PATRON: A Pragmatic Approach for Encoding Laser Fault Injection Resistant FSMs

Muhtadi Choudhury*, Shahin Tajik‡, and Domenic Forte*
* University of Florida, Gainesville, FL, USA
‡ Worcetser Polytechnic Institute, Worcester, MA, USA

***Abstract*—Since Finite State Machines (FSMs) regulate the overall operations in majority of the digital systems, the security of an entire system can be jeopardized if the FSM is vulnerable to physical attacks. By injecting faults into an FSM, an attacker can attain unauthorized access to sensitive states, resulting in information leakage and privilege escalation. One of the powerful fault injection techniques is laser-based fault injection (LFI), which enables an adversary to alter states of individual flip-flops. While standard error correction/detection techniques have been used to protect the FSMs from such fault attacks, their significant overhead makes them unattractive to designers. To keep the overhead minimal, we propose a novel FSM encoding scheme based on decision diagrams that utilizes don't-care states of the FSM. We demonstrate that PATRON outperforms conventional encoding schemes in terms of both security and scalability for popular benchmarks. Finally, we introduce a vulnerability metric to aid the security analysis, which precisely manifests the susceptibility of FSM designs.**

***Index Terms*—Laser Fault Injection, Fault Tolerance, Coding Theory, Decision Diagram.**

I. Introduction

The security and integrity of microprocessors, system-on-chips (SoCs), and cryptographic hardware can be threatened by physical attacks, e.g., side-channel, fault injection, and probing attacks. Fault injection attacks are particularly powerful due to their active nature. For example, by varying the supply voltage or the clock frequency, an attacker can cause erroneous operation of a target device. Among different classes of fault injection attacks, laser fault injection (LFI) attacks are the most sophisticated ones in terms of accuracy and effectiveness. The primary targets of LFI attacks are the memory components, such as SRAM cells and flip-flops, which are essential in the finite state machines (FSMs) of an SoC. By inducing a fault into an FSM, attackers may bypass certain states, referred to as authorized states, and get access to so-called protected states [1]. As a result, they might be able to extract a cryptographic key, gain privileged access to a service, or merely mount a denial of service attack.

Several countermeasures have been proposed to mitigate the shortcomings of conventional FSMs against LFI. While physical countermeasures, such as tamper-proof packaging and light sensors, can be effective, the cost and extra manufacturing steps make the logical circuit-based countermeasures more attractive to designers. For logical countermeasures, CAD tools can be deployed to improve the FSM resiliency. For instance, by duplicating the FSM or changing the encoding, an FSM can become more resilient against fault attacks. Due to the similarity of the fault concept in the test and reliability community, security designers have started to adopt coding-based and information-theoretic mitigation techniques (e.g., self-checking circuits [2], and extend them to match the security requirement of a design against different fault attacks. However, in contrast to self-checking circuits, where natural faults occur randomly in a unidirectional way, an adversary can cause bidirectional non-uniform faults using LFI. Therefore, several linear and non-linear coding schemes have been proposed to transform the encoding of an FSM to a fault-resilient FSM encoding [3] [4].

While the fault resilient coding schemes are provably secure, their main disadvantage is the large overhead in terms of consumed area on the chip. Using coding schemes to make an FSM fault-resilient could mean the number of flip flops (and potentially the associated decoding logic) is substantially increased. Moreover, the extra circuitry for the detection and correction of the faults impose nontrivial timing constraints on the design. The primary reason for this large overhead is the inadequacy of these schemes to differentiate the degrees of sensitivity of states – even though only a few states in an FSM might be security-critical, the conventional encoding schemes protect all states equally. This equality stems from the fact that these coding techniques are designed primarily for communication systems, where all possible states (i.e., transferred messages) have to be protected from a noisy channel to avoid potential errors on the receiver side. However, in an FSM, a considerable number of states are not security-critical and do not expose any confidential information if they are breached by fault injection. If these states are regarded as critical, it may substantially impact the design overhead.

Our Contribution. A key question is whether there exists a scalable and low overhead scheme that can be tuned to protect only specific states against an adversary capable of inducing simultaneous LFI into multiple flip-flops. In this paper, we introduce *pragmatic* encoding that uses decision diagrams to enforce a strong and well-defined LFI model for an FSM. PATRON, i.e., a pragmatic approach for encoding LFI resistant FSMs, incorporates minimum encoding length, number of required normal and sensitive states, number of bit-flips by the attacker, *code rate*, and their interrelationships, which were previously unexplored. Moreover, we compare the efficiency of our proposed approach with conventional encoding schemes. We also propose a novel metric, namely, *vulnerability metric (VM)* that provides a new perspective on

978-3-9819263-5-4/DATE21/©2021 EDAA

FSM susceptibility due to LFI. Note that our intention is not to detect or correct any faults in the FSM; rather, we wish to guarantee the protection of sensitive states from an adversary with a pre-defined laser setup by re-encoding the FSM. This differentiates PATRON from traditional approaches, that require intricate logic for error detection/correction.

II. Background and Terminology

A. FSM and FSM Encoding

An FSM is defined as a 5-tuple $(S, I, O, \varphi, \lambda)$, where S is a finite set of states, I is a finite set of input symbols, O is a finite set of output symbols, $\varphi : S \times I \rightarrow S$ is the next-state function and $\lambda : S \times I \rightarrow O$ is the output function. Let $|\cdot|$ denote cardinality in a set, and thus $|S|$ is the number of states in the FSM. Typically, an FSM is conveniently depicted as a directed graph $G = (S, T)$, referred to as state transition graph (STG), where each state $s \in S$ represents a vertex and each edge $t_{yz} \in T$ represents a transition or edge from state y to the state z. In the STG, each state can be accessed from only the *accessible set of states*

$$A(z) = \{y \mid t_{yz} \in T\} \tag{1}$$

State encoding assigns a unique pattern of 0s and 1s for each state. Two common encoding strategies are discussed below.

Binary Encoding: Here, states are allocated in a binary sequence starting from 0. The number of state flip-flops (FFs), $k = \log_2 |S|$, provides the most efficient utilization of FFs.

One-Hot Encoding: Here, only one bit of the state variable is '1' while all others are '0' for every state in the FSM. For this scheme, $k = |S|$, where the number of state FFs is clearly larger than binary encoding. However, it requires simpler decoding logic than binary.

Note that these popular FSM encoding schemes are unprotected. The impact of encoding practices on security will be discussed more in Section III-B

B. Error-Detection/Correction Codes

Due to their sound theoretical basis, information redundancy techniques based on error-detecting/correction codes have become the most common countermeasure to protect cryptographic hardware implementations against fault attacks. The design of error-detecting codes assumes that faults can be modeled as additive errors. In the presence of fault, a code c becomes $c \oplus e$, where e is an error vector. These encoding approaches enable one to detect/correct up to a certain number of errors by assuring a minimum hamming distance (HD), d between codewords after encoding. For an (n, k, d) code where k-bit message (number of unprotected FSM bits or FFs in our case) is encoded to an n-bit code, using r redundant bits. The primary metric to compare the overhead of different codes is called *code rate* (CR),

$$CR = \frac{k}{n}. \tag{2}$$

As a result, larger values of CR means more efficient code. However, the theoretical Hamming bound represents maximum value of k. Hamming bound provides limitation on the efficiency of error-correcting codes. Codes achieving Hamming bound are called perfect codes. Hamming, Extended Hamming, Golay, and Extended Golay are such codes.

C. Binary Decision Diagrams (BDDs)

A BDD data structure represents Boolean functions. Conceptually, BDDs can be thought of as representing compressed sets. Common operations of Boolean functions can be performed on this compressed representation. Graphically, a BDD is a directed, rooted, and acyclic graph which may contain multiple decision nodes and two terminal nodes. Decision nodes are variables in the Boolean equation. The two types of terminal nodes are terminal-0 node and terminal-1 node. There are only two types of edges: 0-edge and 1-edge that represent the decision node values of logic 0 and logic 1, respectively.

III. FSM Security Analysis

A. Threat Model

Since the original work of [5], substantial research has been done on LFI. Still, a well-accepted threat model is lacking in the literature because of laser dependence on circuit technology, beam features, and setup (wavelength, spot size, pulse width, etc.), and the type of injection (frontside vs. backside) [6]. Admitting that the fault model, adversary, and the intended level of security vary by situation, cost, and performance have been the motivating factors in designing FSMs. Hence, from the designer's perspective, it is necessary to outline the details of the fault model and generalize the countermeasures as much as possible.

Lasers can cause faults in state elements in two ways. First, the laser can be focused on combinational logic, thereby letting the fault transient propagate to the memory cell in the memorization time window. This is known as *Single Event Transient (SET).* Alternatively, there is no time factor involved when the laser beam directly affects the memory cell. The logical value in the memory cell is instantaneously overturned; this is known as *Single Event Upset (SEU).* Further, when a single laser beam upsets several FFs concurrently, it is known as *multiple bit errors*.

The threat model considered here is one of a strong attacker who has physical access and a high-precision LFI system. More specifically, we consider the following assumptions:

1) *FSM knowledge:* The attacker is well-versed on the FSM functionality and the state encoding. All this information can be obtained through an insider or reverse-engineering. This class of attacks is vital as they can be easily performed by any knowledgeable entity with access to the right equipment.
2) *Bidirectionality of bit flips:* Errors can be unidirectional for at least some experimental conditions [7]. This means that some bits in the design may not share an equal probability of faulting to logic 0 and 1. Such effects lead to the models being known as "bit set" or "bit reset." However, in this paper, we consider a bidirectional model as it is comprehensive and represents a more realistic

threat model [8]. Thus, any countermeasure proposed here can cover faults in either direction or both directions.

3) *Number of simultaneous bit flips and their locations:* Without loss of generality, we assume that the attacker can precisely and simultaneously flip up to x number of FFs in one clock cycle. In practice, x depends on the number of lasers in the LFI setup and the FF locations in the IC layout [9]. Thus, our threat model is generalized so it could be extended to more constrained conditions.

Note that setup time violation based attacks have already been investigated in the literature [1] [10] and are considered out of scope in this paper. In those attacks, the attacker hopes to trigger a security policy issue by violating setup time in flips flops by overclocking, heating, altering the supply voltage, etc. Such attacks assume a weaker attacker who has less knowledge about the design and less precision in injecting faults, i.e., bit flips depend on race conditions in the decoding logic. On the other hand, LFI poses a greater threat as the attacker can easily access protected states by controlling the *location* of *single* bits [9]. Nevertheless, any countermeasure that covers our strong attack model will also cover weaker attacks.

B. Vulnerability of State Encoding

Given the above threat model, we analyze common FSM encoding schemes and identify their limitations. As in [10], the designer specifies a set P of *protected states* and a set L of *authorized states*. A state $s \in L$ is allowed access to P, such that $A(L) = \{p \mid p \in P\}$. If $s \notin L$ accesses P then security is breached (e.g., secret key is leaked or privileges are escalated). If a fault results in a normal state transitioning to a state $s \in L$, there is a potential vulnerability in the FSM [11]. Thus, in this paper, we define two sets of states

$$NS = \{s \in S \mid s \notin L \cup P\} \quad (3)$$

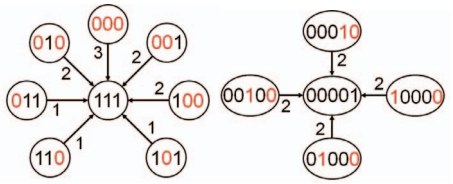
$$SS = \{s \in S \mid s \in L \cup P\} \quad (4)$$

where NS and SS denote the sets of *normal states* and *sensitive states*, respectively. Consider two FSM states s_i and s_j which are encoded to binary values of i and j respectively. Let $HD(a, b)$ denote the Hamming distance between two binary strings a and b. In our threat model, if $HD(i, j) \leq x$, $s_i \in S$, and $s_j \in SS$, then the FSM may be exploitable by LFI in one clock cycle. Since an attacker can simultaneously flip any x bits, the normal state s_i can be changed into a sensitive state s_j by laser. We define a subsets of states $VS_x \subseteq S$ as

$$VS_x = \{s_i \in S \mid HD(i, j) \leq x, \ , \ s_j \in SS\} \quad (5)$$

where VS_x represents the set of states vulnerable to x faults. We refer to these as *vulnerable states*. Clearly, the goal of a fault tolerant scheme should be $|VS_x| = 0$.

Binary encoding is the most susceptible to LFI. Fig. 1(a) shows an FSM with $|S| = 8$ and state "111" is in set SS. For successful LFI, the attacker needs to focus the laser beam and shot-instant tune [9] at the respective bit positions that cause an access violation. If $x = 3$, then the attacker can change any 3 bits in the current state and extract critical information from the design. To this end, the attackers could obtain concurrent multiple flips by directly focusing the beams onto the respective FFs or they could utilize the cone partitioning technique with maximum multiplicity [12]. In this way, it is possible to access the SS from all the 7 states in NS. For Binary encoding, if for example $x = 1$, $|VS_1|$=3C_1, where C refers to the combination function. Note that Gray encoding has the same weakness as binary encoding.



(a) (b)

Fig. 1: LFI model for binary and one-hot encoding respectively (a)-(b). The red bits represent the state FF positions for LFI; the edge values represent HD from the sensitive state.

In one-hot encoding, states have maximum HD of 2 from each other. Hence, if $x = 1$, the FSM is not vulnerable as $|VS_1| = 0$. However, if $x > 1$, every state becomes reachable from every other state, $|VS_x| = |S|$. For example, in Fig. 1(b), the attacker can access $SS = 00001$, from "00100" by overturning FF0 and FF2 through LFI.

C. Related Work

There is considerable work on fault-tolerant FSMs for different purposes. The main assumption for self-checking circuits is that the faults occur randomly in a unidirectional way (e.g., a flash memory cell is more susceptible to set to zero than one). Hence, unidirectional error correction codes, such as m-out-of-n codes have been proposed as solutions [2]. However, with the deployment of Random Access Memory, especially in aerospace applications, bidirectional faults caused by cosmic rays have been considered a more realistic scenario. Therefore, the linear error detection/correction techniques, such as cyclic redundancy check codes [13] have been proposed. Other techniques, such as the popular triple modular redundancy, address many hardware implementations. However, the fault model is still assumed to be random single bit upsets. Hence, this is not suitable for a strong adversary who can precisely and concurrently overturn multiple bit flips.

The main disadvantage of linear codes is that they can only guarantee the detection of all errors with a multiplicity less than d, i.e., the minimum HD of the code. As a result, non-linear codes have been proposed to improve the detection capability [14] [15] [16]. However, for an attacker, who is knowledgeable of the precise FSM states and its encoding, an undetectable fault pattern can be calculated and inserted into the FSM. Hence the entire validity of this scheme banks on the attacker not being able to anticipate the next FSM states in the same clock cycle as the fault injection happens [17]. This premise may not always hold true as an adversary can make smart FSM predictions. For instance, in AES controller an adversary can predict the Initial and Final round by detecting

when the plaintext is inputted in the encryption module and when the ciphertext output is engendered [1]. Although some of these non-linear codes are capable of detecting all possible errors with a probability close to 1, they impose very large overhead to the original FSM in terms of the number of redundant flip-flops [15].

Hence, these techniques are not applicable due to their weaker fault models and large overhead (i.e., large number of redundant flip-flops and extra detection/correction circuitry).

IV. Proposed Encoding Schemes

A. Conservative Approaches

In this approach, the encoding requires $HD > x$ between all states. Specifically, all FSM states are considered sensitive states, and thus, trivially results in $|VS_x| = 0$. There are several ways to accomplish this. A *naïve* way, which is an adaptation of the "repetition" error-detection code, is as follows. From Section III-B, we know that one-hot encoding can resist $x = 1$ fault because the HD between all states is 2. By doubling the number of flip flops, the HD can be also doubled as each state is encoded with an additional '1'. For example, the one-hot encoding for an FSM with 4 states is $\{0001, 0010, 0100, 1000\}$. With naïve extension, the new encoding would be $\{00000011, 00001100, 00110000, 11000000\}$ where the HD between all states is now 4. The latter encoding can, thus, resist $x < 4$ faults. While this approach is simple and easy to scale to any x, its shortcoming is the increase in the number of FFs by a factor of 2 to resist 2 additional faults.

An alternative approach is to use perfect code schemes that provide necessary HD ($d > x$). Options include Hamming codes, Golay codes, etc. For perfect codes, all states are still treated equally so the overhead is expected to be high. Moreover, perfect codes exist only for specific parameters, and therefore, they might not match any arbitrary FSM design. It is noteworthy that codes such as Reed-Solomon(RS) code can achieve rates higher than 0.9. However, they behave poorly in terms of overhead unless the FSM size is unrealistically large, e.g. RS codes for CDs can be 256 bits. Hence, while these codes are efficient for transferring massive data through communication channels, they are not well suited for FSMs, since the number of available bits in the hardware is much less than that of a message in a communication channel.

B. Pragmatic Approach (PATRON)

The pragmatic approach is more efficient and flexible for achieving $|VS_x| = 0$ by ignoring the HD between NSs and only focusing on the HD between all states and states in SS. Thus, it minimizes n. Faults that result in transitions between NSs are permitted. If the attacker has x laser capability, all the SSs in the FSM need to be a minimum HD of $x+1$ away from all the NSs. Moreover, the SS elements need to be a HD of a multiple of $x+1$ away from each other; this ensures that the attacker cannot access and transition between any of these SSs using LFI.

From the example shown in Fig. 2, assume that the attacker can flip 1 FF (i.e., $x = 1$) and the FSM requires 4

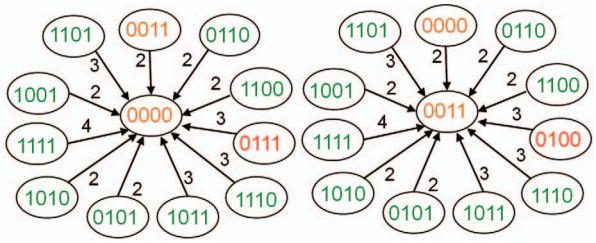


Fig. 2: Pragmatic approach illustrated for FSM with $n = 4$ states, $SS = \{0000, 0011\}$, and $x = 1$: (left) states with $HD \geq 2$ for $\{0000\}$ and (right) states with $HD \geq 2$ for $\{0011\}$. Orange denotes SS. Green denotes NS candidates with $HD \geq 2$ and red denotes states that have $HD \leq 1$ for at least one of the SSs (i.e., $\notin NS$).

NSs and 2 SSs. When $n = 4$ and $SS = \{0000, 0011\}$, there can be as many as 8 NSs in the fault tolerant FSM. Specifically, the set of NS with $HD > 1$ from states in SS consists of $\{0101, 0110, 1001, 1010, 1100, 1101, 1110, 1111\}$. Although the states $\{1101, 1110\}$ are only a HD of 1 away from the state $\{1100\}$ and the attacker can inject a fault to transfer between them, this poses no threat to the states in SS, and thus, can be considered secure. This is the major difference from the conservative approaches.

For a given x and n, in order to determine the total number of sensitive states the following two equations are used:

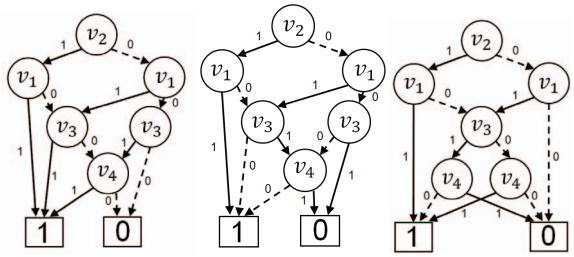
$$|SS| = \begin{cases} \sum_{h=0}^{n} n\, C_{h(x+1)}, & \text{if } x = 1 \\ \lfloor 2^{\,n/(x+1)} \rfloor, & \text{if } x > 1 \end{cases} \tag{6}$$

where $h \in \mathbb{Z}$, $h(x+1) \leq n$, and C refers to the combination function. For $n = 4$ and $x = 1$, $|SS| = 4C_0 + 4C_2 + 4C_4$. When $x > 1$, for $n = 4$ and $x = 2$, we will have $\lfloor 2^{4/3} \rfloor = 2$.

As inferred from the above discussion, given certain constraints in terms of $|SS|$, PATRON outstretches n for accommodating $x+1$, to fulfill the requirement of $|NS|$. Given SS and x as inputs, we find encoding for NS by solving a combinatorial problem. Each state in NS is represented by a vector of length n: $[v_1, v_2, \ldots v_n], v_i\{0,1\}\forall i$, where v_i represent the variable associated with the ith FF in the FSM. A Boolean function over these n variables should yield logic 1 for states in NS when HD is large enough. The general formula to calculate all the NSs, where $SS = \{ss_1, ss_2, \ldots, ss_m\}$ is

$$NS = \{z \in (0,1)^n : HD(z, ss_1) > x \cap HD(z, ss_2) > x \cap \ldots HD(z, ss_m) > x\} \tag{7}$$

For certain FSMs, exploring relatively higher n might be necessary to meet the constraints in which case these boolean expressions might have scalability issues. Therefore, we propose a BDD data structure to represent each of these Boolean functions that correspond to HD calculations with respect to each state in SS shown in Equation 7. After obtaining the BDDs for each of the SS, the BDD data structures are ANDed together, in order to get the intersected sets of combinations, NS. Fig. 3 depicts the individual steps-BDDs representing the Boolean functions for each SS (Fig. 3 (a) and Fig. 3 (b)



(a) $ss_1 = 0000$ (b) $ss_2 = 0011$ (c) NS

Fig. 3: (a-b)BDDs representing Boolean functions for each SS (c) BDD representing NS.

and the subsequent BDD representing the NS, obtained after ANDing the BDDs in Fig. 3 (c).

C. Comparison of Approaches

We propose CR for overhead comparison and introduce the *vulnerability metric*. In Equation (2), CR represents the percentage of useful data bits. We extend this definition of $0 < CR \leq 1$ to additional FF consumption associated with protecting the FSM[1]. In our case, k is the number of state FFs in the original unprotected FSM encoded in binary and n represents the total fault tolerant state FFs. When CR is closer to 1 (0), the number of extra FFs needed is low (high) and the overhead is, therefore, low (high). For example, if a designer wants 6 states altogether with 4 NSs that are a $HD = 2$ away from 2 SSs, the original FSM can be encoded with $k = 3$ (2^3 states), since it consists of 6 states. However, with the HD requirement, the pragmatic approach demands that we increase to $n = 4$ (2^4 states). Hence, $CR = \frac{3}{4} = 0.75$.

For binary encoding, $CR = 1$, but the FSM is susceptible to fault attacks. To capture this, we introduce vulnerability metric (VM) for measuring the FSM's degree of susceptibility to x laser-based faults in one clock cycle,

$$VM(x) = \frac{|VS_x|}{|S|}. \tag{8}$$

Intuitively, VM is the percentage of states where x faults can lead to a sensitive state (e.g., consider $x = 2$ for a binary FSM design consisting of all 2^3 states, $VM(2) = \frac{3C_1+3C_2}{8}$).

Hence if the designer is confident about the SSs, then PATRON is recommended. In situations where the SSs are difficult to be correctly identified, the conservative approach for appropriate x is recommended as it protects *all* states equally. While it requires larger overheads, it will provide an LFI-tolerant-FSM. However, with FSM designs requiring increasing x resistance, all coding approaches start to show limitations as shown in the next section. Hence, for high x resistant designs, only naïve may be readily applicable.

V. Results and Discussion

The proposed approaches are investigated on five controller benchmark circuits, namely AES, SHA, MIPS Processor, Memory Controller and RSA in terms of CR and VM with increasing x. Pertinent discussion on error detection-based approaches, non-linear codes and their inadequacy to LFI underscores the efficiency of PATRON. All benchmark circuits are collected from OpenCores and synthesized using Synopsys Design Compiler with 65-nm library from TSMC and their Power Delay Products (PDPs) are compared. The controller circuit of AES, SHA, and RSA contain FSMs with less than 8 states. For AES, the states "Do Round" and "Final Round" and for SHA-256 the states "Data input", "Block next", and "Valid" are in SS [1]. For RSA among the seven states, "Result", "Square", "Multiply" and "Load2" are regarded as in SS. Note that the proposed approach has the flexibility to increase the $|SS|$ to $|S|$ for all these benchmarks, if intended by the designer. Since most cryptographic algorithms tend to have small number of states (based on designs at OpenCores), we also demonstrate the scalability of PATRON on large controller circuits like memory controller FSM with 66 states.

TABLE I: Code Rate (CR) and Vulnerability Metric (VM) analysis for different encoding schemes; Hamming represents Hamming (7,4). Red and green color denote vulnerable and non-vulnerable FSMs, respectively.

		Binary		One-hot		Hamming		Naïve		Pragmatic	
Benchmark	x	CR	VM	CR	VM	CR	VM	CR	VM	CR	VM
AES	1	1	0.6	0.6	0	0.57	0	0.6	0	0.8	0
$\|SS\|$=2	2	1	1	0.6	1	0.57	0	0.3	0	0.6	0
$\|NS\|$=3	3	1	1	0.6	1	0.57	1	0.3	0	0.5	0
Average		1	0.86	0.6	0.66	0.57	0.33	0.4	0	0.63	0
SHA-256	1	1	0.4	0.4	0	0.57	0	0.4	0	0.8	0
$\|SS\|$=3	2	1	0.9	0.4	1	0.57	0	0.2	0	0.5	0
$\|NS\|$=4	3	1	1	0.4	1	0.57	1	0.2	0	0.4	0
Average		1	0.76	0.4	0.66	0.57	0.33	0.26	0	0.56	0
RSA	1	1	0.4	0.4	0	0.57	0	0.4	0	0.8	0
$\|SS\|$=4	2	1	0.9	0.4	1	0.57	0	0.2	0	0.4	0
$\|NS\|$=3	3	1	1	0.4	1	0.57	1	0.2	0	0.4	0
Average		1	0.76	0.4	0.66	0.57	0.33	0.26	0	0.53	0
MIPS processor	1	1	0.2	0.1	0	-	-	0.1	0	0.5	0
$\|SS\|$=5	2	1	0.8	0.1	1	-	-	0.08	0	0.3	0
$\|NS\|$=14	3	1	1	0.1	1	-	-	0.08	0	0.3	0
Average		1	0.66	0.1	0.66	-	-	0.09	0	0.36	0
Mem. Controller	1	1	0.1	0.04	0	-	-	0.04	0	0.3	0
$\|SS\|$=8	2	1	0.4	0.04	1	-	-	0.02	0	0.2	0
$\|NS\|$=58	3	1	0.9	0.04	1	-	-	0.02	0	0.1	0
Average		1	0.46	0.04	0.66	-	-	0.03	0	0.2	0

Table I shows a comprehensive demonstration of PATRON outperforming conservative approaches in every aspect. Of the conservative approaches, except the naïve approach, all schemes deliver encoding that is limited to a certain HD boundary. For example, Hamming (7,4) cannot generate any $|VS_x| = 0$ encoding beyond $HD > 3$; for $HD \leq 3$, CR is comparatively lower than pragmatic. Another limitation is that the $|S|$ required for MIPS Processor and Memory Controller exceed the number of codewords in Hamming (7,4). For one-hot, once the attacker can control $x = 2$, $VM(2)$ becomes 1 and all the encodings turn unsafe; hence, its CR becomes irrelevant. Similarly for binary encoding, all encodings have $VM(x) > 0$, indicating susceptibility to LFI, rendering the

[1]Note that an FSM includes both state FFs and decoding logic. CR captures worst case in decoding logic assuming that the increase is roughly in proportion to number of FFs

TABLE II: Power Delay Product (PDP) Normalized with PDP of Binary encoding; O=One-hot, H=Hamming(7,4), N=Naïve, P=Pragmatic.

	AES			SHA			RSA			MIPS			Mem. Cont.		
x	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
O	2			1.4			2.6			2.9			0.9		
H	2.5			2.1			4.5			-			-		
N	1.8	2.1	2.1	2.1	2.6	2.6	3.4	3.8	3.8	4.2	5	5	1	1.2	1.2
P	1.4	1.9	2.1	1.1	1.6	1.8	1.5	1.7	2	1.5	1.8	2.4	1.1	1.1	1.2

high CRs futile. With large benchmarks (MIPS and Mem. Controller), PATRON performs even better compared to naïve with an average improvement of 366% in CR than smaller benchmarks (87%). The reason is that pragmatic utilizes the don't-care states whereas naïve's treatment of all states as SS is costlier. Note that along with Hamming (7,4), the applicability for Extended Hamming, Perfect Binary Golay and Extended Binary Golay were also evaluated but not tabulated as they share similar limitations as Hamming (7,4) i.e., in terms of limited codewords availability and low CR. From the table, the following conclusions can be confirmed:

- For designs requiring high x resilience to LFI, only pragmatic approach is projected to achieve the highest CR and with $VM(x) = 0$ (i.e, no vulnerable states). PATRON has average improvement of 54% in CR compared to naïve.
- VM takes precedence before CR as the state encoding should have $VM(x) = 0$ first, after which CR comparison becomes relevant.

With regard to nonlinear codes, robust and partially robust codes are seen as promising [14]. However, these codes both have minimum HD of 1. This trait alone makes them inapplicable for LFI where HD flexibility is of paramount importance. Most of these codes have relatively low CR too [15]. The CRs for only Quadratic Sum code and Punctured Cubic Code measure up to 0.5 [16].

Table II shows post synthesis results indicating PDP normalized by the PDP of Binary encoding. Area and power are highly correlated, so PDP is a comprehensive metric for comparison in this case. The table validates that pragmatic approach has much lower PDP on average than the alternatives only varying from 10% to 240% from binary encoding; hence, PATRON manifests the least overhead. One-hot for Mem. Controller has PDP< 1 because of simpler decoding logic. Comparing both the tables, CR is confirmed to be the worst case estimate; in reality, the overhead is much less, e.g., for RSA with increasing x, CR is halved for PATRON whereas PDP only increases 1.3 times.

VI. Conclusion and Future Work

In this paper, we proposed and compared fully secure LFI-tolerant-FSM schemes. Due to the lack of a well-accepted threat model for LFI, we defined a general strong threat model that is extendable to other weaker models. We demonstrated how the existing encoding approaches fail to meet this particular threat model, especially as the attacker's LFI capabilities improve, which necessitates our low overhead PATRON. Our proposed decision diagram based scheme allows the designer to quantitatively compare the degree of vulnerability using a proposed metric. If vulnerabilities exist in the design then appropriate measures with pragmatic or conservative approaches (depending on the FSM knowledge of the designer) can be applied to obtain an LFI-resistant FSM. For future work, we plan to demonstrate LFI on protected/unprotected FSMs on FPGAs along with investigating LFI effects on layout.

References

[1] A. Nahiyan, F. Farahmandi, P. Mishra, D. Forte, and M. Tehranipoor. "Security-aware FSM design flow for identifying and mitigating vulnerabilities to fault attacks." IEEE Transactions on Computer-aided design of integrated circuits and systems 38, no. 6 (2018): 1003-1016.

[2] D. Anderson, and G. Metze. "Design of totally self-checking check circuits for m-out-of-n codes." IEEE Transactions on Computers 100, no. 3 (1973): 263-269.

[3] K. Akdemir, G. Hammouri, and B. Sunar. "Non-linear error detection for finite state machines." In International Workshop on Information Security Applications, pp. 226-238. Springer, Berlin, Heidelberg, 2009.

[4] V. Tomashevich, Y. Neumeier, R. Kumar, O. Keren, and I. Polian. "Protecting cryptographic hardware against malicious attacks by nonlinear robust codes." In 2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 40-45. IEEE, 2014.

[5] S. Skorobogatov, and R. Anderson. "Optical fault induction attacks." In International workshop on cryptographic hardware and embedded systems, pp. 2-12. Springer, Berlin, Heidelberg, 2002.

[6] R. Leveugle, et al. "Laser-induced fault effects in security-dedicated circuits." In 2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1-6. IEEE, 2014.

[7] C. Roscian, J. Dutertre, and A. Tria. "Frontside laser fault injection on cryptosystems-Application to the AES'last round." In 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 119-124. IEEE, 2013.

[8] F. Courbon, P. Moundi, J. Fournier, and A. Tria. "Adjusting laser injections for fully controlled faults." In International workshop on constructive side-channel analysis and secure design, pp. 229-242. Springer, Cham, 2014.

[9] Agoyan, M., J. M. Dutertre, A. P. Mirbaha, and A. Tria. "How to Flip a Bit?, On-Line Testing Symposium (IOLTS)." (2010).

[10] A. Nahiyan, K. Xiao, K. Yang, Y. Jin, D. Forte, and M. Tehranipoor. "AVFSM: a framework for identifying and mitigating vulnerabilities in FSMs." In Proceedings of the 53rd Annual Design Automation Conference, pp. 1-6. 2016.

[11] V. Rathor, B. Garg, and G. Sharma. "New Lightweight Architectures for Secure FSM Design to Thwart Fault Injection and Trojan Attacks." Journal of Electronic Testing 34, no. 6 (2018): 697-708.

[12] A. Papadimitriou, D. Hély, V. Beroulle, P. Maistri, and R. Leveugle. "A multiple fault injection methodology based on cone partitioning towards RTL modeling of laser attacks." In 2014 Design, Automation Test in Europe Conference & Exhibition (DATE), pp. 1-4. IEEE, 2014.

[13] W. Peterson, and D. Brown. "Cyclic codes for error detection." Proceedings of the IRE 49, no. 1 (1961): 228-235.

[14] M. Karpovsky, and A. Taubin. "New class of nonlinear systematic error detecting codes." IEEE Transactions on Information Theory 50, no. 8 (2004): 1818-1819.

[15] K. Kulikowski, Z. Wang, and M. Karpovsky. "Comparative analysis of robust fault attack resistant architectures for public and private cryptosystems." In 2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 41-50. IEEE, 2008.

[16] Y. Neumeier, and O. Keren. "Punctured Karpovsky-Taubin binary robust error detecting codes for cryptographic devices." In 2012 IEEE 18th International On-Line Testing Symposium (IOLTS), pp. 156-161. IEEE, 2012.

[17] B. Sunar, G. Gaubatz, and E. Savas. "Sequential circuit design for embedded cryptographic applications resilient to adversarial faults." IEEE Transactions on Computers 57, no. 1 (2007): 126-138.