# Cross-level Monte Carlo Framework for System Vulnerability Evaluation against Fault Attack

Meng Li[1], Liangzhen Lai[2], Vikas Chandra[2], and David Z. Pan[1]

[1]ECE Department, University of Texas at Austin, Austin, TX, USA
[2]ARM Research, San Jose, CA, USA

## ABSTRACT

Fault attack becomes a serious threat to system security and requires to be evaluated in the design stage. Existing methods usually ignore the intrinsic uncertainty in attack process and suffer from low scalability. In this paper, we develop a general framework to evaluate system vulnerability against fault attack. A holistic model for fault injection is incorporated to capture the probabilistic nature of attack process. Based on the probabilistic model, a security metric named as System Security Factor (SSF) is defined to measure the system vulnerability. In the framework, a Monte Carlo method is leveraged to enable a feasible evaluation of SSF for different systems, security policies, and attack techniques. We enhance the framework with a novel system pre-characterization procedure, based on which an importance sampling strategy is proposed. Experimental results on a commercial processor demonstrate that compared to random sampling, a 2500X speedup is achieved with the proposed sampling strategy. Meanwhile, 3% registers are identified to contribute to more than 95% SSF. By hardening these registers, a 6.5X security improvement can be achieved with less than 2% area overhead.

## 1. INTRODUCTION

System vulnerability against fault attacks has become a serious concern when designing secure systems [1–3]. Different from passive side channel analysis [4, 5], fault attack actively injects errors into hardware, which can lead to leakage of critical information [6–8] and nullification of security policies [9, 10]. In recent works [6–8], it has been demonstrated that with deliberate fault attacks, the cryptographic keys of the system can be recovered and the system execution privilege can be altered as well. Therefore, it becomes important to take the fault attack into consideration in the design stage, which leads to a strong requirement on the framework that is capable of evaluating the system vulnerability systematically.

Several methods have been proposed to evaluate the system vulnerability against fault attack in recent papers [11–13]. In [11], the authors propose a framework that is able to identify and mitigate fault attack vulnerability in the finite state machine (FSM). By extracting the state transition and identifying the don't-care states, the vulnerability of FSM against fault attacks can be analyzed. In [13], the authors analyze the system vulnerability against attacks that cause timing violation in the circuits. By considering the fault generation and propagation, accurate fault impact can be modeled for combinational logic. In [6, 14, 15], fault attack analysis is proposed for cryptographic modules including RC4, AES, DES and so on, based on the mathematical abstraction of these modules. The analysis regards error injection as a deterministic process and models the injected fault as a single-bit or single-byte error, based on which both attack strategies and countermeasures are proposed.

Despite the extensive researches on fault attack analysis, there are still fundamental problems that have not been solved. The first question is how to model the fault injection and propagation process accurately. *The fault attack process is in nature probabilistic due to the following two reasons.* First, a wide range of fault injection attack techniques have been developed, which have different capability to inject faults at targeted time, denoted as temporal accuracy, and cycle-to-cycle technique parameter variation. Secondly, the attack strategies can also be deterministic or probabilistic. Capturing the uncertainty in attack process is important but is usually ignored in existing works.

Besides the requirement on fault modeling, the second problem is how to evaluate the system vulnerability efficiently. Because the system state space grows exponentially with the increase of system size, construction of the system FSM or calculation of state transition probability quickly become intractable even for small circuits. To avoid the exploration of FSM, different simulation-based [16] and analytical methods [17] have been proposed. Simulation-based methods provide a general approach to deal with different attack scenarios with high granularity, but may suffer from slow convergence due to large sample variance. Analytical methods [17] provide fast evaluation speed but have relatively low accuracy and granularity. How to increase the efficiency without sacrificing the accuracy is thus another critical question.

In this paper, we propose a cross-level framework that directly targets at the two problems in fault attack evaluation. A holistic model is first proposed to capture the probabilistic nature of attack process by regarding the attack parameters as samples of random variables following different distributions. Based on the model, a security metric, defined as *System Security Factor* (SSF), is proposed to measure the vulnerability of the system and provide guidance for further design optimization. SSF is defined as the probability of a successful attack, which incorporates the uncertainty of attack process to allow for a much more accurate evaluation of different attack techniques and strategies. A cross-level evaluation framework is developed to evaluate SSF in general systems and further enhanced with a novel system pre-characterization procedure. Based on the pre-characterization, registers in the system are classified to enable a hybrid evaluation with both analytical analysis and importance Monte Carlo sampling. We summarize our contributions as follows:

- A holistic model is proposed to capture the intrinsic uncertainty of attack techniques in a probabilistic manner, based on which SSF is defined to measure the system vulnerability.
- A Monte Carlo framework is developed for SSF evaluation and further enhanced with a novel system pre-characterization procedure for better efficiency.
- Our framework is verified on a commercial processor and demonstrates good efficiency and accuracy.

The rest of the paper is organized as follows. In Section 2, we describe the motivation of our work. Section 3 formally defines the attack model, the holistic model for fault injection process and the proposed metric on

*DAC '17, June 18-22, 2017, Austin, TX, USA*
© 2017 ACM. ISBN 978-1-4503-4927-7/17/06. . . $15.00
DOI: http://dx.doi.org/10.1145/3061639.3062220

system vulnerability. In Section 4, we present our pre-characterization procedure, based on which an importance sampling strategy is proposed. Section 5 describes our cross-level fault modeling strategy. We demonstrate the performance of the framework in Section 6 and conclude our work in Section 7.

# 2. MOTIVATION

Security mechanisms in modern systems are usually built in a hierarchical manner. To guarantee certain security policy, cross-level mechanisms, including hardware security primitives in circuit level, instruction set architecture support in architecture level, firmware in application level and so on, are combined together. Higher-level security protections rely on the correct functionality of lower-level primitives. With the security hierarchy, an end-to-end protection can be realized under cost and performance constraints.

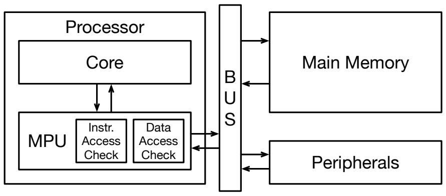


**Figure 1:** Diagram for MPU execution.

Fault attack, however, directly targets at the lower level hardware. By injecting errors into key security primitives, the functionality of the hardware can be altered temporally, which fundamentally changes the assumption on the hardware that the entire system security mechanisms rely on. For example, in a processor, memory protection unit (MPU), as shown in Figure 1, provides memory isolation features. After the data and instruction access patterns are defined, all memory access from the core and peripherals will be checked by MPU to determine whether the operations are legitimate or not. Once illegal memory operations are detected, higher-level security mechanisms will execute to isolate the process, and thus, prevent malwares from affecting other processes maliciously. If the correct behavior of an MPU is altered, it is possible for the malware to bypass the MPU and then, the entire security mechanism.

Because fault attack threatens the system security seriously, we propose our security metric and framework towards an accurate and efficient evaluation, target at helping the designers from the following aspects:

- Quantitatively characterize and compare the system vulnerability against different fault attack techniques.
- Identify security critical system components for protection under cost and overhead constraints.
- Evaluate and compare the effectiveness of different countermeasures and guide further design optimization.

In the following sections, we will formally define our security metric and describe the proposed framework.

# 3. PROBLEM FORMULATION

In this section, we formally define the proposed metric on the system vulnerability against fault attack. We first define the attack model by specifying the knowledge and target of the attackers as well as the attack flow. Then, a holistic model that captures the uncertainty in the fault injection process is proposed. The security metric is then defined based on the holistic model.

## 3.1 Attack Model

We define the attack model from the following three aspects: 1) the knowledge of the attackers; 2) the target of the attackers; 3) the attack flow. The attacker's knowledge is stated as below:

- The attacker knows the system and its physical implementation.
- The attacker has physical access to the system and can inject faults into the system.
- The attacker can choose the workload program.
- The attacker does not know the system configuration.

The target of the attacker can be roughly classified into two categories: 1) bypassing the existing security mechanisms to enable malicious operations on the system [10], e.g. illegal memory access; and 2) causing leakage of important system information [11], e.g. cryptographic keys.

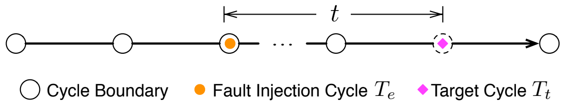


**Figure 2:** Fault attack flow.

To enable a successful attack, we consider the following unified attack flow as shown in Figure 2, which is capable of representing both of the two different scenarios. We distinguish two important timing in the attack, including target cycle $T_t$ and error injection cycle $T_e$, and define $t = T_t - T_e$. Depending on the attack scenarios, the three parameters can have different meanings. For the first scenario, $T_t$ represents the time in the workload program when the operations that violate system security policies are carried out while $T_e$ represents the time when errors are injected into the system to help the malicious operations from being detected. For the second scenario, $T_e$ still represents the time when errors are injected while $T_t$ denotes the time when system information can be observed illegally, as stated in [14]. How to choose $T_t, T_e$ and $t$ highly depends on the target and attack strategies of the attackers. In our paper, we focus on the first scenario but the proposed metric and framework are flexible for different attack categories. We assume the attackers want to access and modify certain memory locations without permission. Then, illegal memory access and modification is carried out in $T_t$ while errors are injected in $T_e$ to help with the illegal memory access.

## 3.2 Holistic Fault Injection Modeling

To inject faults into the hardware, different techniques have been proposed, including modification in power supply voltage or clock, chip overheating, Focused Ion Beam (FIB), and so on [2]. In order to provide a generic model, we consider timing distance $t$ and technique parameters $\boldsymbol{p}$ to characterize attack process. $\boldsymbol{p}$ is a vector of characterization parameters that may vary depending on the attack techniques. For example, for attacks based on clock modification, $\boldsymbol{p}$ consists of the amplitude and duration of injected clock glitches, region impacted by the injection and so on, while for radiation-based attacks, $\boldsymbol{p}$ includes the number of impacted cycles, location and area of the radiated spot, and so on. Because intrinsic uncertainty exists in fault injection process, we regard both $t$ and $\boldsymbol{p}$ as samples from random variable $T$ and $\boldsymbol{P}$ that follow the distribution $f_{T,\boldsymbol{P}}$. $f_{T,\boldsymbol{P}}$ is determined by considering the uncertainty of both different attack techniques and attack strategies.

In this paper, we consider fault injection with radiation-based techniques. Because the physical mechanism of radiation-based techniques is similar to that of soft error induced by partial strike [2], we leverage the physical model following [16]. To characterize the fault injection process, we consider $\boldsymbol{p} = [g, r]$, where $g$ denotes the center of the radiation and $r$ denotes the radius of the radiated region. We assume that only one cycle is impacted by each fault injection, but our framework can easily incorporate multi-cycle impact into the evaluation. We assume one radiation can cause voltage transients at all the gates that are in the radiated region and leverage the method in [18] to determine all the impacted gates based on $g$ and $r$. Due to the temporal accuracy and parameter variation of the attack techniques, we assume the corresponding random variable $T$ and $\boldsymbol{P}$ follow a uniform distribution with the range centered at the targeted time and expected parameter. We will investigate how the change of distribution will impact the overall system vulnerability in experimental results.

## 3.3 System Security Factor

In this section, we define the security metric that measures the system vulnerability. The execution of the system can be formally abstracted as an FSM $F = \langle \Omega_I, \Omega_O, \Omega_S, \delta, \rho, s_0 \rangle$, where $\Omega_I$, $\Omega_O$, $\Omega_S$, $s_0$ denote the set of possible primary inputs (PI), outputs (PO), internal states and initial states, respectively [16]. $\delta : \Omega_S \times \Omega_I \to \Omega_S$ is the state-transition function while $\rho : \Omega_S \times \Omega_I \to \Omega_O$ is the output function.

Due to the fault injection attack with $t$ and $\boldsymbol{p}$, the original FSM is extended to a faulty FSM with a new set of states $\Omega'_S$, state transition function $\delta'$ and output function $\rho'$, which has an additional dependence on attack time and parameters:

$$\delta' : \Omega'_S \times \Omega_I \times \Omega_T \times \Omega_{\boldsymbol{P}} \to \Omega'_S,$$
$$\rho' : \Omega'_S \times \Omega_I \times \Omega_T \times \Omega_{\boldsymbol{P}} \to \Omega_O.$$

where $\Omega_T$ and $\Omega_{\boldsymbol{P}}$ denotes the set of possible values for $T$ and $\boldsymbol{P}$, respectively.

For an error-free system, malicious operations or observations may subject to existing security mechanisms and cause security violations. Fault attack can help change the original system state transitions so that security mechanisms are not triggered. Therefore, how easy are the attackers able to create such illegal transition to bypass existing mechanisms becomes the key question to answer for system vulnerability evaluation. We define SSF as the criterion to measure the probability for such illegal transition considering the uncertainty in fault attack process. More formally, let $e$ be an indicator variable representing whether the illegal transition is created. For a given benchmark, $e$ is a function of attack parameters $t$ and $\boldsymbol{p}$. Therefore, we are able to define SSF as

$$SSF = \mathrm{E}_{T,\boldsymbol{P}}(E).$$

where $E$ represents the corresponding random variable of $e$.

To evaluate SSF, we propose to use Monte Carlo based method. Let $N$ denote the total number of fault attacks at timing distance $t_1, \ldots, t_N$. Then, an empirical finite-sample estimate of SSF becomes

$$S\hat{S}F = \frac{1}{N} \sum_{t_i, \boldsymbol{p}_i \sim f_{T,\boldsymbol{P}}} e(t_i, \boldsymbol{p}_i).$$

To analyze the convergence of the Monte Carlo framework, we rely on the weak Law of Large Number (LLN). After $N$ samples, we have

$$\Pr[|\frac{\sum_{i=1}^{N} e(t_i, \boldsymbol{p}_i)}{N} - \mathrm{E}_{T,\boldsymbol{P}}[E]| \geq \epsilon] \leq \frac{\sigma_E^2}{N\epsilon^2},$$

where $\sigma_E^2$ represents the sample variance and $\epsilon$ denotes the empirical risk. As we can see, for fixed $\epsilon$, the convergence rate of the Monte Carlo framework is determined by the $\sigma_E^2$. To reduce the sample variance and increase the convergence rate, we propose an importance sampling strategy. The basic intuition of importance sampling is to increase the probability to sample from the cycles and the attack parameters that are more likely to result in a successful attack. Let $g_{T,\boldsymbol{P}}$ denote the actual sampling distribution, then, after $N$ samples, the SSF can be estimated by

$$S\hat{S}F' = \frac{1}{N} \sum_{t_i, \boldsymbol{p}_i \sim g_{T,\boldsymbol{P}}} \frac{f_{T,\boldsymbol{P}}(t_i, \boldsymbol{p_i})}{g_{T,\boldsymbol{P}}(t_i, \boldsymbol{p_i})} e(t_i, \boldsymbol{p_i}).$$

$g_{T,\boldsymbol{P}}$ is critical for importance sampling. In Section 4, we will propose our pre-characterization strategy to determine $g_{T,\boldsymbol{P}}$.

# 4. IMPORTANCE SAMPLING VIA SYSTEM PRE-CHARACTERIZATION

In this section, we describe our novel system pre-characterization procedure and derive the sampling distribution $g_{T,\boldsymbol{P}}$. The system pre-characterization is carried out based on the following observations.

**Observation 1:** *In a system, only a few modules are related to certain security mechanism. These modules communicate with the core by a few signals and determine the related transitions of FSM.*

We denote these modules as security-critical modules and these signals as responding signals. When malicious operations are conducted and detected by these critical modules, responding signals are set to control the transition of circuit FSM to handle the security violation. To create illegitimate transitions at the presence of malicious operation to bypass the security policy, the attackers need to either prevent the security-critical modules from setting the responding signals or prevent the responding signals notifying the processor. Because only the circuits in the fanin and fanout cones of the responding signals can potentially impact them, we just consider attacks on these parts of the circuit.

Therefore, the first step of our system pre-characterization is identify responding signals based on the system specification. Then, to get the circuit components in the fanin and fanout cones of the responding signals, we unroll the circuit netlist and traverse the unrolled netlist in a breadth-first order starting from the identified signals. Because the number of responding signals is usually small, the circuits in the fanin and fanout cones are only of a small portion of the entire netlist, which helps to reduce the overall sample space and the sample variance significantly.

**Observation 2:** *Bit flips at different circuit nodes can have different correlation with the flips at responding signals.*

The observation enables us to further distinguish the gates or registers in the fanin and fanout cones of the responding signals. As we have described, because the machine state transitions are controlled by these responding signals, to create illegitimate transitions, the attackers need to inject faults to either change the value of responding signals or block their propagation to the core. Circuit nodes that have a high bit flip correlation with responding signals are more likely to accomplish the target and thus, should have a higher probability in the sampling distribution.

To evaluate the bit flip correlation, we define the switching signature for each circuit node in the fanin and fanout cones. The switching signature is a sequence of binary values that indicate whether the logic value for the circuit node switches or not at each cycle. Let a binary vector $\mathrm{ss}(g)$ denotes the switching signature for circuit node $g$. For the $i$th cycle, $\mathrm{ss}_i(g) = 1$ if the logic value switches between the $(i-1)$th cycle and the $i$th cycle and $\mathrm{ss}_i(g) = 0$ otherwise. Consider circuit node $g$ in the $i$th unrolled circuit, then, the bit flip correlation between $g$ and responding signal $rs$ is computed by

$$\mathrm{Corr}_i(g, rs) = \frac{|\mathrm{ss}(g) \& (\mathrm{ss}(rs) \ll i)|}{|\mathrm{ss}(g)|},$$

where $\&$ represents the bitwise and operation and $|\cdot|$ denotes the $\ell_1$ norm, which calculates the hamming weight of the binary vector. $\mathrm{ss}(rs)$ is shifted to the left by $i$ bits because it takes $i$ cycles for the bit flips in the $i$th unrolled circuit to reach the responding signal. Note that $i \geq 0$ if $g$ is in the fanin cone of $rs$ and $i < 0$ if $g$ is in the fanout cone of $rs$.

To evaluate the bit flip correlation, in the second step of our system pre-characterization, we first do a RTL-level simulation with synthetic benchmarks and keep a record of the logic value for each register. Then, gate-level logic simulation is carried out to derive the logic value as well as switching signature for each circuit node, i.e. combinational gates. Because we are able to use fast bit-parallel calculation, the overall calculation can be very efficient. Then, starting from the responding signals, we traverse their fanin and fanout cones and calculate the bit flip correlation for each circuit node. We use the following example to illustrate the calculation of bit flip correlation in a responding signal's fanin cone.

Example As shown in Figure 3, after the logic-level simulation with synthetic benchmark, the logic value for each circuit node is recorded. The switching signature for each node is derived accordingly. Then, the switching correlation for gate $g_1$, $g_2$, $g_3$ and responding signal $rs$ can be computed as

$$\mathrm{Corr}_0(g_1, rs) = \frac{|00101101 \& (01001101 \ll 0)|}{|00101101|} = \frac{3}{4}$$
$$\mathrm{Corr}_0(g_2, rs) = \frac{|01100111 \& (01001101 \ll 0)|}{|01100111|} = \frac{3}{5}$$
$$\mathrm{Corr}_1(g_3, rs) = \frac{|01001111 \& (01001101 \ll 1)|}{|01001111|} = \frac{2}{5}.$$

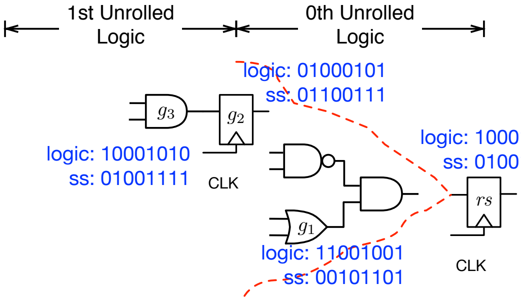


**Figure 3:** Example of calculating the bit flip correlation between internal gates, i.e. $g_1, g_2, g_3$, and responding signals $rs$.

**Observation 3:** *Bit errors can stay in some registers without propagating to other registers or getting masked.*

For these registers, sampling based method can be inefficient. This is because for bit errors injected into these registers even with a large timing distance $t$, it is still possible for them to cause illegitimate machine state transitions. Therefore, to enable an accurate evaluation, $\Omega_T$ has to be enlarged to cover a large range of values for $T$, which may lead to large sample space and large sample variance.

We denote these registers as memory-type registers and the other registers as computation-type registers. To characterize the registers, we propose two parameters: error lifetime and error contamination number. Error lifetime captures the number of cycles one bit error stays in the system before gets masked, while error contamination number captures the propagation of bit errors originating from one register to other registers. For the memory-type registers, since the bit errors tend to stay locally without further propagation or getting masked, they usually have a long error lifetime and close-to-0 error contamination number.

Because of the different characteristics of the memory-type and computation-type registers, we choose to use different strategies to evaluate the system vulnerability originated from these registers. For memory-type register, though long error lifetime makes sampling method inefficient, it actually enables us to analyze the error outcome analytically. This is because the outcome of fault attack on these registers is not determined by the timing distance between fault injection cycle and target cycle but mainly by the functionality of the memory-type registers in the system. Therefore, we choose to evaluate these registers analytically considering the system configuration, faulty registers, and benchmarks, which avoids error injection simulation without compromising the overall accuracy.

For computation-type registers, though we still use sampling based method for the evaluation, we can also increase the convergence rate due to the following two reasons. On one hand, since the error lifetime for these registers is usually small, only a small range of $T$ need to be considered. On the other hand, after we finish sampling the timing distance $t$, we only need to consider those registers with error lifetime larger than $t$, while for the rest, we know the attack will fail because the injected errors will get masked before the target cycle.

Therefore, the third step of our system pre-characterization would be to determine the error lifetime and contamination number for each register. We leverage fast RTL-level simulation with synthetic benchmarks. Bit errors are injected into each register that is in the fanin and fanout cones of the responding signals and the two characterization parameters are then collected. We show the collected statistics from the commercial processor we use in Figure 4 (a) and (b). As we can see, more than half of the total registers have long lifetime and 0 contamination number, which are classified as memory-type registers.

Therefore, based on the observations and the pre-characterization procedure above, we propose the following sampling strategy:

- First, we decompose the overall sampling distribution $g_{T,\boldsymbol{P}}$ as

$$g_{T,\boldsymbol{P}} = g_T \cdot g_{\boldsymbol{P}|T}.$$

- Then, we sample the timing distance $t$ following the distribution

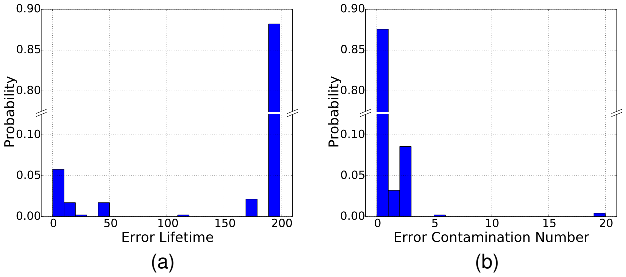


**Figure 4:** Distribution of characterization parameters of different registers in the commercial processor: (a) error lifetime and (b) error contamination number.

$g_T$. Here, for certain timing distance $t = i$, we have

$$g_T(t = i) = \frac{\omega_i}{\sum_i \omega_i},$$

where $\omega_i = \sum_{g \in \Omega_i} (1 + \alpha \mathrm{Corr}_i(g, rs) \delta(\mathrm{L}(g) \geq \beta i))$, $\Omega_i$ denotes the set of gates in the fanin and fanout cones of responding signals in the $i$th unrolled circuit and $\mathrm{L}(g)$ represents the error lifetime of $g$. If $g$ is a register, $\mathrm{L}(g)$ is the error lifetime of itself. If $g$ is a combinational gate, $\mathrm{L}(g)$ equals to the maximum error lifetime of the registers in the fanout cone of $g$ in the $i$th unrolled circuit. $\alpha$ and $\beta$ are configurable parameters that control the calculation of the distribution.

- Third, we sample the technique related parameters $\boldsymbol{p} = [g, r]$ for cycle $t = i$ following the conditional distribution $g_{\boldsymbol{P}|T}$ where

$$g_{\boldsymbol{P}|T}(\boldsymbol{p}|t = i) = \begin{cases} \frac{(1+\alpha \mathrm{Corr}_i(g,rs)\delta(\mathrm{L}(g) \geq \beta i))\mathrm{Unif}(r)}{\sum_{h \in \Omega_i} 1+\alpha \mathrm{Corr}_i(h,rs)\delta(\mathrm{L}(h) \geq \beta i)} & \text{if } g \in \Omega_i \\ 0 & \text{otherwise} \end{cases}$$

where $\mathrm{Unif}(r)$ denotes the uniform distribution for the radius of the radiated area.

Following the procedure above, we can get one sample of $t$ and $\boldsymbol{p}$ from $g_{T,\boldsymbol{P}}$. To determine $g_{T,\boldsymbol{P}}$, only logic level simulation is required for the three steps in our pre-characterization, which can be finished efficiently. Meanwhile, since the pre-characterization only needs to be conducted once, its introduced overhead on the overall evaluation is negligible.

# 5. CROSS-LEVEL FAULT PROPAGATION SIMULATION

In this section, we describe the proposed cross-level simulation framework. The overall flow is illustrated in Figure 5. In the first step, we run the RTL-level simulation without any fault attack, denoted as golden run. During the golden run, golden checkpoints that contain the logic values of all registers are dumped. Then, we follow the importance sampling strategy to generate the sample of fault injection cycle and attack parameters. In the third step, we restart the RTL-level simulation from the nearest golden checkpoint and run the simulation to the fault injection cycle. The simulation then switches to gate level in the fourth step to evaluate the gate-level mechanisms and calculate the errors latched by registers at the end of fault injection cycle. Depending on the type of the faulty registers, we may use analytical evaluation or restore RTL level simulation. One fault propagation simulation is terminated when the error outcome can be determined.

## 5.1 RTL-level Golden Simulation

Before the fault attack run, a complete run of the benchmark is performed, termed as the golden run. During the golden run, golden checkpoints are dumped at intermediate points. The golden run is only performed in RTL level. Because the golden checkpoints contain the logic values of all registers in the system, they permit the following advantages:

- Golden checkpoints help reduce the warm-up simulation before the fault injection cycle in fault attack runs since the simulation can be restarted directly at the nearest golden checkpoints.

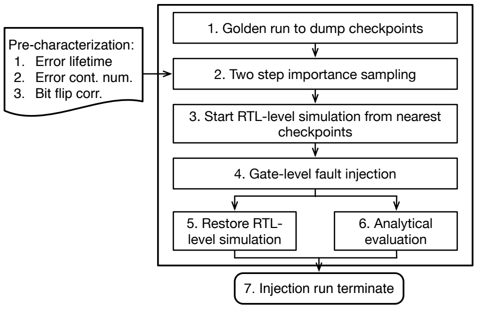

Pre-characterization:
1. Error lifetime
2. Error cont. num.
3. Bit flip corr.
1. Golden run to dump checkpoints
2. Two step importance sampling
3. Start RTL-level simulation from nearest checkpoints
4. Gate-level fault injection
5. Restore RTL-level simulation
6. Analytical evaluation
7. Injection run terminate

**Figure 5:** Overall cross-level evaluation framework.

- By comparison with golden checkpoints, the outcome of fault attack run can be determined.

For one benchmark, the RTL-level golden run only needs to be conducted once.

## 5.2 RTL-level Fault Attack Simulation

After the golden run, we first sample the timing distance $t$ and attack parameter vector $\boldsymbol{p}$ following the importance sampling strategy. Then, the RTL-level simulation is restarted from the nearest golden checkpoints and runs to the fault injection cycle. The simulation is then switched to gate level to determine the bit errors latched by registers by the end of fault injection cycle. Depending on the type of faulty registers, we choose different strategies for the downstream evaluation. When errors only exist in memory-type registers, we only need analytical evaluation to determine the error impact. Otherwise, we will inject the bit error back to RTL level to restart the RTL-level simulation. Once the RTL-level simulation resumes, the logic value of each register is dumped at different checkpoints and compared with the golden run to determine whether the target illegitimate transitions are created or not. One fault attack run is completed thereby. The whole process is continued until the empirical estimate converges.

## 5.3 Gate-level Fault Attack Modeling

Gate-level simulation only runs for the fault injection cycle. From the sampled attack parameter vector $\boldsymbol{p}$, we can get the radiated region and determine all the voltage transients at the output of the impacted gates as described in Section 3.2. Then, as shown in Figure 6 (a), we traverse the whole circuit netlist in a topological order to propagate the voltage transients to the registers. A voltage transient can get latched if it arrives at the register and satisfies the setup and hold time requirements as in Figure 6 (b). Because the generated voltage transients can get latched by more than one registers, by the end of fault injection cycle, we determine a set of faulty registers.

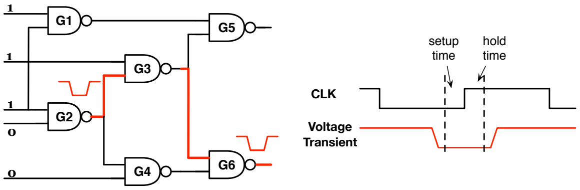


**Figure 6:** Gate-level modeling: (a) propagation of voltage transients; (b) latching condition.

To demonstrate the necessity of including the gate-level modeling into the overall framework, we collect the bit error patterns by the end of fault injection cycle. As shown in Figure 7 (a), around 14.5% errors exist in multiple bytes while for the errors within a single byte, none of the errors exists in all the bits. This proves the inaccuracy of current assumption on either single-bit or single-byte error. We also compare the number of error patterns captured by considering the attack on combinational gates and registers. As shown in Figure 7 (b), the number of error patterns induced by the attack on combinational gates is much larger than that of registers, which proves that simply considering fault attack on sequential elements, i.e. registers, is not accurate enough.

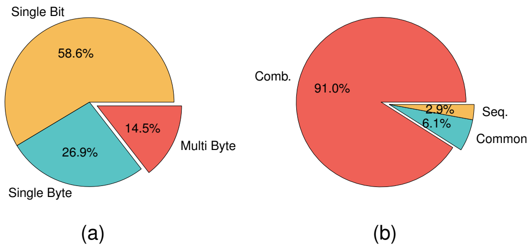


**Figure 7:** Bit error patterns generated by gate-level simulation: (a) error distribution; (b) comparison between error patterns induced by attack on combinational gates and sequential elements.

# 6. EXPERIMENTAL RESULTS

In this section, we report on the extensive experiments to demonstrate the effectiveness of the proposed Monte Carlo evaluation framework. All the experimental results are carried out on a commercial processor. We evaluate the security policy related to the memory access pattern for both core and peripherals and study the fault attack vulnerability associated with MPU. We follow the attack flow as discussed in Section 3 and evaluate SSF as the probability for the attacker to bypass the memory access policy. The benchmark we use is written in C++ which includes illegal memory write and read operations. We use Synopsys VCS as the RTL-level simulator and implement our own gate-level simulator with C++ following the algorithm shown in [16].

We first demonstrate the effectiveness of the importance sampling strategy. The sampling distribution for the timing distance $t$ is shown in Figure 8 (a). In Figure 8 (b), we demonstrate the reduction of sample space. As we can see, with the importance sampling strategy, the sample space is reduced significantly. To demonstrate the increase of the convergence rate, we compare random sampling, importance sampling from fanin/fanout cones, and our mixed strategy that combines importance sampling with analytical analysis for memory-type registers. The range of $t$ is 50 cycles and the range for $P$ includes a sub-block of gates of around $1/8$ of MPU identified following [18]. As shown in Figure 9 (b), our importance sampling strategy permits much faster convergence rate compared to the other two methods. We list the detailed statistics in Table 9. As we can see, the sample variance of our importance sampling strategy is $9.70 \times 10^{-05}$. Compared to 0.0261 of random sampling, it permits more than 2500X increase. Although the speedup is related to the systems, benchmarks and uncertainty of attack process, the increase of convergence rate significantly increases the potential of our framework to deal with large practical systems.

From the SSF evaluation, we are able to recognize that there are around 3% registers that contribute to more than 95% SSF. Since we consider radiation-based attacks, the physical mechanisms of which are similar to that of soft error induced by particle strike [2], we are able to leverage similar protection techniques as proposed in [19, 20]. Suppose we use error resilient designs for the identified 3% registers, which permits around 10X better resilience with 3X area overhead [19, 20], then the overall SSF can be reduced by up to 6.5X with less than 2% increase of MPU area. Though the results depend on the attack and mitigation techniques, they can demonstrate the capability of our framework to identify key circuit components that are critical to the overall system security, which can be used to guide design optimization.

Then, we show the statistics of fault attacks on combinational gates. In Figure 10 (a), as we can see, 68.4% of the fault attacks are masked. 28.6% of the attacks only cause bit errors at memory-type registers, which only requires analytical analysis. Only 3.1% of the total fault attack runs require further RTL-level simulation. This further proves the effectiveness to distinguish between memory- and computation-type registers. We also compare the SSF induced by attacks on combinational gates and registers. As shown in Figure 10 (b), SSF due to fault attacks

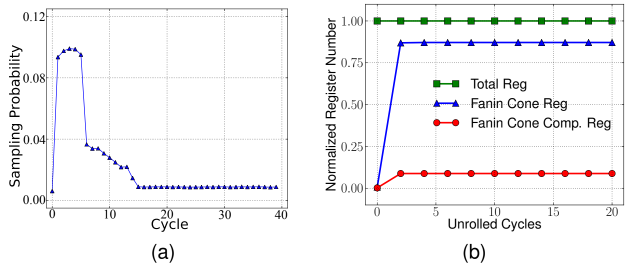
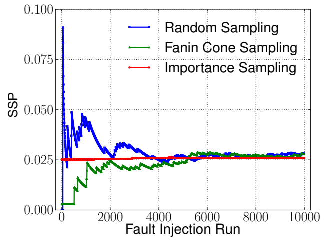


**Figure 8:** Effectiveness of importance sampling strategy: (a) sampling distribution for different value in $\Omega_T$; (b) reduction of sample space with our importance sampling strategy.

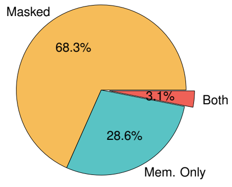
on combinational gates is around 25.8% of the SSF induced by attacks on registers. We also notice that all these successful attacks are from the fault injection into the fanin cone of the 3% registers as we have identified above. Therefore, to protect the MPU against fault attack, both the registers themselves and gates in their fanin cone need to be protected.



| Strategy | $s^2$ |
|---|---|
| Random | 0.0261 |
| Fanin Cone | 0.0210 |
| Our | $9.70 \times 10^{-05}$ |

**Figure 9:** Convergence comparison of different sampling strategies: (a) convergence plot; (b) detailed statistics for different strategies, including the number of successful attacks out of 2000 attacks and sample variance.

We also evaluate the impact of the temporal accuracy and parameter variation of the attack techniques on the overall SSF. For fixed attack parameter variation, we assume the temporal accuracy follows a uniform distribution within a specific range. As shown in Figure 11(a), with the overall range decreasing, the normalized SSF increases significantly for both benchmarks on illegal memory write and read. To evaluate the impact of parameter variation, we fix the temporal accuracy. As shown in Figure 11 (b), as the variation of radiated gates changes from the largest, i.e. uniform distribution over all the gates, to the smallest, i.e. delta function centered at target gates, the normalized SSF also increases significantly. As indicated by the results, different attack techniques can have very different impact on the overall system security. The capability to capture the intrinsic uncertainty in attack process is thus important to enable a practical evaluation for different attack techniques and a feasible guidance for different systems.



| Strategy | # Succ. Attack | SSF |
|---|---|---|
| Registers | 271 | 0.027 |
| Comb. Gates | 70 | 0.007 |

**Figure 10:** Comparison between SSF induced by attack on combinational gates and registers: (a) error statistics induced by attacking combinational gates; (b) SSF comparison.

## 7. CONCLUSION

In this paper, we propose an accurate and efficient framework to evaluate system vulnerability due to fault attack. A holistic model is proposed to capture the uncertainty of fault injection process, based on which a security metric named as SSF is defined. To evaluate SSF in practical systems, a Monte Carlo framework is proposed and further enhanced by an importance sampling strategy based on a novel system precharacterization. A cross-level fault propagation modeling strategy that is fully compatible with the Monte Carlo simulation is also integrated. Experimental results demonstrate that with importance sampling strategy, the convergence rate is increased by up to 2500X. We also examine the impact of temporal accuracy and attack parameter variation to prove the necessity of considering the intrinsic uncertainty in attack process.

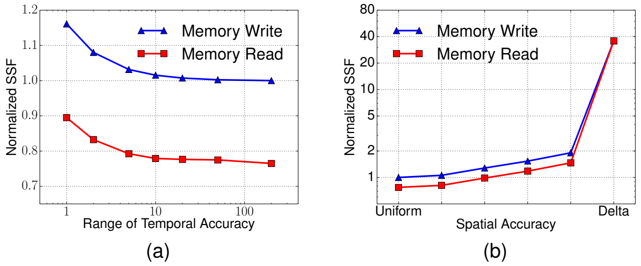


**Figure 11:** Impact of temporal accuracy and attack parameter variation on SSF.

## 8. REFERENCES

[1] Y. Li et al., "Fault sensitivity analysis," in *Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems*, 2010.

[2] A. Barenghi et al., "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proc. of the IEEE*, 2012.

[3] B. Yuce et al., "Improving fault attacks on embedded software using risc pipeline characterization," in *Proc. IEEE Workshop Fault Diagnosis and Tolerance in Cryptography*, 2015.

[4] D. Agrawal et al., "The EM side-channels," in *Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems*, 2002.

[5] R. Hund et al., "Practical timing side channel attacks against kernel space aslr," in *Proc. IEEE Symp. on Security and Privacy*, 2013.

[6] M. Tunstall et al., "Differential fault analysis of the advanced encryption standard using a single fault," in *Proc. Int. Workshop on Information Security Theory and Practices*, 2011.

[7] L. Hemme, "A differential fault attack against early rounds of (triple-) des," in *Proc. Int. Conf. on Cryptographic Hardware and Embedded Systems*, 2004.

[8] E. Biham et al., "Impossible fault analysis of rc4 and differential fault analysis of rc4," in *Proc. Int. Workshop on Fast Software Encryption*, 2005.

[9] J. G. Van Woudenberg et al., "Practical optical fault injection on secure microcontrollers," in *Proc. IEEE Workshop Fault Diagnosis and Tolerance in Cryptography*, 2011.

[10] B. Yuce et al., "FAME: Fault-attack aware microprocessor extensions for hardware fault detection and software fault response," 2016.

[11] A. Nahiyan et al., "AVFSM: a framework for identifying and mitigating vulnerabilities in FSMs," in *Proc. IEEE/ACM Design Automation Conf.*, 2016.

[12] H. Salmani et al., "Analyzing circuit vulnerability to hardware trojan insertion at the behavioral level," in *Proc. IEEE Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2013.

[13] B. Yuce et al., "TVVF: Estimating the vulnerability of hardware cryptosystems against timing violation attacks," in *Proc. IEEE Int. Symp. on Hardware Oriented Security and Trust*, 2015.

[14] C.-N. Chen et al., "Differential fault analysis on aes key schedule and some countermeasures," in *Proc. Australasian Conf. on Information Security and Privacy*, 2003.

[15] J. Fan et al., "State-of-the-art of secure ecc implementations: A survey on known side-channel attacks and countermeasures.," in *Proc. IEEE Int. Symp. on Hardware Oriented Security and Trust*, 2010.

[16] M. Li et al., "A monte carlo simulation flow for seu analysis of sequential circuits," in *Proc. IEEE/ACM Design Automation Conf.*, 2016.

[17] S. S. Mukherjeeet al., "The soft error problem: An architectural perspective," in *Proc. Int. Symp. on High-Performance Computer Architecture*, 2005.

[18] M. Fazeli et al., "Soft error rate estimation of digital circuits in the presence of multiple event transients (METs)," in *Proc. Design, Automation and Test in Europe*, 2011.

[19] S. Mitra et al., "Robust system design with built-in soft-error resilience," *J. of Computer*, 2005.

[20] M. Zhang et al., "Sequential element design with built-in soft error resilience," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 2006.