Kalpa Publications in Computing

Volume 7, 2018, Pages 50–67

PROOFS 2018. 7th International Workshop
on Security Proofs for Embedded Systems

A Non-Reversible Insertion Method for Hardware Trojans Based on Path Delay Faults

Akira Ito, Rei Ueno, Naofumi Homma, and Takafumi Aoki

Tohoku University, Sendai, Miyagi, Japan
{ito, ueno, homma}@riec.tohoku.ac.jp, aoki@ecei.tohoku.ac.jp

Abstract

This paper presents a non-reversible method for stealthily inserting hardware Trojan (HT) based on a path delay fault called Path Delay HT (PDHT). While PDHT is hardly detected by the conventional methods including Monte-Carlo tests, its practicality is still unclear because a rarely sensitized path used for PDHT is selected and exploited in a deterministic manner. Such deterministic method indicates that we can find possible PDHT-inserted paths by its reversed method. In addition, the conventional method uses a genetic algorithm to add extra delays onto the selected path for inducing a path delay fault, and therefore, we have a difficulty in evaluating the resistance/vulnerability of a circuit to PDHT. This paper first presents a new method for selecting sufficiently rare paths to insert PDHT at random. We then show that the detectability/stealthiness of PDHT is related to switching activity (i.e., glitch effect), and present a new systematic method for inducing a path delay fault instead of GA. We demonstrate through an experimental PDHT-insertion and a Monte-Carlo test that the PDHT inserted by our method is sufficiently undetectable in comparison with the conventional method.

1 Introduction

Threats from hardware Trojans (HTs), which behave as hardware-oriented backdoors introducing faults or leaking out secret information, have gained increasing attention in academia, industry, and government agencies. HTs can be inserted by many involved companies (e.g. fabless companies, design houses, IP vendors, and semiconductor foundries) throughout the production process. In particular, malicious parties can introduce an HT into cryptographic hardware in order to retrieve secret information from the chip users and/or their clients.

There are many previous works related to HT detection mechanisms and countermeasures. Conventional HTs usually modify the functionality of target circuits at the register transfer level (RTL), net-list level, layout-level, or dopant-level in order to obtain secret information directly, to induce a fault for differential fault analysis (DFA), or to disable/degrade an embedded (pseudo) random number generator [1, 2, 3, 4]. Because these approaches should necessarily change the physical characteristics and/or the structures of the circuit, many countermeasures have been reported to detect such HTs. For example, we can examine side-channel information (e.g., power consumption) or compare a scanning electronic microscope (SEM) image of a chip layout with the reference.

In recent years, however, a new type of HT, called path delay HT (PDHT), was proposed at CHES 2016 [5]. A PDHT is inserted to a multiplier used for public key cryptographic hardware (e.g., RSA and ECC) to retrieve a secret key using the bug attack [6]. Such PDHT is inserted only by increasing the delay of a rarely sensitized path called rare path (RP). The PDHT can be triggered by applying a specific input sensitizing the RP to the HT-inserted multiplier. The propagation delay along the RP is larger than the original clock period, and therefore a setup time violation occurs when the RP is sensitized, which results in a buggy output. Only one buggy output of the multiplier is enough for the bug attack to retrieve the secret key of ECC and RSA. A PDHT is inserted without modifying the circuit function (i.e., input-output relation), datapath, nor structure in contrast to the conventional HTs. Furthermore, according to [5], the increase of a logic gate delay can be done with minor change in appearance or physical characteristics (e.g., cell size and power consumption), even in comparison with a dopant-level HT. As a result, it is very difficult to detect the PDHT using the conventional detection methods such as reverse-engineering, visual inspection, side-channel profiling, and SEM scanning.

However, the generality and practicality of PDHT are still unclear because of some questions about the resistance (i.e., stealthiness/detectability) of PDHTs in multipliers. The PDHT-insertion consists of two phases: (a) RP selection and (b) delay addition. The RP selection phase finds an RP in a multiplier in accordance with two features of each gate: controllability and observability. The delay addition phase determines the amount of extra delay for each gate along the selected RP in order to minimize the probability of setup time violation (i.e., the possibility of PDHT detection) during chip test via the Monte Carlo method. Here, the RP selection method in [5] selects only one RP in a deterministic manner. This means that we can reversely find the RP using the same selection method. In addition, the delay addition is performed using a genetic algorithm (GA). This indicates that we encounter a difficulty in evaluating its stealthiness and effectiveness in an analytic manner because of the irreproducibility and large processing time of GA. In particular, a time-consuming Monte-Carlo-based method would be required to determine the fitness function in GA, which has a significant impact on the resulting stealthiness/detectability in the context of PDHD.

This paper presents the possibility of inserting PDHTs in a non-reversible manner and analyzing the PDHT stealthiness and effectiveness. We first formulate the sensitization probability of a path using switching activity (i.e., glitches), while the previous paper did not mention the relation between the sensitization probability and glitches. Thereafter, we define a new metric, which is called gate switching detectability, to derive the sensitization probability of a path according to path features including glitches effects. Our approach makes it possible to discuss how much a path is undetectable if a PDHT is inserted. In addition, we present a systematic delay addition method that offers comparable stealthiness to the conventional GA-based method. In an example of an experimentally PDHT-inserted multiplier, the proposed method achieves an equivalent detection probability against the Monte-Carlo test. The result shows that the PDHT inserted by the proposed method is hardly detected by its reversed method and can be a practical threat to public key cryptographic hardware.

2 Previous work

The PDHT exploits a path delay fault, while conventional HTs usually employ a stack-at-fault or a trigger-payload logic. Path delay fault is a kind of fault that is far less detectable than stack-at-fault because the number of paths in a circuit grows exponentially with circuit size [7]. The original idea of PDHT is to add extra delay to the most rarely sensitized path (Rare Path: RP) for causing path delay fault intentionally. Because the HT-inserted path is selected

to be hardly sensitized, only attacker who knows how to activate the path can obtain the faulty output available for the bug attack owing to the setup time violation. Therefore, reducing the sensitization probability of HT-inserted path is quite important. If the sensitization probability is high, the PDHT can be detected via a typical Monte-Carlo test in quality check after chip fabrication. In practice, it is not simple to reduce the detection probability of PDHT because a path delay fault can be detected even when the path is partially sensitized. This indicates that the extra delay for PDHT should be added to each gate along RP such that its influence on other paths is minimum. To address the purpose, the PDHT-insertion method in [5] consists of RP selection and delay addition phases to reduce the detection probability. We briefly describe the two phases below.

2.1 Rare path selection

This phase selects the most rarely sensitized path in a multiplier in accordance with two features of gates: controllability and observability. Controllability is a probability of the gate output (i.e., a wire) for "1" when a random vector is applied to the multiplier. Observability is a probability that the value of gate output can be observed from the primary output of the circuit. A low observability of a gate indicates that its output value is unlikely to propagate to the primary output. Therefore, from the viewpoint of detectability, it is better to insert PDHT to a path with low controllability and observability. Controllability and observability are the metrics focusing on the static state of a circuit, which indicates that all gate switchings are completed after applying an input vector. In other words, they are calculated without considering glitch effects, and can be calculated from a logical expression (or gate-level netlist) of the multiplier. However, it is infeasible to calculate the exact controllability of each gate for practical multipliers even with the state-of-the-art SAT solvers. Therefore, in practice, controllability and observability are asymptotically calculated using a Monte-Carlo test.

In the path selection presented in [5], we first find a gate with the lowest controllability. We then extend a path from the gate to a primary input wire and output wire according to controllability and observability. This method selects gates such that its output is the most uncontrollable and unobservable among candidates. Finally, the SAT solver is used for examining whether the path is logically sensitizable. Thus, the attacker can select a rarely sensitized path and determine the input vectors to sensitize it.

However, the above method selects only one path in a deterministic manner. This means that we can reversibly find the path for PDHT using the same method and eventually detect the PDHT, while the path is usually not detected with Monte-Carlo tests. This implies the impracticality of PDHT. To show that PDHT can be a practical threat to cryptographic hardware, we should show a possibility of a non-reversible method that can randomly insert PDHT with a low detectability. Note here that it is still not sufficient to randomly select an initial gate with a low controllability under a threshold though it increases RP candidates by only the number of candidates for initial gates. In other words, we should randomly select a path with a low sensitizability rather than initial gates. However, it should be noted that if we select a path in a completely random manner, we can detect PDHT by Monte-Carlo tests [8].

2.2 Delay addition

After an RP is selected, the RP delay (more precisely, the delays of gates along the RP) is increased in order to make the total RP delays larger than the clock period. Because the gates along the RP are also included in many other paths, it is important to increase the RP delay while influencing the other paths as little as possible. In [5], a genetic algorithm is used to

determine the increase of RP delay. GA is an optimization technique that (locally) minimizes a cost function called the fitness function. We need to give some constraints on gate delays and determine the fitness function in order to apply GA to inserting extra delays on the RP. In [5], the fitness function is given by Monte Carlo tests, and the constraints on delays for each gate is given by

$$t'_{RP} = \sum_{i=0}^{n} t'_{\zeta}, \tag{1}$$

$$t_{\zeta} + \upsilon_{\zeta} - c \leq t'_{\zeta} \leq t_{\zeta} + \upsilon_{\zeta} + c, \tag{2}$$

where t'_{RP} is the resulting RP delay, υ_{ζ} is a slack of the ζ^{th} gate, and c is a constant. The slack represents a possible range of delay, and the constant c localizes the solution of GA for efficient computation. In addition, t_{ζ} and t'_{ζ} are the delays of the original and resulting i^{th} gate on RP, respectively. The first equation indicates that the resulting RP delay should be equal to the target delay t'_{RP} which exceeds the original critical delay (i.e., clock period), while the second equation helps the GA to discover reasonable delays for each gate. The slack for each gate is determined according to the delay of the gate, data dependency, and the clock period [9].

There are several issues involved in using GA in the context of PDHT. One is that GA usually takes a huge computation time. To make the resolution of detectability sufficiently low, a large number of random vectors are required for the fitness function, which implies a significant amount of time to evaluate the fitness function. In addition, large numbers of generations and individuals (each of which requires an evaluation of the fitness function) are necessary to obtain a good solution in GA. Thus, delay addition for a PDHT is quite time-consuming, which means that it is quite difficult to analyze the detectability/stealthiness of PDHT with a statistical means. Furthermore, the usage of GA also has a problem contradicting the definition of PDHT. In evaluating the fitness function, we should apply a sufficient number of input vectors required to guarantee a low detectability. In other words, because we should detect PDHT to evaluate the fitness function, the number of input vectors should be never enough to ensure that the PDHT is significantly undetectable. This problem originates from the difficulty in evaluating the quality of a solution in GA in a quantitative manner. Thus, it is difficult to evaluate and compare the detectability/stealthiness of PDHT in many multipliers in an analytical and quantitative manner because of the usage of GA, which makes difficult to develop multipliers robust to PDHT. To develop countermeasures, a more systematic/analytical method for delay addition is desirable.

3 Gate Switching Detectability (GSD)

This section presents a new analytical parameter, called Gate Switching Detectability (GSD), for deriving PDHTs. We first explain that the sensitization probability of a path can be formulated by the switching activity of gates included in the path (i.e., glitch effect) while the conventional PDHT-insertion method does not consider it. We then approximate the sensitization probability of a path based on the switching activity of the included gates. We finally introduce GSD derived from the approximated sensitization probabilities of paths including the gate. GSD represents how large a gate switching has an influence on the path sensitization probability. GSD is used for the proposed PDHT-insertion method consisting of the non-reversible path selection and systematic delay addition presented in Sect 4.

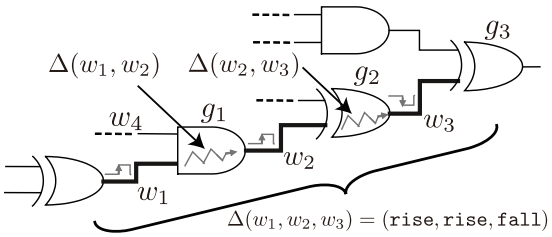

Figure 1: Circuit example.

3.1 Notations for path sensitization probability

This subsection introduces mathematical notations to represent the path sensitization probability in a formal manner. Figure 1 shows an example of circuit. We model a circuit as sets of Wires. Gates, and paths are then represented with wires. The notations for wire, gate, and path are as follows.

3.1.1 Wire

A circuit represented with wires $w_1, w_2, \ldots, w_u, \ldots, w_e$, where e is the number of wires. We call the signal transition of $0 \to 1$ and $1 \to 0$ on a wire `rise` and `fall`, respectively. Let Δw_u be the signal transition on wire w_u. The transitions of `rise` and `fall` on w_u are denoted by $\Delta w_u = \texttt{rise}$ and $\Delta w_u = \texttt{fall}$, respectively. The notations for wires and their signal transitions are listed below.

- $w_1, w_2, \ldots, w_u, \ldots, w_e$: a wire in the circuit, where e is the number of wires in the circuit.
- `rise`: the transition from 0 to 1.
- `fall`: the transition from 1 to 0.
- `null`: no transition (i.e., transitions from 0 to 0 and from 1 to 1).
- Δw_u: Transition of a signal on wire w_u. A rising and falling transition of w_u are denoted by $\Delta w_u = \texttt{rise}$ and $\Delta w_u = \texttt{fall}$, respectively. No transition of w_u is denoted by $\Delta w_u = \texttt{null}$
- $\mathcal{W}$: the set of all wires in the circuit.
- $\mathcal{W}_{\mathrm{PI}}$: the set of all wires connected directly from the primary input(s).
- $\mathcal{W}_{\mathrm{PO}}$: the set of all wires connected directly to the primary output(s).
- $\mathcal{S}_{\texttt{rfn}} = \{\texttt{rise}, \texttt{fall}, \texttt{null}\}$: the set of signal transitions.
- $\mathcal{S}_{\texttt{rf}} = \{\texttt{rise}, \texttt{fall}\}$: the set of signal switching.
- $s_u \in \mathcal{S}_{\texttt{rfn}}$: variable of signal transitions for Δw_u.
- $\sigma_u \in \mathcal{S}_{\texttt{rf}}$: variable of signal switchings for Δw_u

For example, in Fig. 1, w_1, w_2 and w_3 are wire, and $\Delta w_1 = \texttt{rise}$ indicates that the signal on w_1 is rising.

3.1.2 Gate

Gates $g_1, g_2, \ldots, g_i, \ldots, g_m$ are given as the pairs of the set of input wires and a output wire, where m is the number of gates.

- $g_i = (\mathcal{W}_{g_i}, w_i)$: a logic gate, where $\mathcal{W}_{g_i}$ denotes the set of input wires connected to g_i, and w_i denotes the output wire of g_i. For a d-input gate g_i, $\mathcal{W}_{g_i}$ consists of d wires $w_{i_1}, w_{i_2}, \ldots, w_{i_h}, \ldots, w_{i_d}$. Note that a gate g_i corresponds to the output wire w_i because we assume here that every logic gate has only one output, without loss of generality[1].
- $\mathcal{G}$: the set of all gates in the circuit.
- $\mathcal{W}_{\mathrm{PRE}}(w_i) = \{w_t \mid g_i = (\mathcal{W}_{g_i}, w_i) \in \mathcal{G}, w_t \in \mathcal{W}_{g_i}\}$: the set of wires given from the predecessor of w_i.
- $\mathcal{W}_{\mathrm{SUC}}(w_i) = \{w_f \mid g_f = (\mathcal{W}_{g_f}, w_f) \in \mathcal{G}, w_i \in \mathcal{W}_{g_f}\}$: set of wires of successor of w_i.

For example, in Fig. 1, g_1 is given as $(\mathcal{W}_{g_1}, w_2)$, where $\mathcal{W}_{g_1} = \{w_1, w_4\}$ and w_2 is an output wire.

3.1.3 Path

Paths are given as tuples of wires connected via gates from an input to an output. A series of signal transitions throughout a path is called *path sensitization (state)*. The notations for path and path sensitization are listed below.

- $p = (w_{v_1}, w_{v_2}, \ldots, w_{v_j}, \ldots, w_{v_l})$: a path through l wires $w_{v_1}, w_{v_2}, \ldots, w_{v_l}$, where v_j is an integer in the range of $1 \leq v_j \leq e$. In addition, w_{v_1} and w_{v_l} are the startpoint and endpoint wires, respectively.
- $|p|$: the length (i.e., the number of wires) of p. Note here that a wire w_{v_1} corresponds to a path with $|(w_{v_1})| = 1$.
- $\mathrm{Head}(p)$: the startpoint wire of p (i.e., w_{v_1}).
- $\mathrm{Tail}(p)$: the endpoint closest of p (i.e., w_{v_l}).
- $\mathcal{P}$: the set of all paths in the circuit.
- $\mathcal{P}_{\mathcal{W}_{\mathrm{H}},\mathcal{W}_{\mathrm{T}}} = \{p \mid p \in \mathcal{P}, \mathrm{Head}(p) \in \mathcal{W}_{\mathrm{H}}, \mathrm{Tail}(p) \in \mathcal{W}_{\mathrm{T}}\}$: the set of paths from $\mathrm{Head}(p) \in \mathcal{W}_{\mathrm{H}}$ to $\mathrm{Tail}(p) \in \mathcal{W}_{\mathrm{T}}$, where $\mathcal{W}_{\mathrm{H}}$ and $\mathcal{W}_{\mathrm{T}}$ are an arbitrary subset of $\mathcal{W}$.
- $\Delta p = \Delta(w_{v_1}, w_{v_2}, \ldots, w_{v_l}) = (\sigma_{v_1}, \sigma_{v_2}, \ldots, \sigma_{v_l})$: path sensitization state (i.e., propagation of signal transitions).
- $\Delta\mathcal{P}_{\mathrm{sub}} = \bigcup_{p_x \in \mathcal{P}_{\mathrm{sub}}} \{\Delta p_x = (\sigma_{x_1}, \sigma_{x_2}, \ldots, \sigma_{x_{|p_x|}}) \mid (\sigma_{x_1}, \sigma_{x_2}, \ldots, \sigma_{x_{|p_x|}}) \in \mathcal{S}_{\mathtt{rf}}^{|p_x|}\}$: the set of possible sensitization states of all paths in $\mathcal{P}_{\mathrm{sub}}$ excluding ones containing `null` wires, where $\mathcal{P}_{\mathrm{sub}}$ denotes a subset of $\mathcal{P}$. Note that elements of $\Delta\mathcal{P}_{\mathrm{sub}}$ are given in the form of $\Delta p_x = (\sigma_{x_1}, \sigma_{x_2}, \ldots, \sigma_{x_{|p_x|}})$.

[1] We can discuss multi-output gates (e.g., full adder cell) as plural gates with only one output. For example, assuming a gate with two output w_i and w_{i+1}. In this case, we consider this gate as two gates $g_i = (\mathcal{W}_{g_i}, w_i)$ and $g_{i+1} = (\mathcal{W}_{g_{i+1}}, w_{i+1})$ with $\mathcal{W}_{g_i} = \mathcal{W}_{g_{i+1}}$.

For example, in Fig. 1, $p_{\mathrm{ex}} = (w_1, w_2, w_3)$ is a path, and $\Delta p_{\mathrm{ex}} = \Delta(w_1, w_2, w_3) = (\texttt{rise}, \texttt{rise}, \texttt{fall})$ indicates the path sensitization state that the `rise` transition of the signal on w_1 causes the `rise` transition of the signal on w_2 and finally causes the `fall` transition of the signal on w_3.

3.2 Formula for path sensitization probability between inputs and output of each gate

We now focus on the sensitization probability of a path. Let $\Pr(A)$ be the probability of event (i.e., occuring transition(s)) A. From the above notation, given a path $p = (w_{v_1}, \ldots, w_{v_l})$, its transition is represented by $\Delta p = (\Delta w_{v_1}, \ldots, \Delta w_{v_l})$. Hence, the sensitization probability of p is represented by $\Pr(\Delta p = (\sigma_{v_1}, \ldots, \sigma_{v_l}))$. However, we cannot compute the sensitization probability because the relation between the switching probability of each wire and the sensitization probability of a path is unclear. Therefore, let us consider the simplest case where a signal switching of an input wire propagates to the output of a gate. Let $g_i = (\mathcal{W}_{g_i}, w_i)$ be a d-input gate, where $w_{i_1}, \ldots w_{i_h}, \ldots w_{i_d} \in \mathcal{W}_{\mathrm{PRE}}(w_i)$. Using the probability marginalization, the signal transition probability of w_i is given by

$$\Pr(\Delta w_i = \sigma_i) = \sum_{(s_{i_1}, \ldots, s_{i_d}) \in \mathcal{S}^d_{\mathrm{rfn}}} \Pr(\Delta w_i = \sigma_i, \Delta w_{i_1} = s_{i_1}, \ldots, \Delta w_{i_d} = s_{i_d}). \tag{3}$$

Here, we assume that two (or more) input signals never propagate to the gate output simultaneously, because it is physically impossible for two (or more) events to occur at completely at the same time in the real world[2]. In other words, at most only one input signal is either rising or falling at a moment. Hence, Eq. 3 is simplified as follows:

$$\begin{aligned}
\Pr(\Delta w_i = \sigma_i) = \Pr(&\Delta w_i = \sigma_i, \Delta w_{i_1} = \texttt{rise}, \Delta w_{i_2} = \texttt{null}, \ldots, \Delta w_{i_d} = \texttt{null}) \\
&+ \Pr(\Delta w_i = \sigma_i, \Delta w_{i_1} = \texttt{fall}, \Delta w_{i_2} = \texttt{null}, \ldots, \Delta w_{i_d} = \texttt{null}) \\
&+ \Pr(\Delta w_i = \sigma_i, \Delta w_{i_1} = \texttt{null}, \Delta w_{i_2} = \texttt{rise}, \ldots, \Delta w_{i_d} = \texttt{null}) \\
&+ \Pr(\Delta w_i = \sigma_i, \Delta w_{i_1} = \texttt{null}, \Delta w_{i_2} = \texttt{fall}, \ldots, \Delta w_{i_d} = \texttt{null}) \\
&\qquad\qquad \vdots \\
&+ \Pr(\Delta w_i = \sigma_i, \Delta w_{i_1} = \texttt{null}, \Delta w_{i_2} = \texttt{null}, \ldots, \Delta w_{i_d} = \texttt{rise}) \\
&\quad + \Pr(\Delta w_i = \sigma_i, \Delta w_{i_1} = \texttt{null}, \Delta w_{i_2} = \texttt{null}, \ldots, \Delta w_{i_d} = \texttt{fall}).
\end{aligned} \tag{4}$$

In Eq. 4, each term on the right side can be considered as the sensitization probability of a path consisting of an input and output wire of the gate. Therefore, we can confirm the relation:

$$\begin{aligned}
\Pr(\Delta w_i = \sigma_i, \Delta w_{i_h} = \sigma_{i_h}, \Delta w_{i_1} = \texttt{null}, \ldots, \Delta w_{i_{h-1}} = \texttt{null}, \\
\Delta w_{i_{h+1}} = \texttt{null}, \ldots, \Delta w_{i_d} = \texttt{null}) = \Pr(\Delta(w_{i_h}, w_i) = (\sigma_{i_h}, \sigma_i)).
\end{aligned} \tag{5}$$

Using Eq. 5, we can derive

$$\Pr(\Delta w_i = \sigma_i) = \sum_{w_{i_h} \in \mathcal{W}_{g_i}} \sum_{\sigma_{i_h} \in \mathcal{S}_{\mathrm{rf}}} \Pr(\Delta(w_{i_h}, w_i) = (\sigma_{i_h}, \sigma_i)). \tag{6}$$

[2] In fact, modern EDA tools for generating SAIF assume that signals of two or more input wires never propagete to the output simultaneously.

Eq. 6 indicates the relation between the switching probability of the gate output and the sensitization probability of each path from an input to the output of the gate.

Now let us consider the probability of a specific transition propagating through the gate represented by $\Pr(\Delta w_i = \sigma_i, \Delta w_{i_h} = \sigma_{i_h})$. Using a probability marginalization similar to the above, the sensitization probability $\Pr(\Delta w_i = \sigma_i, \Delta w_{i_h} = \sigma_{i_h})$ is transformed to

$$\begin{aligned}
&\Pr(\Delta w_i = \sigma_i, \Delta w_{i_h} = \sigma_{i_h}) \\
&= \sum_{(s_{i_1},\ldots,s_{i_{h-1}},s_{i_{h+1}},\ldots s_{i_d}) \in \mathcal{S}_{\mathrm{rfn}}^{d-1}} \Pr(\Delta w_i = \sigma_i, \Delta w_{i_h} = \sigma_{i_h}, \Delta w_{i_1} = s_{i_1}, \ldots, \Delta w_{i_{h-1}} = s_{i_{h-1}}, \\
&\qquad\qquad \Delta w_{i_{h+1}} = s_{i_{h+1}}, \ldots, \Delta w_{i_d} = s_{i_d}) \\
&= \Pr(\Delta w_i = \sigma_i, \Delta w_{i_h} = \sigma_{i_h}, \Delta w_{i_1} = \texttt{null}, \ldots, \Delta w_{i_{h-1}} = \texttt{null}, \\
&\qquad\qquad \Delta w_{i_{h+1}} = \texttt{null}, \ldots, \Delta w_{i_d} = \texttt{null}). \\
&= \Pr(\Delta(w_{i_h}, w_i) = (\sigma_{i_h}, \sigma_i)) \qquad (7)
\end{aligned}$$

Eq. 7 indicates the sensitization probability of a path from input to output of the gate is equal to the simultaneous probability of a signal switchings of input and output wires. In next subsection, we consider the sensitization probability of a path of which length is more than 2 using the above relation.

3.3 Calculation of path sensitization probability

This subsection formulates the sensitization probability of a path of which length is more than 2. The switching activity of each wire and each gate is derived from gate-level timing simulation because modern EDA tools (e.g., VCS and NcVerilog) can record transitions of wires and paths from an input to the output of each gate as switching activity information file (SAIF). For example, when the number of `rise` transitions of a wire $n_{\Delta w_i = \texttt{rise}}$ is recorded as the switching activity in SAIF, the `rise` transition probability of the wire is given as $\Delta t/T \times n_{\Delta w_i = \texttt{rise}}$, where Δt is the `rise` transition delay of the wire and T is simulation time that elapses in the timing simulation to obtain the switching activities. Since in this paper we assume that Δt of all wires are same, the transition probability of each wire has a factor of $\Delta t/T$. Therefore, GSD mentioned later must have the same factor, too. However, it doesn't matter whether the exact value of $\Delta t/T$ can be obtained because only the magnitude relationship between GSDs of gates plays an important role in the PDHT-insertion. Thus, when we calculate the transition probability of each wire, the $\Delta t/T$ can be practically ignored.

On the other hand, in the case of practical cryptographic hardware, we cannot calculate most of the path sensitization probability directly from the switching activities because the number of signal propagations though a path of which length is more than 2 cannot be recorded in SAIF. Therefore, in order to calculate path sensitization probabilities practically with a sufficient enough accuracy, we assume that a probability of path sensitization is recursively derived as the simultaneous probability of its partial path sensitization with length up to 2. For example, let us consider a path p_{ex} of three serially-connected gates with output wires w_1, w_2 and w_3 in Fig. 1. The path sensitization probability $\Pr(\Delta p_{\mathrm{ex}} = (\texttt{rise}, \texttt{rise}, \texttt{fall}))$ is given as the

conditional probability of path sensitization as follows:

$$\begin{aligned}\Pr(\Delta p_{\mathrm{ex}} = (\mathtt{rise},\mathtt{rise},\mathtt{fall})) &\approx \Pr(\Delta w_3 = \mathtt{fall}|\Delta w_2 = \mathtt{rise})\Pr(\Delta w_1 = \mathtt{rise}, \Delta w_2 = \mathtt{rise}) \\ &= \frac{\Pr(\Delta w_2 = \mathtt{rise}, \Delta w_3 = \mathtt{fall})}{\Pr(\Delta w_2 = \mathtt{rise})}\Pr(\Delta w_1 = \mathtt{rise}, \Delta w_2 = \mathtt{rise}).\end{aligned} \tag{8}$$

Note that $\Pr(A|B) = \Pr(A,B)/\Pr(B)$. Because $\Pr(\Delta w_2 = \mathtt{fall}, \Delta w_3 = \mathtt{rise})$ and $\Pr(\Delta w_1 = \mathtt{rise}, \Delta w_2 = \mathtt{rise})$ and $\Pr(\Delta w_2 = \mathtt{rise})$ are derived from SAIF, we can calculate the sensitization probability $\Pr(\Delta p_{\mathrm{ex}} = (\mathtt{rise},\mathtt{rise},\mathtt{fall}))$. Using Eq.7, we derive

$$\Pr(\Delta p_{\mathrm{ex}} = (\mathtt{rise},\mathtt{rise},\mathtt{fall})) \approx \frac{\Pr(\Delta(w_1,w_2) = (\mathtt{rise},\mathtt{rise}))\Pr(\Delta(w_2,w_3) = (\mathtt{rise},\mathtt{fall}))}{\Pr(\Delta w_2 = \mathtt{rise})}. \tag{9}$$

Eq. 9 indicates that the sensitization probability with length of more than 2 is approximated by the sensitization probabilities of paths with length of up to 2.

We then generalize Eq. 8 to arbitrary path $p = (w_{v_1}, \ldots, w_{v_l})$ with length of more than 3. Along with Eq. 8, the path sensitizability $\Pr(\Delta p = (\sigma_{v_1}, \ldots, \sigma_{v_l}))$ is decomposed to the products of conditional and simultaneous probabilities as follows:

$$\begin{aligned}\Pr(\Delta p = (\sigma_{v_1}, \ldots, \sigma_{v_l})) &\approx \Pr(\Delta w_{v_1} = \sigma_{v_1}) \prod_{j=2}^{|p|} \Pr(\Delta w_{v_j} = \sigma_{v_j}|\Delta w_{v_{j-1}} = \sigma_{v_{j-1}}) \\ &= \Pr(\Delta w_{v_1} = \sigma_{v_1}) \prod_{j=2}^{|p|} \frac{\Pr(\Delta w_{v_j} = \sigma_{v_j}, \Delta w_{v_{j-1}} = \sigma_{v_{j-1}})}{\Pr(\Delta w_{v_{j-1}} = \sigma_{v_{j-1}})}.\end{aligned} \tag{10}$$

Because the terms at the right hand side of Eq. 10 are given only by switching activity of each wire and each gate on path p (i.e., $\Pr(\Delta w_{v_{j-1}} = \sigma_{v_{j-1}})$ and $\Pr(\Delta w_{v_j} = \sigma_{v_j}, \Delta w_{v_{j-1}} = \sigma_{v_{j-1}})$) we can calculate a path sensitization probability from an SAIF according to Eq. 10. In addition, we can efficiently compute the sum of the sensitization probabilities of paths required for GSD using Eq. 10, as described in Sect. 3.4.

3.4 Formulation and efficient computation of GSD

Using the notation in Sect. 3.1, we define GSD (denoted by γ) as a sum of sensitization probabilities of all paths between a primary input and output through g_i, which represents the influence of g_i on PDHT detection.

Definition 1. *$\gamma_{g_i,w_{i_h}}(\sigma_{i_h},\sigma_i)$, which is the GSD of $g_i = (\mathcal{W}_{g_i}, w_i)$ on an input wire $w_{i_h} \in \mathcal{W}_{g_i}$ for σ_{i_h} and σ_i, is defined as*

$$\begin{aligned}&\gamma_{g_i,w_{i_h}}(\sigma_{i_h},\sigma_i) \\ &\quad= \sum_{\Delta p_y \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{PI}},\mathcal{P}_{\mathrm{PRE}}(w_{i_h})}} \sum_{\Delta p_z \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{SUC}}(w_i),\mathcal{W}_{\mathrm{PO}}}} \Pr(\Delta(w_{y_1}, \ldots, w_{y_{|p_y|}}, w_{i_h}, w_i, \\ &\qquad\qquad w_{z_1}, \ldots, w_{z_{|p_z|}}) = (\sigma_{y_1}, \ldots, \sigma_{y_{|p_y|}}, \sigma_{i_h}, \sigma_i, \sigma_{z_1}, \ldots, \sigma_{z_{|p_z|}})),\end{aligned} \tag{11}$$

where $p_y = (w_{y_1}, w_{y_2}, \ldots, w_{y_{|p_y|}})$ and $p_z = (w_{z_1}, w_{z_2}, \ldots, w_{z_{|p_z|}})$. Note here that w_{y_1} and $w_{z_{|p_z|}}$ should be a wire of respectively primary input and output, and $(w_{y_1}, \ldots w_{y_{|p_y|}}, w_{i_h}, w_i, w_{z_1}, \ldots, w_{z_{|p_z|}})$ is a path from a primary input to output.

Note also that a d-input gate has $4d$ GSDs on input wires $w_{i_0}, w_{i_1}, \ldots, w_{i_h}, \ldots, w_{i_d}$ for $\sigma_{i_h} \in \{\texttt{rise}, \texttt{fall}\}$ and $\sigma_i \in \{\texttt{rise}, \texttt{fall}\}$. This definition originates the fact that the modern EDA tools define gate switching as a propagation of signal transition from an input wire to the output wire.

In the following, according to Eq. 10, we transform the GSD in Eq. 11 into a form that can be efficiently computed. Using Eq. 10, we can rewrite the above GSD as follows:

$$\begin{aligned}\gamma_{g_i,w_{i_h}}(\sigma_{i_h},\sigma_i) = &\frac{\Pr(\Delta(w_{i_h},w_i)=(\sigma_{i_h},\sigma_i))}{\Pr(\Delta w_{i_h}=\sigma_{i_h})}\\ &\times \sum_{\Delta p_y \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{PI}},\mathcal{W}_{\mathrm{PRE}(w_{i_h})}}} \Pr(\Delta(w_{y_1},\ldots,w_{y_{|p_y|}},w_{i_h})=(\sigma_{y_1},\ldots,\sigma_{y_{|p_y|}},\sigma_{i_h}))\\ &\times \sum_{\Delta p_z \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{SUC}(w_i)},\mathcal{W}_{\mathrm{PO}}}} \frac{\Pr(\Delta(w_i,w_{z_1},\ldots,w_{z_{|p_z|}})=(\sigma_i,\sigma_{z_1},\ldots,\sigma_{z_{|p_z|}}))}{\Pr(\Delta w_i=\sigma_i)}. \quad (12)\end{aligned}$$

On the right hand side of Eq. 12, the term $\sum_{\Delta p_y}\Pr(\Delta(w_{y_1},\ldots,w_{y_{|p_y|}},w_{i_h})=(\sigma_{y_1},\ldots,\sigma_{y_{|p_y|}},\sigma_{i_h}))$ represents the sum of the sensitization probabilities of all paths from the primary input to w_{i_h}, and the term $\Pr(\Delta(w_i,w_{z_1},\ldots,w_{z_{|p_z|}})=(\sigma_i,\sigma_{z_1},\ldots,\sigma_{z_{|p_z|}}))$ represents that of paths from w_i to primary output. In other words, the former and later terms are the controllability and observability, considering dynamic hazard (i.e., glitch effect), respectively. We name them *dynamic controllability* on w_{i_h} for σ_{i_h} and *dynamic observability* on w_i for σ_i, which are denoted by $\mathrm{DyCon}_{w_{i_h}}(\sigma_{i_h})$ and $\mathrm{DyOb}_{w_i}(\sigma_i)$, respectively.

Using Eq. 12, we introduce Lemma 1 for efficient computation of dynamic controllability.

Lemma 1. *Given a wire w_{i_h}, the dynamic controllability is computed as the switching probability of signal on w_{i_h} as follows:*

$$\mathrm{DyCon}_{w_{i_h}}(\sigma_{i_h}) = \Pr(\Delta w_{i_h}=\sigma_{i_h}). \quad (13)$$

Similarly, for the dynamic controllability, we then introduce Lemma 2 below on the efficient computation of dynamic observability.

Lemma 2. *Given a wire w_i, $\mathrm{DyOb}_{w_i}(\sigma_i)$ can be computed from the dynamic observability of its successive wire $w_{i'}$ recursively as follows:*

$$\mathrm{DyOb}_{w_i}(\sigma_i) = \sum_{w_{i'}\in\mathcal{W}_{\mathrm{SUC}}(w_i)}\sum_{\sigma_{i'}\in\mathcal{S}_{\mathrm{rf}}} \frac{\Pr(\Delta w_{i'}=\sigma_{i'})\times\mathrm{DyOb}_{w_{i'}}(\sigma_{i'})}{\Pr(\Delta w_i=\sigma_i)}. \quad (14)$$

If $w_i \in \mathcal{W}_{\mathrm{PO}}$, then $\mathrm{DyOb}_{w_i}(\sigma_i)=1$.

Lemma 2 indicates that the dynamic observabilities can be computed iteratively from the primary output to w_i. See Appendices for the proofs of Lemmas 1 and 2.

Finally, GSD is represented by

$$\gamma_{g_i,w_{i_h}}(\sigma_{i_h},\sigma_i) = \Pr(\Delta(w_{i_h},w_i)=(\sigma_{i_h},\sigma_i))\times\mathrm{DyOb}_{w_i}(\sigma_i). \quad (15)$$

Because $\Pr(\Delta(w_{i_h},w_i)=(\sigma_{i_h},\sigma_i))$, and DyOb can be computed from only SAIF according to Lemmas 2, we can efficiently compute GSD based on Eq. 15. Note that DyCon is used for calculating γ while DyCon does not appear in Eq. 15.

Algorithm 1 Path Selection

Input: Circuit description $(\mathcal{W}, \mathcal{G}, \mathcal{P})$

Output: Rarely sensitized path p and its sensitization state $\Delta p = (\sigma_{v_1}, \ldots, \sigma_{v_l}) \in \mathcal{S}_{\mathtt{rf}}^{|p|}$

1: **parameter** α; ▷ Number of candidates (initial path seeds) for random selection
2: **function** SelectRP$(\mathcal{W}, \mathcal{G}, \mathcal{P})$
3: **set** $\Gamma = \{\}$;
4: **for all** $g_i = (\mathcal{W}_{g_i}, w_i) \in \mathcal{G}$ **do**
5: **for all** $w_{i_h} \in \mathcal{W}_{g_i}$ **do**
6: **for all** $(\sigma_i, \sigma_{i_h}) \in \mathcal{S}_{\mathtt{rf}}^2$ **do**
7: $\Gamma \leftarrow \Gamma \cup \{\gamma_{g_i, w_{i_h}}(\sigma_{i_h}, \sigma_i)\}$;
8: **end for**
9: **end for**
10: **end for**
11: **set** $\Gamma_\alpha \leftarrow \mathrm{Threshold}(\Gamma, \alpha)$; ▷ Extract α GSDs in order from the lowest one
12: $\gamma_{g_{\mathrm{ini}}, w_{i_{\mathrm{ini}}}}(\sigma_{i_{\mathrm{ini}}}, \sigma_{\mathrm{ini}}) \leftarrow \mathrm{Pick}(\Gamma_\alpha)$; ▷ Determine initial path randomly
13: **path** $p \leftarrow (w_{i_{\mathrm{ini}}}, w_{\mathrm{ini}})$; $\Delta p \leftarrow (\sigma_{i_{\mathrm{ini}}}, \sigma_{\mathrm{ini}})$;
14: **while** $\mathrm{Head}(p) \notin \mathcal{W}_{\mathrm{PI}}$ **do** ▷ Extend path backward to primary input
15: **set** $\Gamma_{\mathrm{cand}} \leftarrow \{\gamma_{g_c, w_{i_c}}(\sigma_{i_c}, \sigma_i) \mid w_c = \mathrm{Head}(p), w_{i_c} \in \mathcal{W}_{\mathrm{PRE}}(w_c), \sigma_{i_c} \in \mathcal{S}_{\mathtt{rf}}\}$;
16: $[p, \Delta p] \leftarrow$ ExtendPath$(p, \Delta p, \Gamma_{\mathrm{cand}})$;
17: **end while**
18: **while** $\mathrm{Tail}(p) \notin \mathcal{W}_{\mathrm{PO}}$ **do** ▷ Extend path forward to primary output
19: **set** $\Gamma_{\mathrm{cand}} \leftarrow \{\gamma_{g_c, w_{i_c}}(\sigma_{i_c}, \sigma_c) \mid w_{i_c} = \mathrm{Tail}(p), w_c \in \mathcal{W}_{\mathrm{SUC}}(w_{i_c}), \sigma_c \in \mathcal{S}_{\mathtt{rf}}\}$;
20: $[p, \Delta p] \leftarrow$ ExtendPath$(p, \Delta p, \Gamma_{\mathrm{cand}})$;
21: **end while**
22: **return** $[p, \Delta p]$;
23: **end function**

4 Proposed PDHT-insertion method

The proposed PDHT-insertion method consists of two phases: path selection and delay addition, similarly to the conventional method. Each phase of the proposed method has the following features for making PDHTs more practical and stealthy. The proposed path selection algorithm selects a path from a sufficient number of possible paths at random, which make it difficult to find the selected path by reversing the algorithm. The proposed delay addition imposes extra delay throughout the path according to the GSD values.

4.1 Path selection

Algorithm 1 shows an outline of the proposed path selection algorithm. The algorithm employs a parameter α to represent the number of candidate path seeds for a random selection. At Lines 3–10, we compute the GSDs (i.e., γs) of all gates in the given circuit. The computation of γs can be performed in a time proportional to the square of the number of gates according to Eq. 15 and Lemma 2. At Line 11, we obtain a set of a partial Γ consisting of α GSDs in increasing order. At Line 12, we randomly select a gate (more precisely, a pair of input and output wires on a gate) from the set. Here, a gate with the lowest γ would be the most suitable to PDHT-insertion. However, a gate is randomly selected from Γ_α in order to avoid the gate being found by reversing this algorithm. At Line 13, we set an initial path from the selected gate with length of 2. At Lines 14–17 and 18–21, we recursively extend the path to a primary input and a primary output, respectively. Here, the extension is performed by a function named "ExtendPath," which is shown in Algorithm 2 and explained in the following paragraph. At

Algorithm 2 Path Extension

Input: Path p, sensitization state Δp, Set of GSDs for extension direction Γ_{cand}
Output: Extended path p', extended sensitization state $\Delta p'$

```
 1: parameter ρ;
 2: function ExtendPath(p, Δp, Γ_cand)
 3:     float r ← rand(0, 1);
 4:     while Γ_cand ≠ {} do
 5:         if r ≤ ρ then
 6:             γ_{g_c,w_{i_c}}(σ_c, σ_{i_c}) ← MinGammaIn(Γ_cand);                ▷ Get lowest γ
 7:         else
 8:             γ_{g_c,w_{i_c}}(σ_c, σ_{i_c}) ← Pick(Γ_cand);                      ▷ Get γ randomly
 9:         end if
10:         wire w_{c'} ← w_c or w_{i_c}; σ_{c'} ← σ_c or σ_{i_c};   ▷ Determine a wire to primary input or output
11:         path p' ← ConcatPath(p, w_{c'}); Δp' ← ConcatPathSensiState(Δp, σ_{c'});
12:         if IsSensitizable(p', σ') = true then
13:             return [p', Δp'];
14:         else
15:             Γ_cand ← Γ_cand − {γ_{g_c,w_{i_c}}(σ_{i_c}, σ_c)};
16:         end if
17:     end while
18:     path p' ← RemoveGate(p); Δp' ← RemoveSensiState(Δp)   ▷ Remove the startpoint or endpoint gate
19:     return [p', Δp'].
20: end function
```

Line 15, we obtain a set of candidate GSDs (i.e., wires) for extending p to a primary input. Note that g_c and σ_c are determined in the previous loop (or Lines 12–13 initially). At Line 16, we extend the path by choosing a candidate from Γ_{cand} using the algorithm "ExtendPath." At Line 19, we obtain a set of candidate GSDs (i.e., wires) for extending p to a primary output. In contrast to Line 15, g_{i_c} and σ_{i_c} are determined in the previous loop (or Lines 12–13 initially). At Line 20, we extend the path by choosing a candidate according to "ExtendPath" as at Line 16. Finally, at Line 22, we return a rarely sensitized path from a primary input to output and its sensitization state.

Algorithm 2 shows a function for extending a path used at Lines 16 and 20 in Alg. 1. Algorithm 2 utilizes a parameter $\rho \in [0, 1]$ for randomness, in order not to detect the selected path reversely. At Line 3, we also obtain a random value $r \in [0, 1]$. At Lines 4–17, we determine an extension direction (i.e., a gate) for the path. Note here that we avoid simply selecting the gate with lowest γ because such PDHT is easily detected by its reverse algorithm. At Line 6, we select a candidate gate having the lowest γ if r is smaller than ρ; otherwise, at Line 8, we select a candidate gate randomly. Hence, if ρ is closer to 1, we extend the path mainly based on GSD. Conversely, if ρ is closer to 0, we extend the path mainly at random. Thereafter, at Line 11, we derive an extended path of p by concatenating p with $w_{c'}$ (i.e., w_c or w_{i_c}), and also derive an extended path sensitization state of Δp. At Line 13, we return the extended path and its sensitization state after checking its sensitizability by a SAT solver because p' may not be sensitizable. If not sensitizable, we repeat the above process after deleting the candidate at Line 15.

Thus, we can select a rarely sensitized path with a randomness induced by α and ρ. The entropy induced by α and ρ is evaluated in Sect. 5.

4.2 Delay addition

The proposed delay addition is also performed based on GSDs. This is because GSD represents an influence of a gate on a path sensitization probability, which corresponds to PDHT

detectability. Note that this delay addition can be performed in a deterministic manner in contrast to the above path selection. The delay increase of a gate g_j along the selected path p (denoted by $\tau(g_j)$) given by

$$\tau(g_j) = d\frac{\left(\log_2 |\gamma_{g_j,w_{i_j}}(\sigma_{i_j},\sigma_j)+\epsilon|\right)^{\lambda}}{\sum_{\eta=1}^{|p|}\left(\log_2 |\gamma_{g_\eta,w_{i_\eta}}(\sigma_{i_\eta},\sigma_\eta)+\epsilon|\right)^{\lambda}}, \tag{16}$$

where d is the total delay increase of the PDHT-inserted path and w_{i_j} is the input coming from the predecessor gate on the path, and ϵ is the very small value to avoid $\log_2(0)$. ϵ is needed because $\gamma_{g_j,w_{i_j}}(\sigma_{i_j},\sigma_j)$ becomes 0 when the transition probability of w_{i_j} is recorded as 0 in SAIF due to not enough number of applied vectors to estimate the switching activity. Because γ varies in the exponential range (e.g.,$10^{-12} \leq \gamma \leq 0.25$) empirically, we use the logarithm of γ for delay addition. In addition, because the optimal weight based on GSD depends on the structure of the multiplier, we introduce a natural number λ to optimize delay addition. The effect of λ is evaluated in the next section.

5 Experimental results

In this section, we evaluate the validity of the proposed PDHT-insertion method by an experimental insertion of PDHTs into a typical multiplier consisting of a partial product generator, a multiple-input adder, and a final stage adder. The target multiplier employs an AND gate array and a Wallace Tree as typical partial product generation and accumulator, respectively. On the other hand, two typical 2-input adders are used for the final stage adder: one is Ripple Carry Adder (RCA) which is the smallest and simplest, and another is Kogge Stone Adder (KSA) which is the fastest. The input bit length of the multiplier is set to 32×2 bits, which makes it difficult for functional verification and chip test in an exhaustive manner. The total delays of each selected paths varies from 1.8 to 1.2 times the critical path delay. In the evaluation, ρ and λ change while α is set to 5.

The detection rate of the inserted PDHTs is evaluated by the Monte Carlo test, which judges the case where the operation of the multiplier was not completed within the delay of the critical path as a fault when a random vector is input. For comparison, we also evaluate the detection rate of PDHTs given from the conventional method by the same experimental setting. The

Table 1: Detectability and path entropy of PDHTs inserted into multiplier based on RCA ($\times 10^{-4}$)

		$\rho=0$			$\rho=0.4$			$\rho=0.8$			Previous
		$\lambda=0$	$\lambda=1$	$\lambda=2$	$\lambda=0$	$\lambda=1$	$\lambda=2$	$\lambda=0$	$\lambda=1$	$\lambda=2$	work
$d=1.2$	mean	0.03	0.11	0.72	0.03	0.17	0.72	0.02	0.08	0.15	NA
	min.	0.01	0.01	0.25	0	0	0.04	0	0	0	0
$d=1.4$	mean	0.57	0.74	1.72	0.49	0.97	1.89	0.15	0.29	0.74	NA
	min.	0.15	0.05	0.86	0.01	0.11	0.3	0	0	0.04	0
$d=1.6$	mean	4.01	1.76	2.21	2.70	2.49	2.63	0.65	0.98	1.79	NA
	min.	0.67	0.28	0.85	0.25	0.63	0.57	0.05	0.07	0.15	0
$d=1.8$	mean	23.9	2.79	3.44	7.34	3.61	3.96	2.14	1.96	3.38	NA
	min.	2.79	1.46	1.34	0.41	1.2	1.21	0.34	0.2	0.45	0
Entropy		62.00			54.64			29.07			0

parameter c of the conventional method is 0.2 times the delay of the RP, or twice the minimum c that satisfies the constraints (1) and (2). The numbers of individuals and generations of GA are 10.

Table 1 and 2 show the detection rate of PDHTs and the entropy for the number of path candidates by the proposed method and the conventional method when RCA and KSA are used as the final stage adder, respectively. The detection rate was calculated by a Monte Carlo test with one million vectors, and it was set to 0 when detection was impossible. The first column of Tables 1 and 2 represents the magnification ratio of the added delay to the original one. Since the proposed method randomly selects a path for each trial, we performed 10 trials in the case of each condition and evaluated the detection rate of the 10 paths. Table 1 shows the lowest detection rate among the 10 paths as "min." and the logarithmic average of the detection rates of the 10 paths as "mean". When calculating the logarithm average, we assumed that the minimum detection rate is 10^{-7} in order to prevent the detection rate from being undefined.

One goal is that the detection rate required for PDHT is less than 10^{-6}. This is because the failure rate of an large scale integration (LSI) is assumed to be about 10^{-6} or so. If the PDHT detection rate is less than 10^{-6} or less, it becomes difficult to distinguish between LSI failure and PDHT. Therefore, it is desirable that the detection rate is less than 10^{-6}, that is "0", in Tables 1 and 2.

In the case of a multiplier based on KSA, we can confirm that the detection rate of the proposed method decreases clearly as the value of λ is larger. Also, the detection rate is lower as ρ is smaller. In the case of RCA, we can find that it becomes easy to detect PDHTs by increasing the value of λ. The major reason would be that glitch is more likely to occur structurally in RCA than KSA. On the other hand, the detection rate is lower as ρ is larger.

Comparing the proposed method with the conventional method, we can see that the proposed method is better or comparable for the cases of KSA and the conventional method is better for the cases of RCA. Note however that since the conventional method select only one path in a deterministic manner, the path can be easily detected by the same method as the insertion. On the other hand, the proposed method can prevent the inserted PDHT from being detected by the same method due to the random selection of the path.

In the proposed method, the entropy of the path candidate is determined by the values of α and ρ. The entropy in Tables 1 and 2 are calculated as follows. First, let the average number of gates included in the selected path be $\mathbb{E}[|p|] - 1$. Here, assuming 2 input gates, we select gates on the basis of GSD preferentially with probability ρ and randomly select with probability

Table 2: Detectability and path entropy of PDHTs inserted into multiplier based on KSA ($\times 10^{-4}$)

		$\rho = 0$			$\rho = 0.4$			$\rho = 0.8$			Previous
		$\lambda = 0$	$\lambda = 1$	$\lambda = 2$	$\lambda = 0$	$\lambda = 1$	$\lambda = 2$	$\lambda = 0$	$\lambda = 1$	$\lambda = 2$	work
$d = 1.2$	mean	1.05	0.03	0.002	0.08	0.004	0.004	2.33	0.04	0.006	NA
	min.	0	0	0	0	0	0	0	0	0	0
$d = 1.4$	mean	2.208	1.52	0.006	7.59	0.23	0.003	29.32	0.50	0.013	NA
	min.	0.39	0.01	0	0.51	0.04	0	3.2	0.03	0	0
$d = 1.6$	mean	102.7	11.84	0.06	50.91	2.17	0.01	122.2	4.24	0.013	NA
	min.	8.82	17.0	0	06.65	0.15	0	35.32	0.24	0	0
$d = 1.8$	mean	258.4	37.24	0.20	145.7	9.34	0.03	299.8	17.75	0.036	NA
	min.	48.12	6.16	0	28.85	0.44	0	129.7	1.14	0	1.45
Entropy		30.74			27.24			15.11			0

$1-\rho$. This is equivalent to selecting a gate of the lower GSD with probability $(1+\rho)/2$, and selecting another gate with probability $(1-\rho)/2$. In the path selection, this gate selection can be considered to repeat by the average number of gates. Therefore, the average entropy $\mathbb{H}(\rho)$ is given as

$$\mathbb{H}(\rho) = \frac{\mathbb{E}\left[|p|-1\right]}{2}\left((1+\rho)\log_2\frac{1+\rho}{2} + (1-\rho)\log_2\frac{1-\rho}{2}\right). \tag{17}$$

When ρ is 0, the entropy for the number of candidates is equal to the exponent part to represent the number of possible paths. In this experiment, the numbers of possible paths from a selected gate in KSA and RCA are about 2^{30} and 2^{60}, respectively. The values of $\mathbb{E}[|p|-1]$ are then estimated to be roughly 30 and 60 for KSA and RCA, respectively. In addition, if the number of candidate paths does not heavily change depending on the selected gate, we can simply add $\log_2(\alpha)$ to Eq. 17 for calculating the entropy which increases by the number of seeds for the path selection α. In fact, we confirmed that the number of candidate paths did not change much depending on experimentally selected gates. The results of Tables 1 and 2 were given by the above calculation. At a first glance, it seems that the entropy of the number of path candidates is small. However, for all these candidate paths, it is necessary to investigate a possible input to sensitize the path with a SAT solver. Since the time taken by a SAT solver was about from 0.5 seconds to several ten minutes experimentally, it is difficult to examine all the candidate paths with a realistic time if the entropy becomes about 20. Therefore, the PDHT inserted by the proposed method can be a threat when the detection rate by the Monte Carlo test is less than 10^{-6} and the entropy is about 20. We can say that the proposed method can be a sufficiently possible threat because the above conditions are satisfied by both KSA and RCA by adjusting the parameters.

6 Conclusion

This paper presented a non-reversible insertion method of PDHTs. First, we theoretically analyzed the relation between switching activity and path sensitization probability, and derive a new metric named Gate Switching Detectability (GSD) in order to evaluate the effect of the delay added to a logic gate on the whole circuit. Then, we proposed a new PDHT-insertion method consisting of a non-reversible path selection method and a delay addition method based on GSD. Unlike the conventional method based on the deterministic method, the proposed method has the feature that makes it difficult to detect PDHTs by tracing the insertion method. Furthermore, through the Monte Carlo test on multipliers with PDHT inserted by the proposed method and the conventional inserted, we confirmed that the proposed PDHT-insertion method could be a threat in practice.

In the future, a more detailed evaluation of PDHT detection rates would be required, and effective countermeasure/detection method for PDHTs should be studied.

Acknowledgment

This research has been supported by JSPS KAKENHI Grants No. 17H00729, No. 16K12436 and No. 16J05711.

References

[1] Matthew Hicks, Murph Finnicum, Samuel T King, Milo MK Martin, and Jonathan M Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 159–172. IEEE, 2010.

[2] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. *Advances in Cryptology—CRYPTO*, pages 513–525, 1997.

[3] Raghavan Kumar, Philipp Jovanovic, Wayne Burleson, and Ilia Polian. Parametric trojans for fault-injection attacks on cryptographic hardware. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 18–28. IEEE, 2014.

[4] Georg T Becker, Francesco Regazzoni, Christof Paar, and Wayne P Burleson. Stealthy dopant-level hardware trojans. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 197–214. Springer, 2013.

[5] Samaneh Ghandali, Georg T Becker, Daniel Holcomb, and Christof Paar. A design methodology for stealthy parametric trojans and its application to bug attacks. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 625–647. Springer, 2016.

[6] Eli Biham, Yaniv Carmeli, and Adi Shamir. Bug attacks. In *Annual International Cryptology Conference*, pages 221–240. Springer, 2008.

[7] Mohammad Reza Kakoee, Valeria Bertacco, and Luca Benini. At-speed distributed functional testing to detect logic and delay faults in nocs. *IEEE Transactions on Computers*, 63(3):703–717, 2014.

[8] Akira Ito, Rei Ueno, Naofumi Homma, and Takafumi Aoki. On the detectability of hardware trojans embedded in parallel multipliers. In *IEEE International Symposium on Multiple-Valued Logic (ISMVL), Linz, Austria*, 2018.

[9] Xiaoyong Tang, Hai Zhou, and Prith Banerjee. Leakage power optimization with dual-v th library in high-level synthesis. In *Proceedings of the 42nd annual Design Automation Conference*, pages 202–207. ACM, 2005.

A Proof of Lemma 1

Lemma 1. *Given a wire w_{i_h}, the dynamic controllability is computed as the switching probability of signal on w_{i_h} as follows:*

$$\mathrm{DyCon}_{w_{i_h}}(\sigma_{i_h}) = \Pr(\Delta w_{i_h} = \sigma_{i_h}). \tag{18}$$

Proof. Let $\xi(w_{i_h})$ be the longest path length among paths from the primary input to w_{i_h}, namely,

$$\xi(w_{i_h}) = \max_{p_y \in \mathcal{P}_{\mathcal{W}_{\mathrm{PI}}, \mathcal{W}_{\mathrm{PRE}}(w_{i_h})}} |p_y|. \tag{19}$$

Prove Lemma 1 using mathematical induction. Let $\mathrm{Q}(k)$ be the statement of Lemma 1 where $k = \xi(w_{i_h})$.
We observe that $\mathrm{Q}(2)$ is true from Eq. 6.
Assume that $\mathrm{Q}(n)$ is true for some $n \in \mathbb{N}$. It remains to show that $\mathrm{Q}(n+1)$ holds, then, we

have

$$\begin{aligned}
&\mathrm{DyCon}_{w_{i_h}}(\sigma_{i_h}) \\
&= \sum_{\Delta p_y \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{PI}},\mathcal{W}_{\mathrm{PRE}}(w_{i_h})}} \Pr(\Delta(w_{y_1},\ldots,w_{y_{|p_y|}},w_{i_h}) = (\sigma_{y_1},\ldots,\sigma_{y_{|p_y|}},\sigma_{i_h})) \\
&= \sum_{\Delta p_y \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{PI}},\mathcal{W}_{\mathrm{PRE}}(w_{i_h})}} \frac{\Pr(\Delta p_y = (\sigma_{y_1},\ldots,\sigma_{y_{|p_y|}}))}{\Pr(\Delta w_{y_{|p_y|}} = \sigma_{y_{|p_y|}})} \times \Pr(\Delta(w_{y_{|p_y|}},w_{i_h}) = (\sigma_{y_{|p_y|}},\sigma_{i_h})) \\
&= \sum_{w_{i'_{h'}} \in \mathcal{W}_{\mathrm{PRE}}(w_{i_h})} \sum_{\sigma_{i'_{h'}} \in \mathcal{S}_{\mathrm{rf}}} \left(\frac{\Pr(\Delta(w_{i'_{h'}},w_{i_h}) = (\sigma_{i'_{h'}},\sigma_{i_h}))}{\Pr(\Delta w_{i'_{h'}} = \sigma_{i'_{h'}})} \right. \\
&\qquad \left. \times \sum_{\Delta p_{y'} \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{PI}},\mathcal{W}_{\mathrm{PRE}}(w_{i'_{h'}})}} \Pr(\Delta(w_{y'_1},\ldots,w_{y'_{|p_{y'}|}},w_{i'_{h'}}) = (\sigma_{y'_1},\ldots,\sigma_{y'_{|p_{y'}|}},\sigma_{i'_{h'}})) \right).
\end{aligned} \tag{20}$$

Here, if $\xi(w_{i'_{h'}}) \leq n$, then $\mathrm{Q}(\chi)$ (i.e., $\sum_{\Delta p'_y} \Pr(\Delta(w_{y'_1},\ldots,w_{y'_{|p_{y'}|}},w_{i'_{h'}}) = (\sigma_{y'_1},\ldots,\sigma_{y'_{|p_{y'}|}},\sigma_{i'_{h'}}))$ $= \Pr(\Delta w_{i'_{h'}} = \sigma_{i'_{h'}})$) is true for $2 \leq \chi \leq n$. Because $p_{y'}$ should be shorter than p_y (i.e., $|p_{y'}| < n$), we have

$$\begin{aligned}
&\sum_{w_{i'_{h'}} \in \mathcal{W}_{\mathrm{PRE}}(w_{i_h})} \sum_{\sigma_{i'_{h'}} \in \mathcal{S}_{\mathrm{rf}}} \frac{\Pr(\Delta(w_{i'_{h'}},w_{i_h}) = (\sigma_{i'_{h'}},\sigma_{i_h}))}{\Pr(\Delta w_{i'_{h'}} = \sigma_{i'_{h'}})} \times \Pr(\Delta w_{i'_{h'}} = \sigma_{i'_{h'}}) \\
&= \sum_{w_{i'_{h'}} \in \mathcal{W}_{\mathrm{PRE}}(w_{i_h})} \sum_{\sigma_{i'_{h'}} \in \mathcal{S}_{\mathrm{rf}}} \Pr(\Delta(w_{i'_{h'}},w_{i_h}) = (\sigma_{i'_{h'}},\sigma_{i_h})) \\
&= \Pr(\Delta w_{i_h} = \sigma_{i_h}),
\end{aligned} \tag{21}$$

according to Eq. 6. □

B Proof of Lemma 2

Lemma 2. *Given a wire w_i, $\mathrm{DyOb}_{w_i}(\sigma_i)$ can be computed from the dynamic observability of its successive wire $w_{i'}$ recursively as follows:*

$$\mathrm{DyOb}_{w_i}(\sigma_i) = \sum_{w_{i'} \in \mathcal{W}_{\mathrm{SUC}}(w_i)} \sum_{\sigma_{i'} \in \mathcal{S}_{\mathrm{rf}}} \frac{\Pr(\Delta w_{i'} = \sigma_{i'}) \times \mathrm{DyOb}_{w_{i'}}(\sigma_{i'})}{\Pr(\Delta w_i = \sigma_i)}. \tag{22}$$

If $w_i \in \mathcal{W}_{\mathrm{PO}}$, then $\mathrm{DyOb}_{w_i}(\sigma_i) = 1$.

Proof. If $w_i \in \mathcal{W}_{\mathrm{PO}}$, it is trivial from definition of the dynamic observability. Otherwise, we

have

$$
\begin{aligned}
&\mathrm{DyOb}_{w_i}(\sigma_i) \\
&= \sum_{\Delta p_z \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{SUC}}(w_i),\mathcal{W}_{\mathrm{PO}}}} \frac{\Pr(\Delta(w_i, w_{z_1}, \ldots, w_{z_{|p_z|}}) = (\sigma_i, \sigma_{z_1}, \ldots, \sigma_{z_{|p_z|}}))}{\Pr(\Delta w_i = \sigma_i)} \\
&= \sum_{\Delta p_z \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{SUC}}(w_i),\mathcal{W}_{\mathrm{PO}}}} \Pr(\Delta p_z = (\sigma_{z_1}, \sigma_{z_2}, \ldots, \sigma_{z_{|p_z|}})) \times \frac{\Pr(\Delta(w_i, w_{z_1}) = (\sigma_i, \sigma_{z_1}))}{\Pr(\Delta w_i = \sigma_i)\Pr(\Delta w_{z_1} = \sigma_{z_1})} \\
&= \sum_{w_{i'} \in \mathcal{W}_{\mathrm{SUC}}(w_i)} \sum_{\sigma_{i'} \in \mathcal{S}_{\mathtt{rf}}} \Bigg(\frac{\Pr(\Delta(w_i, w_{i'}) = (\sigma_i, \sigma_{i'}))}{\Pr(\Delta w_i = \sigma_i)} \\
&\qquad \times \sum_{\Delta p_{z'} \in \Delta\mathcal{P}_{\mathcal{W}_{\mathrm{SUC}}(w_{i'})\,\mathcal{W}_{\mathrm{PO}}}} \frac{\Pr(\Delta(w_{i'}, w_{z'_1}, \ldots, w_{z'_{|p_{z'}|}}) = (\sigma_{i'}, \sigma_{z'_1}, \ldots, \sigma_{z'_{|p_{z'}|}}))}{\Pr(\Delta w_{i'} = \sigma_{i'})} \Bigg).
\end{aligned} \tag{23}
$$

Because the summation for $p_{z'}$ corresponds to the dynamic observability of $w_{i'}$, we substitute $\mathrm{DyOb}_{w_{i'}}(\sigma_{i'})$ to the result of Eq. 23. □