sufficient for cryptographic circuits according to our experience while it facilitates the decomposition.

3 The Fault-Resistance Verification Problem

A fault injection attack actively injects faults into the execution of a cryptographic circuit and then infers sensitive data (such as the cryptographic key) via statistical analysis [3,9,8]. A general introduction refers to [21]. In particular, both non-invasive fault injections (i.e., clock glitches, underpowering and voltage glitches) and semi-invasive fault injections (i.e., electromagnetic pulses and laser beams) have been widely studied to compromise the security of cryptographic circuits, varying with attack cost and attack effectiveness [30]. There are detection- and correction-based countermeasures to mitigate fault injection attacks [1,35]: the former aims to detect fault injection attacks and raise an error flag once the attack is detected, so sensitive data can be destroyed in time; the latter aims to correct faults induced by attacks and produce the desired outputs.

3.1 Security Notions

We consider the following three fault types that suffice to capture both non-invasive fault injections and semi-invasive fault injections (cf. [30,37]):

- bit-set fault τ_s: when injected on a gate g, its output becomes $\mathtt{1}$, namely, the gate g becomes the faulty gate g_s, denoted by $\tau_s(g)$;
- bit-reset fault τ_r: when injected on a gate, its output becomes $\mathtt{0}$, namely, the gate g becomes the faulty gate g_r, denoted by $\tau_r(g)$;
- bit-flip fault τ_{bf}: when injected on a gate, its output is flipped, namely, the gate g becomes the faulty gate $\overline{g}$, denoted by $\tau_{bf}(g)$;

Fix a circuit $\mathcal{S}[k] = (\mathcal{I}, \mathcal{O}, \mathcal{R}, \boldsymbol{s}_0, \mathcal{C})$ protected using either a detection-based or correction-based countermeasure, where $\mathcal{C} = \{C_1, \cdots, C_k\}$ and for each $i \in [k]$, $C_i = (V_i, I_i, O_i, E_i, \mathsf{g}_i)$. We assume $o_{\mathtt{flag}} \in \mathcal{O}$, where $o_{\mathtt{flag}}$ is an error flag indicating whether a fault was detected when $\mathcal{S}$ adopts a detection-based countermeasure. If $\mathcal{S}$ adopts a correction-based countermeasure (i.e., no error flag is involved), we simply assume that $o_{\mathtt{flag}}$ is always $\mathtt{0}$. We denote by $\mathbf{B}$ the blacklist of invulnerable gates that are protected against fault injection attacks. $\mathbf{B}$ usually contains the gates used in implementing a countermeasure.

Definition 3. *A* fault vector *on the circuit $\mathcal{S}$ with the blacklist $\mathbf{B}$ and a set of fault types T, denoted by $\mathsf{V}(\mathcal{S}, \mathbf{B}, T)$, is a set of fault events*

$$\mathsf{V}(\mathcal{S}, \mathbf{B}, T) := \big\{\mathsf{e}(\alpha_1, \beta_1, \tau_1), \cdots, \mathsf{e}(\alpha_m, \beta_m, \tau_m) \mid i \neq j \implies (\sigma_i \neq \sigma_j \vee \beta_i \neq \beta_j)\big\},$$

where each fault event $\mathsf{e}(\sigma, \beta, \tau)$ consists of

- $\sigma \in [k]$ *specifying the clock cycle of the fault injection, namely, the fault injection occurs at the* σ*-th clock cycle;*
- $\beta \in \mathcal{R} \cup V_\sigma \setminus (I_\sigma \cup O_\sigma)$ *specifying the vulnerable gate on which the fault is injected (note that* $\beta \notin \mathbf{B}$*);*
- $\tau \in T$ *specifying the fault type.*

A fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}, T)$ yields a faulty circuit $\mathcal{F}(\mathcal{S}, \mathbf{B}, T) := (\mathcal{I}, \mathcal{O}, \mathcal{R}, \boldsymbol{s}_0, \mathcal{C}')$, where $\mathcal{C}' = \{C'_1, \cdots, C'_k\}$, for each $i \in [k]$: $C'_i := (V_i, I_i, O_i, E_i, \mathsf{g}'_i)$ and $\mathsf{g}'_i(\beta) := \tau(\mathsf{g}_i(\beta))$ if $\mathsf{e}(i, \beta, \tau) \in \mathsf{V}(\mathcal{S}, \mathbf{B}, T)$, otherwise $C'_i := C_i$ and $\mathsf{g}'_i(\beta) := \mathsf{g}_i(\beta)$.

Intuitively, the faulty circuit $\mathcal{F}(\mathcal{S}, \mathbf{B}, T)$ is the same as the circuit $\mathcal{S}$ except that for each fault event $\mathsf{e}(i, \beta, \tau) \in \mathsf{V}(\mathcal{S}, \mathbf{B}, T)$, the gate $\mathsf{g}_i(\beta)$ is transiently replaced by its faulty counterpart $\tau(\mathsf{g}_i(\beta))$ in the i-th clock cycle, whereas all the other gates remain the same.

Definition 4. *A fault vector* $\mathsf{V}(\mathcal{S}, \mathbf{B}, T)$ *is* effective *if there exists a sequence of primary input signals* $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ *such that two sequences of primary output signals*

$$[\![\mathcal{S}]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k) \text{ and } [\![\mathcal{F}(\mathcal{S}, \mathbf{B}, T)]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$$

differ at some clock cycle before the error flag $o_{\mathtt{flag}}$ *is set.*

Otherwise, the fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}, T)$ *is* ineffective *and the circuit* $\mathcal{S}$ *is* resistant *against the fault vector* $\mathsf{V}(\mathcal{S}, \mathbf{B}, T)$.

An effective fault vector results in faulty primary output signals where the fault is *not* successfully detected (i.e., the error flag $o_{\mathtt{flag}}$ is not set in time). Note that there are two possible cases for an ineffective fault vector: either $[\![\mathcal{S}]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ and $[\![\mathcal{F}(\mathcal{S}, \mathbf{B}, T)]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ are the same or the fault is successfully detected.

Inspired by the consolidated fault model [30], we define the security model for fault-resistance verification which characterizes the capabilities of the adversary.

Definition 5. *A* fault-resistance model *for the circuit* $\mathcal{S}$ *with the blacklist* $\mathbf{B}$ *is given by* $\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$*, where*

- $\mathtt{n}_e$ *is the maximum number of fault events per clock cycle;*
- $\mathtt{n}_c$ *is the maximum number of clock cycles in which fault events can occur;*
- $T \subseteq \{\tau_s, \tau_r, \tau_{bf}\}$ *specifies the set of allowed fault types; and*
- $\ell \in \{\mathtt{c}, \mathtt{r}, \mathtt{cr}\}$ *defines vulnerable gates:* `c` *for logic gates in combinational circuits,* `r` *for registers and* `cr` *for both logic gates and registers.*

For example, $\mathtt{m}(\mathtt{n}_e, k, \{\tau_s, \tau_r, \tau_{bf}\}, \mathtt{cr})$ models the strongest adversary, who can inject faults to all the gates simultaneously at any clock cycle (except for those protected in the blacklist $\mathbf{B}$) while $\mathtt{m}(1, 1, \{\tau_s\}, \mathtt{c})$ only allows the adversary to choose one logic gate to inject a set fault in one chosen clock cycle.

Formally, the fault-resistance model $\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$ defines the following set $[\![\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)]\!]$ of possible fault vectors that can be applied by the adversary:

$$[\![\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)]\!] := \left\{ \mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T) \;\middle|\; \begin{array}{c} \sharp\mathtt{MaxE}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) \leq \mathtt{n}_e \\ \text{and} \\ \sharp\mathtt{Clk}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) \leq \mathtt{n}_c \end{array} \right\}$$

where

- $\mathbf{B}_\ell := \begin{cases} \mathbf{B}, & \text{if } \ell = \mathtt{cr}; \\ \mathbf{B} \cup \mathcal{R}, & \text{if } \ell = \mathtt{c}; \\ \mathbf{B} \cup \bigcup_{i\in[k]} V_i \setminus (I_i \cup O_i), & \text{if } \ell = \mathtt{r}; \end{cases}$
- $\sharp\mathtt{MaxE}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) := \max_{\alpha\in[k]} |\{\mathtt{e}(\alpha, \beta, \tau) \in \mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)\}|$, i.e., the maximum number of fault events per clock cycle in the fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)$;
- $\sharp\mathtt{Clk}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) := |\{\alpha \mid \mathtt{e}(\alpha, \beta, \tau) \in \mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)\}|$, i.e., the number of clock cycles when fault events can occur.

Definition 6. *The circuit $\mathcal{S}$ is* fault-resistant *against $\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$, denoted by $\langle \mathcal{S}, \mathbf{B}\rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$, if all the fault vectors $\mathsf{V}(\mathcal{S}, \mathbf{B}, T) \in [\![\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)]\!]$ are ineffective.*

The fault-resistance verification problem *is to determine whether or not $\langle \mathcal{S}, \mathbf{B}\rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$.*

By Definition 6, it is straightforward to show that:

Proposition 1. *If $\langle \mathcal{S}, \mathbf{B}\rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T_1, \mathtt{cr})$, then $\langle \mathcal{S}, \mathbf{B}'\rangle \models \mathtt{m}(\mathtt{n}'_e, \mathtt{n}'_c, T_2, \ell)$ for any $\mathbf{B} \subseteq \mathbf{B}'$, $\mathtt{n}'_e \leq \mathtt{n}_e$, $\mathtt{n}'_c \leq \mathtt{n}_c$, $T_2 \subseteq T_1$, $\ell \in \{\mathtt{c}, \mathtt{r}, \mathtt{cr}\}$.*

By adapting the proof of NP-completeness [37] which reduces from the SAT problem, we can show

Theorem 1. *The problem of determining whether a k-clock cycle circuit $\mathcal{S}[k]$ for any fixed $k \geq 3$ is not fault-resistant is NP-complete.*

3.2 Motivating Example

A motivating example is given in Fig. 1, which is a simplified implementation of AES with a detection-based countermeasure [1]. The circuit has three cryptographic blocks (`B1`, `B2`, `B3`), three redundancy blocks (`RB1`, `RB2`, `RB3`), two selective blocks (`MUX1`, `MUX2`) and a check block `CHECK`, where all the gates in the check block `CHECK` are added to the blacklist $\mathbf{B}$. The cryptographic blocks and the two selective blocks together implement the functionality of AES, while the others implement a detection-based countermeasure.

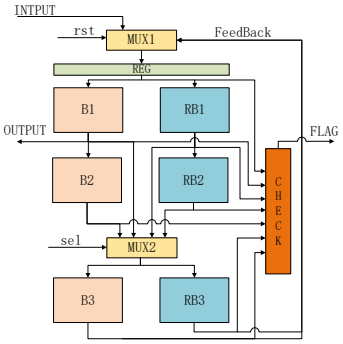

Fig. 1. The AES circuit.

The *first round* starts with a reset signal `rst` (i.e., `rst` =1) after which the primary input signals `INPUT` are selected by `MUX1` and stored in the registers `REG`. Moreover, `rst` is set to `0`. Next, the values stored in the registers `REG` are processed by the cryptographic and redundancy blocks. The cryptographic block `B1` produces primary output signals of the current round; the results of the cryptographic block `B3` and redundancy block `RB3` are stored in the registers `REG` as inputs of the next round (called feedback). Furthermore, the values of registers and the results of all the cryptographic and redundancy

blocks are fed to the check block `CHECK` which checks whether a fault injection attack occurs. The primary output `FLAG` is the error flag.

The *internal rounds* are the same as the first round except that the feedback from the previous round is stored in the registers, instead of the primary input signals, because the reset signal `rst` has been set to `0` in the first round. The *last round* is the same as the internal rounds except that the results of the cryptographic block `B1` (resp. the redundancy block `RB1`) are fed to the cryptographic block `B3` (resp. the redundancy block `RB3`) by setting the input signal `sel=1` of the selective block `MUX2`, respectively.

To verify its fault-resistance, one can unroll it according to the clock cycle (cf. [38]), then enumerate and check the effectiveness of each possible fault vector by analyzing the unrolled and faulty counterparts via BDD [31] or SAT/SMT [37]. However, there are two shortcomings which hurdle their efficiency and scalability. (1) One shall verify the equivalence of the primary outputs of the circuit and its faulty counterpart, which must be done for each round (unless the error flag is set). Since the subsequent round depends upon preceding rounds, the size of the SAT/SMT formulas or BDDs usually increases dramatically, which incurs a blowup in rounds of circuits. (2) To achieve completeness (or at least a high coverage), a large number of possible fault vectors have to be checked, which incurs a blowup in the number of fault vectors. Our work proposes a novel compositional approach to combat these two types of blowups in fault-resistance verification by decomposing the verification of an entire circuit into the verification of (typically much smaller) single-round sub-circuits.

4 Compositional Verification

In this section, we first describe the overview of our approach and our decomposition, next briefly recap two symbolic approaches (SAT/SMT- and BDD-based) for verifying sub-circuits, and finally present three acceleration techniques to improve the verification efficiency.

4.1 Overview of the Approach

Our approach relies on the structural feature of (round-based) cryptographic primitives, e.g., block ciphers, for which countermeasures are developed round-by-round accordingly, aiming to isolate the effects of fault injection in each round. Furthermore, the rounds are often similar, many of which are even the same, For instance, the first $(k-1)$ rounds in Fig. 1 are the same except that the first round uses the primary input signals while the other (internal) rounds use the feedback from the previous round (i.e., the values stored in the registers).

Based on the above key observation, as shown in Fig. 2, given a circuit $\mathcal{S}$, a blacklist $\mathbf{B}$ of gates on which faults cannot be injected and a fault-resistance model $\mathfrak{m}(n_e, n_c, T, \ell)$, our approach first decomposes the circuit $\mathcal{S}$ into single-round sub-circuits $(S_1, \cdots, S_r)$ where each S_i for $i \in [r]$ implements one round. As many sub-circuits are indeed identical, we only need to verify a small number

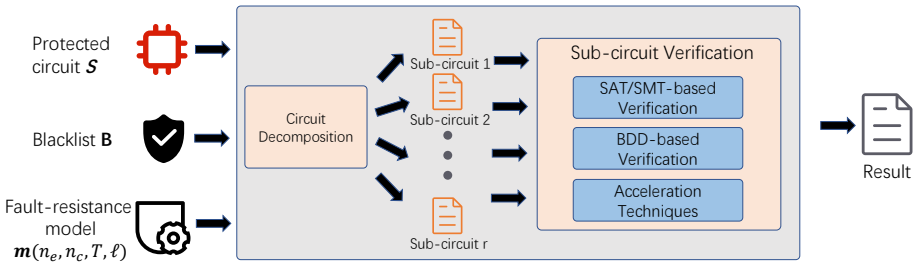

Fig. 2. Overview of our approach.

of single-round sub-circuits in isolation whereby the fault-resistance of the entire circuit $\mathcal{S}$ is guaranteed. For instance, in the motivating example, we only need to verify the first and the k-th (i.e., last) round, because the first $(k-1)$ rounds are virtually the same. It reduces the verification of a k-round circuit to the verification of two single-round sub-circuits.

To verify each sub-circuit, we leverage two symbolic verification approaches, based on SAT/SMT and BDD. To further improve efficiency, we also study various acceleration techniques exploiting fault effects and propagation.

4.2 The Decomposition

For a k-clock cycle circuit $\mathcal{S}[k] = (\mathcal{I}, \mathcal{O}, \mathcal{R}, \boldsymbol{s}_0, \mathcal{C})$ where $\mathcal{R} = \mathcal{R}_{in} \cup \mathcal{R}_s$, $\mathcal{C} = \{C_1, \cdots, C_k\}$ and $C_i = (V_i, I_i, O_i, E_i, \mathrm{g}_i)$ for each $i \in [k]$, let r be the number of rounds of $\mathcal{S}[k]$. An *r-decomposition* of $\mathcal{S}[k]$ is $(S_1[k_1], \cdots, S_r[k_r])$, where for every $i \in [r]$, $S_i[k_i]$ is a *single-round, k_i-clock cycle sub-circuit* $(\mathcal{I}^{(i)}, \mathcal{O}^{(i)}, \mathcal{R}^{(i)}, \boldsymbol{s}^{(i)}, \mathcal{C}^{(i)})$ defined as (note that $\sum_{i\in[r]} k_i = k$)

- $\mathcal{I}^{(i)} = \mathcal{I} \cup \mathcal{I}_{fb}$, where $\mathcal{I}_{fb}$ comprises additional primary inputs used for representing the signals passed from the previous round, i.e., the values stored in the state registers $\mathcal{R}_s$ at the end of the $(i-1)$-th round;
- $\mathcal{O}^{(i)} = \mathcal{O} \cup \mathcal{O}_{fb}$, where $\mathcal{O}_{fb}$ comprises additional primary outputs used for representing the signals passed to the next round, i.e., the values stored to the state registers $\mathcal{R}_s$ at the end of the $(i-1)$-th round;
- $\mathcal{R}^{(i)} = \mathcal{R}'_{in} \cup \mathcal{R}'_s$ where $\mathcal{R}'_{in} = \mathcal{R}_{in} \cup \mathcal{R}^{in}_s$, $\mathcal{R}^{in}_s \subseteq \mathcal{R}_s$ comprises registers used for storing signals passed from one round to the next round, and $\mathcal{R}'_s \subseteq \mathcal{R}_s$ comprises the registers used for connecting combinational circuits of $\mathcal{C}^{(i)}$ (note that $\mathcal{R}'_s$ can be $\emptyset$ if $k_i = 1$, i.e., the round has one clock cycle);
- $\boldsymbol{s}^{(1)} = \boldsymbol{s}_0$ and $\boldsymbol{s}^{(i)}$ for $i \geq 2$ is not defined;
- $\mathcal{C}^{(i)} = \{C_{i,1}, \cdots, C_{i,k_i}\}$ with $C_{1,1}, \cdots, C_{1,k_1}, \cdots, C_{r,1}, \cdots, C_{r,k_r} = C_1 \cdots C_k$, and the connection between any two adjacent single-rounds sub-circuits via the registers $\mathcal{R}'_s$ is the same as that in $\mathcal{S}$;
- the registers in $\mathcal{R}^{in}_s$ that were connected by the outputs $O_{i-1,k_{i-1}}$ of $C_{i-1,k_{i-1}}$ are now connected by the additional primary inputs $\mathcal{I}_{fb}$ if $i \geq 2$;
- the outputs $O_{i-1,k_{i-1}}$ of C_{i,k_i} that were connected to the registers in $\mathcal{R}_s$ are now connected to the additional primary outputs $\mathcal{O}_{fb}$.

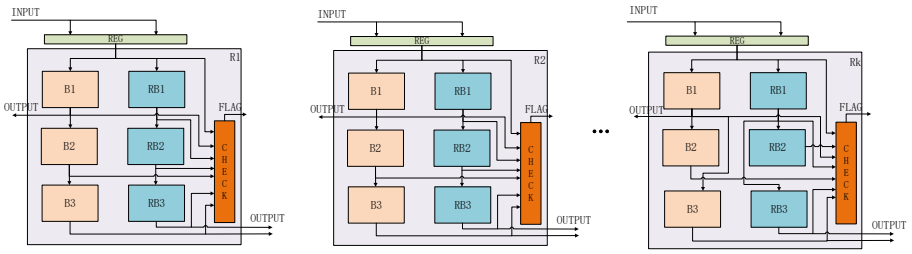

Fig. 3. Single-round sub-circuits of the motivating example.

Two single-round sub-circuits $S_i[k_i]$ and $S_j[k_j]$ are *isomorphic* w.r.t. the blacklist $\mathbf{B}$ if they are identical up to the renaming of the primary inputs/outputs, registers and vertices in the combinational circuits, and the matched gate pairs are either both protected or not protected in $\mathbf{B}$. Note that this condition is much stricter than the semantic equivalence of two circuits, namely, the same input-output relation, which is insufficient for our decomposition theorem. For instance, consider one single-round sub-circuit correctly implements a correction-based countermeasure but the other one does not implement any countermeasure. They are semantically equivalent, but both have to be verified.

Proposition 2. *For any pair of isomorphic circuits* (S_i, S_j) *and fault-resistance model* $\mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$, $\langle S_i, \mathbf{B}\rangle \models \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$ *iff* $\langle S_j, \mathbf{B}\rangle \models \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$. □

Consider the example in Fig. 1. In this case, $r = k$ ($k_i = 1$ for each $i \in [k]$). As illustrated in Fig. 3, our r-decomposition removes all the connections labeled with `FeedBack`, re-connects the outputs of the blocks `B3` and `RB3` to the additional primary outputs that were connected to the registers `REG`, and connects the additional primary inputs to the registers `REG` that were connected by the outputs from the previous round. Then, all the single-sound sub-circuits except for the last one are isomorphic.

Theorem 2. *Given a* k*-clock cycle circuit* $\mathcal{S}[k] = (\mathcal{I}, \mathcal{O}, \mathcal{R}, \boldsymbol{s}_0, \mathcal{C})$ *and a blacklist* $\mathbf{B}$*, let* $(S_1[k_1], S_2[k_2], \cdots, S_r[k_r])$ *be the* r*-decomposition of* $\mathcal{S}[k]$*. For any fault-resistance model* $\mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$*, if* $\langle S_i, \mathbf{B}\rangle \models \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$ *for all single-round sub-circuits* $S_i \in \{S_1, S_2 \cdots, S_r\}$*, then* $\langle \mathcal{S}, \mathbf{B}\rangle \models \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$.

Furthermore, if $\mathtt{n}_c \geq k_i$ *for all* $i \in [r]$*,then* $\langle \mathcal{S}, \mathbf{B}\rangle \models \mathfrak{m}(\mathtt{n}_e, k, T, \ell)$.

We should emphasize that the additional primary inputs $\mathcal{I}_{fb}$, primary outputs $\mathcal{O}_{fb}$, registers $\mathcal{R}_s^{in}$ and their connections are crucial to guarantee that the composition $\mathcal{S}[k]$ of the fault-resistant sing-round sub-circuits $(S_1[k_1], \cdots, S_r[k_r])$ is also fault-resistant. The fault-resistance of all the single-round sub-circuits, i.e., $\langle S_i, \mathbf{B}\rangle \models \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$ for $i \in [r]$, ensures that the primary outputs $\mathcal{O}' = \mathcal{O} \cup \mathcal{O}_{bf}$ remain the same (unless the error flag is set) for any fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}, T) \in \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$. It guarantees that not only the primary outputs $\mathcal{O}$ but also the values stored to the registers $\mathcal{R}_s^{in}$ at the end of each round remain the same (unless the error flag is set) for any fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}, T) \in \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$.

In other words, the single-round sub-circuits are able to detect any fault injections which change the primary outputs $\mathcal{O}$ or the values used by the next round (i.e., isolating fault effects in each round). Thus, our decomposition approach for compositional fault-resistance verification is different from previous ones used for compositional safety and equivalence checking (e.g., [25,24,15]).

4.3 SAT/SMT-based Verification

We adopt the SAT/SMT-based approach used in FIRMER [37] which reduces the problem to SAT/SMT solving. Given a fault-resistance model $\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$ and a (single-round) k-clock cycle circuit $\mathcal{S}[k] = (\mathcal{I}, \mathcal{O}, \mathcal{R}, \boldsymbol{s}_0, \mathcal{C})$, FIRMER first encodes all the possible fault vectors into $\mathcal{S}[k]$ by introducing additional inputs to control if a fault is injected on a gate and which fault type is injected. This will result in a controllable faulty circuit, denoted by $\mathcal{S}_{\mathtt{m}}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$. The fault-resistance verification of $\mathcal{S}[k]$ is reduced to equivalence checking of $\mathcal{S}$ and $\mathcal{S}_{\mathtt{m}}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$ with constraints on the additional inputs and error flag, which in turn is reduced to the SAT/SMT solving. (Cf. [37] for details.)

4.4 BDD-based Verification

We adopt the BDD-based approach used in FIVER [31]. To avoid re-construction of the BDD from scratch for each fault vector, FIVER first attaches each gate g in the circuit $\mathcal{S}$ with a BDD D_g representing the output of the gate in $\mathcal{S}$. Then, for each fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}, T) \in [\![\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)]\!]$, on a copy $\mathcal{S}'$ of the BDD-attached circuit $\mathcal{S}$, the BDD D_g of the gate g is revised according to each fault event $\mathtt{e}(i, g, \tau) \in \mathsf{V}(\mathcal{S}, \mathbf{B}, T)$, where the BDDs of the gates depending upon g are also revised accordingly. Finally, for each clock cycle, FIVER checks each primary output o by comparing the attached BDDs of the primary output o in the circuit $\mathcal{S}$ and its faulty counterpart $\mathcal{S}'$. Furthermore, some optimizations to reduce the number of considered fault vectors and improve the construction of the desired $\mathcal{S}'$ are implemented. (Cf. [31] for details.)

4.5 Acceleration Techniques

For both SAT/SMT-based and BDD-based verification, we apply the following acceleration techniques.

Fixed number of fault events. Recall that to prove fault-resistance, we considered all possible fault vectors $\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)$ such that $\sharp\mathtt{MaxE}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) \leq \mathtt{n}_e$ and $\sharp\mathtt{Clk}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) \leq \mathtt{n}_c$. It turns out that these two conditions can be safely improved to "$\mathtt{n}_e$ fault events for each clock cycle if some fault events occur in this clock cycle" when $\tau_s, \tau_r \in T$ and the number of vulnerable gates is more than $\mathtt{n}_e$ in each clock cycle, reducing the number of fault vectors to be checked. Indeed, if there is an effective fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}, T) \in [\![\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)]\!]$ such that the number of fault events is n in some clock cycle with $1 \leq n < \mathtt{n}_e$, there exists a sequence of primary input signals $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ such that $[\![\mathcal{S}]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ and

$[\![\mathcal{F}(\mathcal{S}, \mathbf{B}, T)]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ differ at some clock cycle before the error flag is set. We can add $(\mathtt{n}_e - n)$ fault events $\mathtt{e}(i, g, \tau)$ to $\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)$, where the output of the gate g under the primary input signals $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ remains the same by choosing $\tau \in \{\tau_s, \tau_r\}$. The resulting fault vector is still effective.

Fault type reduction. Let $\mathcal{T} = \{\tau_s, \tau_r, \tau_{bf}\}$. We find that $\langle \mathcal{S}, \mathbf{B} \rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \mathcal{T}, \ell)$ iff $\langle \mathcal{S}, \mathbf{B} \rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \tau_{bf}, \ell)$ iff $\langle \mathcal{S}, \mathbf{B} \rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \{\tau_s, \tau_r\}, \ell)$, allowing us to consider only $\{\tau_s, \tau_r\}$ if $\{\tau_s, \tau_r\} \subseteq T$ and only τ_{bf} if $\tau_{bf} \in T$ for any set T of fault types. Consider an effective fault vector $\mathsf{V}(\mathcal{S}, \mathbf{B}, \mathcal{T}) \in [\![\mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \mathcal{T}, \ell)]\!]$ and a sequence of primary input signals $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ such that $[\![\mathcal{S}]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ and $[\![\mathcal{F}(\mathcal{S}, \mathbf{B}, \mathcal{T})]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ differ at some clock cycle before the error flag is set.

- For every fault event $\mathtt{e}(i, g, \tau_{bf}) \in \mathsf{V}(\mathcal{S}, \mathbf{B}, \mathcal{T})$, if the output of the gate g at the i-th clock cycle in $[\![\mathcal{F}(\mathcal{S}, \mathbf{B}, \mathcal{T})]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ is flipped from `1` to `0` (resp. from `0` to `1`), $\mathtt{e}(i, g, \tau_{bf})$ can be safely replaced by $\mathtt{e}(i, g, \tau_r)$ (resp. $\mathtt{e}(i, g, \tau_s)$). Thus, $\langle \mathcal{S}, \mathbf{B} \rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \{\tau_s, \tau_r\}, \ell)$ entails $\langle \mathcal{S}, \mathbf{B} \rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \mathcal{T}, \ell)$.
- For every fault event $\mathtt{e}(i, g, \tau) \in \mathsf{V}(\mathcal{S}, \mathbf{B}, \mathcal{T})$ such that $\tau \in \{\tau_s, \tau_r\}$, if the output of the gate g at the i-th clock cycle in $[\![\mathcal{F}(\mathcal{S}, \mathbf{B}, \mathcal{T})]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ is flipped by applying $\mathtt{e}(i, g, \tau)$, $\mathtt{e}(i, g, \tau)$ can be safely replaced by $\mathtt{e}(i, g, \tau_{bf})$; otherwise the output of the gate g at the i-th clock cycle in $[\![\mathcal{F}(\mathcal{S}, \mathbf{B}, \mathcal{T})]\!](\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k)$ remains the same by applying $\mathtt{e}(i, g, \tau)$, $\mathtt{e}(i, g, \tau)$ can be safely removed from $\mathsf{V}(\mathcal{S}, \mathbf{B}, \mathcal{T})$. Thus, $\langle \mathcal{S}, \mathbf{B} \rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \tau_{bf}, \ell)$ entails $\langle \mathcal{S}, \mathbf{B} \rangle \models \mathtt{m}(\mathtt{n}_e, \mathtt{n}_c, \mathcal{T}, \ell)$.

Vulnerable gate reduction. If the output of a gate g is *only* connected to one vulnerable logic gate $g' \notin \mathbf{B}_\ell$, then the gate g can be safely added into the blacklist $\mathbf{B}$ while no protection is required for the gate g. It is because:

- if the output of the gate g does not change at the i-th clock cycle after applying the fault event $\mathtt{e}(i, g, \tau)$, then the effect of the fault event $\mathtt{e}(i, g, \tau)$ terminates at the gate g', thus $\mathtt{e}(i, g, \tau)$ can be removed from any fault vector;
- if the output of the gate g does change at the i-th clock cycle after applying the fault event $\mathtt{e}(i, g, \tau)$, it is flipped either from `1` to `0` or from `0` to `1`, the same effect can be achieved by applying the fault event $\mathtt{e}(i, g', \tau_{bf})$, or the fault event $\mathtt{e}(i, g', \tau_s)$ if it is flipped from `0` to `1` or the fault event $\mathtt{e}(i, g', \tau_r)$ if it is flipped from `1` to `0`.

As a result, it suffices to consider fault injections on the gate g' instead of both g and g' when $\tau_{bf} \in T$ or $\{\tau_s, \tau_r\} \subseteq T$, which reduces the number of vulnerable gates [37]. By a graph traversal of the circuit $\mathcal{S}$, all the gates g whose output is *only* connected to one vulnerable logic gate $g' \notin \mathbf{B}_\ell$ can be identified and then added into the blacklist $\mathbf{B}$.

We finally remark that the above three acceleration techniques can be applied simultaneously except that we cannot fix the number of fault events if the set and reset fault types (i.e., τ_s and τ_r) are unavailable.

5 Implementation and Evaluation

We have implemented our approach as an open-source tool CLEAVE based on the parallel SAT solver Glucose 4.2.1 [6] and SMT solver bitwuzla 1.0-prerelease [28],

where the BDD-based compositional verification is implemented based on FIVER which uses the CUDD package. Given a circuit $\mathcal{S}$ in Verilog gate-level netlist, a blacklist $\mathbf{B}$ and a fault-resistance model $\mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$, CLEAVE determines whether $(\mathcal{S}, \mathbf{B}) \models \mathfrak{m}(\mathtt{n}_e, \mathtt{n}_c, T, \ell)$. Currently, CLEAVE directly extracts single-round sub-circuits from $\mathcal{S}$ by enumerating all the feasible combinations of input signals of selective blocks. One feasible combination gives one single-round sub-circuit on which fault resistance is verified. Though more than one isomorphic single-round sub-circuits may be verified, the computational-expensive (GI-complete) isomorphism checking of pairs of single-round sub-circuits is avoided. For instance, the two distinct single-round sub-circuits of the circuit in Fig. 1) are extracted by fixing the signals of `rst` and `sel` to $(\mathtt{1}, \mathtt{0})$, $(\mathtt{0}, \mathtt{0})$ and $(\mathtt{0}, \mathtt{1})$, respectively, where the first two pairs of signals give the same single-round sub-circuits after adding/re-connecting primary inputs/outputs and registers according to our decomposition.

Benchmarks. We use 9 VHDL implementations [1,35] of 3 cryptographic algorithms (i.e., CRAFT, LED and AES [31]). The VHDL implementations are transformed into Verilog gate-level netlists using the Synopsys design compiler (version O-2018.06-SP2). The blacklists are generated according to [1,35]. The statistics of the benchmarks are given in Table 1. The first column shows the name of the cryptographic algorithm, the maximal number of protected faulty bits per clock cycle (bi), the type of the adopted countermeasure (D for detection-based and C for correction-based). The second column shows the single-round sub-circuit and its number of times used in the implementation, e.g., the 10-round AES-b1-D has two single-round sub-circuits (S1, S2) and S1 is used in 9 rounds. The other columns respectively give the size of the blacklist $\mathbf{B}$, the numbers of primary inputs, primary outputs, gates and each specific gate.

We can observe that CRAFT benchmarks use both detection-based (D) and correction-based (C) countermeasures, many single-round sub-circuits are isomorphic in each implementation, the number of distinct single-round sub-circuits ranges from 1 to 3, and the number of gates in one single-round sub-circuit ranges from 1,020 to 34,351 so that the scalability of CLEAVE can be evaluated.

Setup. The experiments were conducted on a machine with Intel Xeon Gold 6342 2.80GHz CPU, 1T RAM, and Ubuntu 20.04.1. Each verification task is run with 6-hour timeout. All the SAT-based and BDD-based (compositional) verification approaches are run with eight threads while the SMT-based (compositional) verification approaches are run with a single thread, with their default parameters (There are no promising parallel SMT solvers for QF_BV). The verification time is given in seconds with the best one highlighted in **boldface**, column R reports the verification result, and column DR shows the desired verification result. Mark ✓ (resp. ✗) indicates that the circuit is fault-resistant (resp. not fault-resistant) w.r.t. the fault-resistance model.

5.1 Effectiveness of Acceleration Techniques

Recall that we present three acceleration techniques: fixed number of fault events (denoted by FE), fault type reduction (denoted by TR), and vulnerable gate re-

Table 1. Benchmark statistics.

Name	Rnd	#Clk	|B|	#in	#out	#gate	#and	#nand	#or	#nor	#xor	#xnor	#not	#reg
AES-b1-D	S1×9	1	432	256	129	25,008	576	9,446	560	9,705	828	852	2,897	144
	S2×1	1	432	256	129	24,192	544	9,018	624	9,381	816	992	2,673	144
AES-b2-D	S1×9	1	1,055	256	129	34,351	704	12,698	833	13,012	1,440	1,584	3,888	192
	S2×1	1	1,055	256	129	33,423	752	12426	849	12,308	1,392	1,808	3,696	192
CRAFT-b1-D	S1×32	2	240	128	65	1,020	48	202	48	149	212	232	49	80
CRAFT-b2-D	S1×32	2	575	128	65	1,715	65	255	48	271	188	680	96	112
CRAFT-b3-D	S1×32	2	767	128	65	2,111	64	346	65	292	224	896	96	128
CRAFT-b1-C	S1×32	2	2,304	128	64	3,172	0	864	48	656	428	760	304	112
CRAFT-b2-C	S1×32	2	19,568	128	64	20,884	320	7,904	352	6,592	1,484	2,056	2,000	176
LED64-b1-D	S1×1	1	239	128	65	1,632	16	346	32	53	416	608	81	80
	S2×8	1	240	128	65	1,636	16	346	32	53	420	604	85	80
	S3×23	1	240	128	65	1,480	16	346	32	53	352	544	57	80
LED64-b2-D	S1×1	1	575	128	65	2,575	17	479	64	111	512	1168	112	112
	S2×8	1	575	128	65	2,585	17	479	64	111	516	1164	122	112
	S3×23	1	575	128	65	2,333	17	479	64	111	448	1024	78	112

Table 2. SAT-based verification of single-round sub-circuits.

Name	Model	no-opt	GR	GR·FE	GR·TR$_{sr}$	GR·FE·TR$_{sr}$	GR·TR$_{bf}$	R	DR
AES-b1-D	$\mathfrak{m}(1,1,\mathcal{T},\mathtt{cr})$	2,486.33	255.06	219.26	197.15	214.07	**178.58**	✓	✓
AES-b1-D	$\mathfrak{m}(2,1,\mathcal{T},\mathtt{cr})$	2.62	0.81	0.72	**0.60**	0.63	0.65	✗	✗
AES-b2-D	$\mathfrak{m}(2,1,\mathcal{T},\mathtt{cr})$	timeout	2,409.43	2,272.56	2,224.11	2,412.66	**1,595.51**	✓	✓
AES-b2-D	$\mathfrak{m}(3,1,\mathcal{T},\mathtt{cr})$	4.68	1.34	**0.94**	0.99	1.07	1.43	✗	✗
CRAFT-b2-C	$\mathfrak{m}(2,1,\mathcal{T},\mathtt{cr})$	31.80	10.08	10.78	10.95	11.09	**9.40**	✓	✓
CRAFT-b2-C	$\mathfrak{m}(3,1,\mathcal{T},\mathtt{cr})$	0.32	**0.26**	0.35	0.33	0.32	0.30	✗	✗
CRAFT-b3-D	$\mathfrak{m}(3,1,\mathcal{T},\mathtt{cr})$	7.56	0.33	0.32	0.32	**0.31**	0.42	✓	✓
CRAFT-b3-D	$\mathfrak{m}(4,1,\mathcal{T},\mathtt{cr})$	0.08	**0.04**	**0.04**	**0.04**	0.05	0.05	✗	✗

duction (denoted by GR). We denote by "no-opt" the verification without any of these acceleration techniques, by TR$_{sr}$ and TR$_{bf}$ the fault type reduction that reduces to the fault types (τ_s, τ_r) and the fault type τ_{bf}, respectively. The acceleration techniques can be combined, e.g., GR·FE applies "vulnerable gate reduction" with "fixed number of fault events". Note that TR$_{bf}$ cannot be combined with a fixed number of fault events (i.e., no FE·TR$_{bf}$ or GR·FE·TR$_{bf}$). We evaluate all the acceleration techniques and their feasible combinations on the first single-round sub-circuits of AES-b1-D, AES-b2-D, CRAFT-b2-C, and CRAFT-b3-D.

The results of SAT-based verification are reported in Table 2. Overall, all three acceleration techniques and their combinations can improve the SAT-based verification approach (no-opt) for almost all the verification tasks, solving one timeout case and significantly reducing the verification time for the other cases. The combination GR·TR$_{bf}$ outperforms the others because encoding the bit-flip fault type needs fewer fault type selection inputs than that of set and reset fault types. Note that adding more acceleration techniques does not necessarily make an improvement, e.g., GR·TR$_{sr}$ vs. GR·FE·TR$_{sr}$ on AES-bi-D, because $\sharp\mathtt{MaxE}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) = \mathtt{n}_e$ and $\sharp\mathtt{Clk}(\mathsf{V}(\mathcal{S}, \mathbf{B}_\ell, T)) = \mathtt{n}_c$ are encoded

Table 3. Results of fault-resistance verification: compositional vs. monolithic.

Name	Model	Compositional			Monolithic			R	DR
		BDD	SAT	SMT	BDD	SAT	SMT		
AES-b1-D	$\mathtt{m}(1,1,\mathcal{T},\mathtt{cr})$	**173.06**	193.49	15,944.55	timeout	timeout	timeout	✓	✓
AES-b1-D	$\mathtt{m}(2,1,\mathcal{T},\mathtt{cr})$	409.31	**1.65**	5,735.58	timeout	timeout	timeout	✗	✗
AES-b2-D	$\mathtt{m}(2,1,\mathcal{T},\mathtt{cr})$	timeout	**3,175.90**	timeout	timeout	timeout	timeout	✓	✓
AES-b2-D	$\mathtt{m}(3,1,\mathcal{T},\mathtt{cr})$	timeout	**2.25**	timeout	timeout	timeout	timeout	✗	✗
CRAFT-b1-C	$\mathtt{m}(1,1,\mathcal{T},\mathtt{cr})$	**0.13**	0.31	2.07	timeout	10,587.20	timeout	✓	✓
CRAFT-b1-C	$\mathtt{m}(2,1,\mathcal{T},\mathtt{cr})$	0.24	0.05	**0.04**	timeout	510.55	timeout	✗	✗
CRAFT-b2-C	$\mathtt{m}(2,1,\mathcal{T},\mathtt{cr})$	**3.02**	10.04	99.47	timeout	timeout	timeout	✓	✓
CRAFT-b2-C	$\mathtt{m}(3,1,\mathcal{T},\mathtt{cr})$	4.26	**0.38**	1.73	timeout	timeout	timeout	✗	✗
CRAFT-b1-D	$\mathtt{m}(1,2,\mathcal{T},\mathtt{cr})$	0.86	**0.13**	0.56	timeout	144.46	1,000.45	✓	✓
CRAFT-b1-D	$\mathtt{m}(2,2,\mathcal{T},\mathtt{cr})$	37.32	0.03	**0.02**	timeout	12.69	1.26	✗	✗
CRAFT-b2-D	$\mathtt{m}(2,2,\mathcal{T},\mathtt{cr})$	3,188.87	**0.30**	1.91	timeout	137.70	9,943.49	✓	✓
CRAFT-b2-D	$\mathtt{m}(3,2,\mathcal{T},\mathtt{cr})$	3,295.12	**0.04**	**0.04**	timeout	40.53	1.87	✗	✗
CRAFT-b3-D	$\mathtt{m}(3,2,\mathcal{T},\mathtt{cr})$	timeout	**0.44**	11.33	timeout	203.83	9,551.44	✓	✓
CRAFT-b3-D	$\mathtt{m}(4,2,\mathcal{T},\mathtt{cr})$	timeout	**0.05**	**0.05**	timeout	52.42	2.28	✗	✗
LED64-b1-D	$\mathtt{m}(1,1,\mathcal{T},\mathtt{cr})$	**0.93**	1.60	31.90	timeout	5,082.29	timeout	✓	✓
LED64-b1-D	$\mathtt{m}(2,1,\mathcal{T},\mathtt{cr})$	0.96	**0.16**	0.93	timeout	1.04	timeout	✗	✗
LED64-b2-D	$\mathtt{m}(2,1,\mathcal{T},\mathtt{cr})$	6.41	**2.34**	81.85	timeout	4,293.95	timeout	✓	✓
LED64-b2-D	$\mathtt{m}(3,1,\mathcal{T},\mathtt{cr})$	44.55	**0.17**	1.88	timeout	1.60	timeout	✗	✗

as $\mathtt{n}_e \leq \sharp\mathtt{MaxE}(\mathsf{V}(\mathcal{S},\mathbf{B}_\ell,T)) \leq \mathtt{n}_e$ and $\mathtt{n}_c \leq \sharp\mathtt{Clk}(\mathsf{V}(\mathcal{S},\mathbf{B}_\ell,T)) \leq \mathtt{n}_c$ before bit-blasting. Remark that FIRMER [37] indeed is CLEAVE when only GR is enabled. Due to space limitations, the results of SMT- and BDD-based verification are reported elsewhere [38], from which the same conclusion can be drawn. Thus, hereafter, we adopt the combination of acceleration techniques GR·TR$_{bf}$.

5.2 Evaluation of Compositional Verification

To evaluate our compositional approach, we compare it with the monolithic one, both of which adopt the combination of acceleration techniques GR·TR$_{bf}$.

The results are reported in Table 3. Overall, our compositional reasoning is very effective, allowing CLEAVE to verify fault-resistance of almost all the benchmarks while their monolithic counterparts often run out of time. For instance, the monolithic BDD-based approach fails to verify all the benchmarks due to the huge number of BDD variables. Indeed, the maximal number of rounds that can be handled is 2 (cf. [38] for details).

In contrast, the compositional reasoning can verify all the benchmarks, except for AES-b2-D and CRAFT-b3-D where even the single-round sub-circuit cannot be verified by the BDD-based approach. For SAT/SMT-based verification, the compositional reasoning takes significantly less time than its monolithic counterpart. Note that the diverse performance between SAT/SMT- and BDD-based approaches is mainly because we use the parallel SAT solver Glucose (8 threads) versus sequential SMT solver bitwuzla, and there is a cost for building (several) BDDs.

6 Related Work

Equivalence and safety checking play an essential role in the design of circuits. Various SAT/SMT-based approaches (e.g., [22,7,11,10,12]) and BDD-based approaches (e.g., [29,17,14,13]) have been studied. They are orthogonal to our work and cannot be directly applied to check fault-resistance.

Due to the prevalence of fault injection attacks, there are studies for finding the effective fault vectors or checking the effectiveness of the fault vectors provided by users, e.g., [33,4,23]. However, it is virtually impossible to enumerate all the possible fault vectors and valid inputs in practice, thus these approaches are limited in efficiency and scalability. To mitigate these issues, the BDD-based approach, FIVER [31], was proposed which does not need to explicitly enumerate all the possible valid inputs [31], but still has to explicitly enumerate all the possible fault vectors. Very recently, the SAT/SMT-based approach, FIRMER [37], was proposed to implicitly encode all the possible fault vectors into SAT/SMT formulas, and thus no explicit enumeration is required for both possible fault vectors and valid inputs. However, they often fail to verify the entire circuit under all the possible fault vectors and valid inputs. Our compositional approach circumvents the verification of the entire circuit of a large size, and can significantly boost both SAT/SMT-based and BDD-based verification approaches with novel acceleration techniques.

Compositional reasoning is a powerful divide-and-conquer approach for addressing the state-explosion problem. Hence, various compositional reasoning techniques and methods have been investigated, e.g., [25,27,20,19], for safety, equivalence and side-channel security verification. Our compositional reasoning relies on the structural feature of (round-based) cryptographic circuits and the fault-resistance verification problem, thus is different from the prior ones.

Synthesis techniques have been proposed to repair flaws (e.g., [18,40,32]). However, they do not provide security guarantees (e.g., [40,32]) or are limited to one specific type of fault injection attacks (e.g., clock glitch in [18]) and thus may be still vulnerable to other fault injection attacks.

7 Conclusion

We have proposed the first compositional reasoning which decomposes the fault-resistance verification of a whole round-based cryptographic circuit into that of single-round sub-circuits. To efficiently verify single-round sub-circuits, we have proposed various acceleration techniques and studied both SAT/SMT-based and BDD-based approaches. We have implemented our approach in an open-source tool CLEAVE and extensively evaluated it on a set of realistic cryptographic circuits. The experimental results show that our compositional approach and acceleration techniques can significantly improve all the SAT/SMT-based and BDD-based verification approaches, outperforming the state-of-the-art baselines.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aghaie, A., Moradi, A., Rasoolzadeh, S., Shahmirzadi, A.R., Schellenberg, F., Schneider, T.: Impeccable circuits. IEEE Transactions on Computers **69**, 361–376 (2020)
2. Agoyan, M., Dutertre, J., Naccache, D., Robisson, B., Tria, A.: When clocks fail: On critical paths and clock faults. In: Proceedings of the 9th IFIP WG 8.8/11.2 International Conference (CARDIS). pp. 182–193 (2010)
3. Anderson, R.J., Kuhn, M.G.: Low cost attacks on tamper resistant devices. In: Christianson, B., Crispo, B., Lomas, T.M.A., Roe, M. (eds.) Proceedings of the 5th International Workshop on Security Protocols. vol. 1361, pp. 125–136 (1997). https://doi.org/10.1007/BFB0028165
4. Arribas, V., Wegener, F., Moradi, A., Nikova, S.: Cryptographic fault diagnosis using verfi. In: Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 229–240 (2020)
5. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Comput. Networks **54**(15), 2787–2805 (2010)
6. Audemard, G., Simon, L.: On the glucose SAT solver. International Journal on Artificial Intelligence Tools **27**(1), 1840001:1–1840001:25 (2018)
7. Azarbad, M.R., Alizadeh, B.: Scalable SMT-based equivalence checking of nested loop pipelining in behavioral synthesis. ACM Trans. Design Autom. Electr. Syst. **22**(2), 22:1–22:22 (2017)
8. Baksi, A.: Classical and Physical Security of Symmetric Key Cryptographic Algorithms. Springer Singapore (2022)
9. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. Proc. IEEE **94**(2), 370–382 (2006). https://doi.org/10.1109/JPROC.2005.862424
10. Biere, A., Cimatti, A., Clarke, E.M., Fujita, M., Zhu, Y.: Symbolic model checking using SAT procedures instead of BDDs. In: Proceedings of the 36th Conference on Design Automation (DAC). pp. 317–320 (1999)
11. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without BDDs. In: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS). pp. 193–207 (1999)
12. Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., Hanna, Z., Nadel, A., Palti, A., Sebastiani, R.: A lazy and layered SMT($\mathcal{BV}$) solver for hard industrial verification problems. In: Proceedings of the 19th International Conference on Computer Aided Verification (CAV). pp. 547–560 (2007)
13. Burch, J.R., Clarke, E.M., Long, D.E., McMillan, K.L., Dill, D.L.: Symbolic model checking for sequential circuit verification. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **13**(4), 401–424 (1994)
14. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: 10^20 states and beyond. In: Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS). pp. 428–439 (1990)
15. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.): Handbook of Model Checking. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8
16. Dehbaoui, A., Dutertre, J., Robisson, B., Tria, A.: Electromagnetic transient faults injection on a hardware and a software implementations of AES. In: Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC). pp. 7–15 (2012)

17. van Eijk, C.A.J.: Sequential equivalence checking without state space traversal. In: Proceedings of Design, Automation and Test in Europe (DATE). pp. 618–623 (1998)
18. Eldib, H., Wu, M., Wang, C.: Synthesis of fault-attack countermeasures for cryptographic circuits. In: Proceedings of the 28th International Conference on Computer Aided Verification (CAV). pp. 343–363 (2016)
19. Gao, P., Song, F., Chen, T.: Compositional verification of first-order masking countermeasures against power side-channel attacks. ACM Trans. Softw. Eng. Methodol. **33**(3), 79:1–79:38 (2024)
20. Gao, P., Zhang, Y., Song, F., Chen, T., Standaert, F.: Compositional verification of efficient masking countermeasures against side-channel attacks. Proc. ACM Program. Lang. **7**(OOPSLA2), 1817–1847 (2023). https://doi.org/10.1145/3622862
21. Joye, M., Tunstall, M. (eds.): Fault Analysis in Cryptography. Information Security and Cryptography, Springer (2012)
22. Kaiss, D., Skaba, M., Hanna, Z., Khasidashvili, Z.: Industrial strength SAT-based alignability algorithm for hardware equivalence verification. In: Proceedings of the 7th International Conference on Formal Methods in Computer-Aided Design. pp. 20–26 (2007)
23. Khanna, P., Rebeiro, C., Hazra, A.: Xfc: a framework for exploitable fault characterization in block ciphers. In: Proceedings of the 54th Annual Design Automation Conference (DAC). pp. 1–6 (2017)
24. Khasidashvili, Z., Skaba, M., Kaiss, D., Hanna, Z.: Theoretical framework for compositional sequential hardware equivalence verification in presence of design constraints. In: Proceedings of the International Conference on Computer-Aided Design. pp. 58–65 (2004)
25. Khasidashvili, Z., Skaba, M., Kaiss, D., Hanna, Z.: Post-reboot equivalence and compositional verification of hardware. In: Proceedings of the 6th International Conference on Formal Methods in Computer-Aided Design. pp. 11–18 (2006)
26. Malkin, T., Standaert, F., Yung, M.: A comparative cost/security analysis of fault attack countermeasures. In: Proceedings of the 3rd International Workshop on Fault Diagnosis and Tolerance in Cryptography. pp. 159–172 (2006)
27. McMillan, K.L.: A methodology for hardware verification using compositional model checking. Sci. Comput. Program. **37**(1-3), 279–309 (2000). https://doi.org/10.1016/S0167-6423(99)00030-1
28. Niemetz, A., Preiner, M.: Bitwuzla at the SMT-COMP 2020. CoRR **abs/2006.01621** (2020), https://arxiv.org/abs/2006.01621
29. Pixley, C.: A theory and implementation of sequential hardware equivalence. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **11**(12), 1469–1478 (1992)
30. Richter-Brockmann, J., Sasdrich, P., Güneysu, T.: Revisiting fault adversary models - hardware faults in theory and practice. IEEE Transactions on Computers **72**, 572–585 (2023)
31. Richter-Brockmann, J., Shahmirzadi, A.R., Sasdrich, P., Moradi, A., Güneysu, T.: Fiver - robust verification of countermeasures against fault injections. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**, 447–473 (2021)
32. Roy, I., Rebeiro, C., Hazra, A., Bhunia, S.: SAFARI: automatic synthesis of fault-attack resistant block cipher implementations. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **39**(4), 752–765 (2020)

33. Saha, S., Mukhopadhyay, D., Dasgupta, P.: Expfault: An automated framework for exploitable fault characterization in block ciphers. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(2), 242–276 (2018)
34. Selmane, N., Guilley, S., Danger, J.: Practical setup time violation attacks on AES. In: Proceedings of the 7th European Dependable Computing Conference (EDCC). pp. 91–96 (2008)
35. Shahmirzadi, A.R., Rasoolzadeh, S., Moradi, A.: Impeccable circuits ii. Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC) pp. 1–6 (2020)
36. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Proceedings of the 4th International Workshop Redwood Shores on Cryptographic Hardware and Embedded Systems (CHES). pp. 2–12 (2003)
37. Tan, H., Gao, P., Chen, T., Song, F., Wu, Z.: SAT-based formal fault-resistance verification of cryptographic circuits. CoRR **abs/2307.00561** (2023)
38. Tan, H., Gao, P., Chen, T., Song, F., Wu, Z.: CLEAVE (2024), `https://github.com/S3L-official/CLEAVE`
39. Tyagi, A.K., Sreenath, N.: Cyber physical systems: Analyses, challenges and possible solutions. Internet of Things and Cyber-Physical Systems **1**, 22–33 (2021)
40. Wang, H., Li, H., Rahman, F., Tehranipoor, M.M., Farahmandi, F.: SoFI: Security property-driven vulnerability assessments of ICs against fault-injection attacks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **41**(3), 452–465 (2021)
41. Zussa, L., Dutertre, J.M., Clediere, J., Tria, A.: Power supply glitch induced faults on fpga: An in-depth analysis of the injection mechanism. In: Proceedings of the IEEE 19th International On-Line Testing Symposium (IOLTS). pp. 110–115 (2013)