Balancing Exploration and Exploitation: Discriminative-Model-based Fault Injection Parameter Search

Ju-Hwan Kim and Dong-Guk Han

Kookmin University, Seoul, Republic of Korea

Abstract. Fault Injection (FI) attacks are physical attacks that deliberately induce malfunctions in a target device. Inducing the intended fault requires precise tuning of the FI parameters, for which automated search strategies have been proposed. Most favor either exploration or exploitation: the former covers a broad region of the parameter space, while the latter discovers many successful parameters. An effective search, however, should find parameters that are both numerous and broadly spread. We propose a search framework that achieves this with two complementary discriminative models: an exploration model that learns the boundary where faults are likely, and an exploitation model that concentrates on the neighborhoods of discovered faults. We further introduce the Refinement-Oriented Successive Halving Algorithm (RSHA), which selects a highly reliable parameter from the discovered ones, using an evaluation budget derived from a target estimation precision. Across four scenarios—voltage glitching and electromagnetic fault injection on two off-the-shelf microcontrollers, the 8-bit ATXmega and the 32-bit SAM4S—our method finds up to 26.7× as many distinct successful parameters as state-of-the-art methods and attains up to 5.8× their parameter space coverage. At the full budget of 100,000 injections, the parameter that RSHA selects from our discovered set achieves a success rate up to 7.6 percentage points higher than the one it selects from the best baseline's set.

Keywords:

1 Introduction

Fault Injection (FI) attacks disrupt the nominal operation of a target device to induce exploitable behavior. Adversaries typically achieve this by manipulating the supply voltage or clock signal [KK99], or by exposing the device to electromagnetic (EM) [DDRT12] or laser pulses [SA03]. Under such perturbations, a device may exhibit data corruption [PQ03], instruction replacement [GZW+25], or instruction skipping [AK97]. These faults can be exploited to extract secret keys from cryptographic implementations [BS97] or to bypass authentication mechanisms [VOGT21]. A large body of work studies how to exploit faults under an assumed fault model, i.e., assuming that a specific fault has already been induced [SAW25, KGAS25, WMT25, ZZZ+26, HHH+26, JDGJ26, JD26]. By contrast, methods for reliably inducing the fault required by the attack, hereafter referred to as the `intended fault`, have received comparatively little attention.

To induce the `intended fault` reliably, the adversary must precisely calibrate the FI parameters. For instance, the success rate of an Electromagnetic Fault Injection (EMFI) attack strongly depends on the spatial coordinates (x, y) of the EM probe, the pulse voltage, and the injection timing. Since the size of the parameter space grows exponentially

E-mail: zzzz2605@kookmin.ac.kr (Ju-Hwan Kim), christa@kookmin.ac.kr (Dong-Guk Han)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2026-10-20.

with the number of parameters, and high-resolution equipment further enlarges each dimension, exhaustive search is infeasible: even when the injection timing is excluded, the parameter space of our EMFI setup comprises approximately 3.1×10^{13} distinct combinations. Random search and grid search are therefore widely used in practice, but they are inefficient at localizing promising parameter regions [CPB+13]. This inefficiency has motivated the development of more advanced FI parameter search strategies.

1.1 Related Work

Carpi et al. [CPB+13] pioneered automated FI parameter search by proposing three methods: FastBoxing, Adaptive Zoom & Bound (AZB), and a Genetic Algorithm (GA)-based approach. Their framework is predicated on the *boundary hypothesis*, which posits that the parameters inducing an `intended fault` lie in a transition zone where the perturbation is neither weak enough to preserve `normal` behavior nor strong enough to induce a `mute` state, i.e., a system halt. AZB iteratively bisects the intervals between neighboring samples of distinct verdict classes to progressively localize this boundary. Although AZB achieved the highest success rate among their three methods, the authors identified GA as a promising direction, owing to its potential for improvement through hyperparameter tuning.

Subsequent research has focused predominantly on refining the GA [PBJC14, MSPB18, BFP19]. These approaches evolve a population of parameters over successive generations using operators tailored to the FI domain. For instance, the crossover operator generates offspring by interpolating between parents of different verdict classes, thereby steering the population toward the `normal`–`mute` boundary [PBJC14]. The fitness function used for parent selection prioritizes individuals that induce an `intended fault`; to account for the non-deterministic nature of faults near the boundary, it also rewards `changing` behavior, in which a fixed parameter yields varying verdicts across repeated trials. The Memetic Algorithm (MA), which integrates local search and tabu search into the GA, has likewise been adopted to strengthen local refinement [PBBJ15]. Krček et al. showed that initialization techniques that distribute the population more uniformly across the parameter space reduce the variance of the results across runs, relative to random initialization [KFP21]. In follow-up work, Krček et al. used decision tree models to extract prior knowledge from earlier FI campaigns and guide population initialization [KOFP22]. A characteristic common to the GA and MA is their exploitative nature: once an `intended fault` is found, they retain the corresponding parameter with high probability and generate offspring in its immediate vicinity, which may confine the search to a narrow region. To address this, Krček et al. [KO24] proposed GridMA, which partitions the target area into a grid and executes the MA independently within each sector.

More recently, Machine Learning (ML) techniques have been adopted to improve search efficiency. Wu et al. [WRBP20] proposed an ML-based method for rapidly characterizing the parameter space. They traced sensitivity curves—contours along which the ratio of abnormal responses is constant—and used a Multilayer Perceptron (MLP) to reconstruct the full sensitivity map from a limited number of measurements. Their method characterizes the sensitivity of the target rather than directly searching for parameters that induce an `intended fault`. Werner et al. [WMP21] framed FI parameter search as an equipment calibration problem and applied two hyperparameter optimization techniques, namely the Successive Halving Algorithm (SHA) and Sequential Model-based Algorithm Configuration (SMAC). SHA selects a well-performing parameter from a randomly initialized candidate set: it iteratively discards the lower-performing half while doubling the budget allocated to the surviving candidates. SMAC, in contrast, fits a Random Forest surrogate model to predict the success rate of unevaluated parameters and selects subsequent candidates through an acquisition function. However, this surrogate model is uninformative until a sufficient number of `intended fault` parameters have been observed. Werner et al.

addressed this with an initialization phase in which random search continues until at least one `intended fault` parameter is found, after which additional initial candidates are generated by Gaussian mutation. This phase may consume a large portion of the budget when `intended faults` are rare on the target. A generative model, namely a Conditional Generative Adversarial Network (CGAN), has also been used to synthesize promising candidates [KBV24]. Köylü et al. trained a CGAN on `normal` and `mute` samples collected via a budget-constrained grid search. Once trained, the generator is conditioned on an intermediate label to synthesize candidates along the inter-class boundary, and an auxiliary classifier selects those closest to the boundary for injection, in line with the boundary hypothesis. A notable advantage of this approach is that it can identify promising parameters even when no `intended fault` sample is present in the training data.

These FI parameter search strategies strike different balances in the exploration–exploitation trade-off. Exploration-biased methods, such as Random Search (RS) and AZB, evaluate new candidates without explicitly leveraging the distribution of the `intended fault` parameters discovered so far. Exploitation-biased methods, such as GA, MA, and CGAN, concentrate the search on the neighborhoods of these parameters. CGAN initially targets the `normal`–`mute` boundary, but once `intended faults` are observed, it is conditioned to generate candidates that resemble them, thereby shifting its focus toward exploitation. Exploration-biased methods cover a broad region of the parameter space, but this breadth comes at the cost of discovering relatively few `intended fault` parameters. Exploitation-biased methods, conversely, can discover many `intended fault` parameters, but their search tends to remain confined to a narrow region.

The appropriate choice depends on the objective of the search, which is broadly one of two. The first is to discover as many `intended fault` parameters as possible within a limited budget, a goal for which exploitation-biased methods are well suited. The second is to discover `intended fault` parameters spread across a broad region of the parameter space. An adversary seeking the parameter with the highest attack success rate, or a security evaluator assessing the resistance of a target to FI attacks comprehensively, should identify many `intended fault` parameters distributed across the space rather than clustered in a single region. Given a sufficiently large budget, exploration-biased methods can meet this second objective, as they eventually discover enough parameters across the space. In most practical settings, however, neither exploration-biased nor exploitation-biased methods meet this requirement on their own; the second objective therefore calls for a strategy that combines broad exploration with focused exploitation. SMAC and GridMA aim to strike such a balance, but each has a limitation. SMAC balances the two through its acquisition function, yet its initialization can be costly when `intended faults` are rare. GridMA allocates the budget uniformly across grid sectors, so part of it may be spent on sectors unlikely to contain any `intended fault` parameter.

1.2 Our Contributions

Our goal is to discover a diverse set of `intended fault` parameters within a limited budget. To this end, we propose a parameter search framework built on two specialized discriminative models, one for exploration and one for exploitation. We further propose RSHA, which selects a highly reliable parameter from the discovered ones. Our contributions are summarized as follows:

- **Discriminative-model-based parameter generation framework.** We leverage a distinctive property of the FI parameter space to turn discriminative models, which are easier to train than generative ones, into generators of promising parameters. The framework employs two complementary models: an *exploration model* that approximates the boundary between the `normal` and `mute` regions under the bound-

ary hypothesis, trained without any `intended fault` sample; and an *exploitation model* that generates parameters in the vicinity of the discovered `intended fault` parameters. The two models operate in parallel, thereby balancing exploration and exploitation.

- **Refinement-Oriented Successive Halving Algorithm (RSHA).** RSHA is an SHA-based post-search procedure. Unlike the prior use of SHA, which calibrates a randomly initialized set of parameters, RSHA refines a candidate set in which every parameter has already induced the `intended fault` at least once. It derives the injection budget from a target precision for the success-rate estimates, thereby identifying a reliable parameter at low cost.
- **Extensive validation in diverse attack settings.** We validate our method across four scenarios spanning two fault sources (voltage glitching and EMFI) and two microcontrollers (8-bit and 32-bit). Our method identifies up to 26.7× as many unique `intended fault` parameters as state-of-the-art methods and attains up to 5.8× their parameter space coverage. At the full budget of 100,000 injections, the parameter that RSHA selects from our discovered set achieves a success rate up to 7.6 percentage points higher than the one it selects from the best baseline's set.

2 Preliminaries

2.1 Fault Injection Attacks

FI attacks intentionally perturb a target device to induce exploitable malfunctions. These malfunctions typically manifest as data corruption or modification of the program flow. To induce such faults, the attacker must calibrate the FI parameters.

Fault Parameters. We categorize the FI parameters into two classes according to their adjustability:

- **Hardware parameters:** Static physical configurations, such as the wire length for Voltage Glitching (VG), or the probe diameter and coil winding direction for EMFI. These are generally fixed and ill-suited to algorithmic optimization.
- **Software parameters:** Dynamic, programmable variables. For VG, these include the injection timing, the glitch width, and the repeat count; for EMFI, the injection timing, the spatial coordinates (x, y), and the pulse voltage.

This work focuses on the automated optimization of the software parameters.

Fault Verdict Classes. The outcome of a fault injection is classified into one of the following verdict classes:

- `normal`: The device maintains nominal operation despite the perturbation.
- `mute`: The device halts, resets, or becomes unresponsive.
- `intended fault`: The desired fault occurs (e.g., skipping the target instruction).
- `inconclusive`: Any anomalous behavior that does not fall into the preceding three classes (e.g., corrupting the wrong instruction).

In addition, some prior studies [CPB+13, PBJC14, PBBJ15] introduce a `changing` verdict for parameters that yield non-deterministic outcomes across repeated trials under identical settings.

Boundary Hypothesis. As introduced in Section 1.1, many FI parameter search methods [CPB+13, PBJC14, PBBJ15, MSPB18, KBV24] build on the *boundary hypothesis*, which states that `intended faults` concentrate in the transition zone between the `normal` and `mute` regions. A `normal` verdict arises when the pulse voltage is too low to disturb the computation, or when the probe is placed over a region unrelated to the target operation. A `mute` verdict arises when the voltage is too high, or when the probe is placed over a component whose disruption halts execution, such as the arithmetic logic unit, the register file, or the clock circuitry. An `intended fault` is thus most likely to arise in between, where the perturbation is strong enough to alter the computation but not so strong as to halt it. Prior methods are designed to explore this `normal`–`mute` boundary efficiently.

2.2 Discriminative and Generative Models

In supervised learning, the goal is to learn a mapping from input features $x \in \mathcal{X}$ to labels $y \in \mathcal{Y}$. From a probabilistic perspective, the models used for this purpose are broadly categorized as discriminative or generative.

A discriminative model estimates the conditional probability $\Pr(Y \mid X)$, directly modeling the decision boundary between classes in the feature space. A generative model, in contrast, learns the joint distribution $\Pr(X, Y)$ or the class-conditional density $\Pr(X \mid Y)$; CGAN, for example, models $\Pr(X \mid Y)$ to synthesize new samples for a target class. Generative models, however, are often harder to learn effectively. Classical generative models such as Naive Bayes often converge to a higher asymptotic error than their discriminative counterparts [NJ01], and modern deep generative models such as GANs are prone to training instability and mode collapse—a failure mode in which the generator produces samples of low diversity and fails to capture the full variability of the underlying data distribution [GPM+14, SC22].

The relationship between the two paradigms is governed by Bayes' rule:

$$\Pr(Y \mid X) = \frac{\Pr(X \mid Y)\Pr(Y)}{\Pr(X)}.$$

If the joint distribution $\Pr(X, Y)$ is known, the conditional $\Pr(Y \mid X)$ can be obtained by marginalizing over $\mathcal{Y}$; a generative model can therefore act as a discriminative one. The converse, however, generally does not hold, because estimating the marginal likelihood $\Pr(X)$ for real-world data is typically intractable, as reflected in the manifold hypothesis [BCV13]. As an illustration, consider the MNIST dataset of 28×28 grayscale images of handwritten digits. Although the feature space is enormous ($256^{28\times 28} \approx 10^{1888}$ possible images), valid images lie on a lower-dimensional manifold that occupies a negligible fraction of this space. Because of this difficulty in characterizing $\Pr(X)$, a discriminative model generally cannot serve as a generative one.

3 Methodology

3.1 Theoretical Basis

As discussed in Section 2.2, a discriminative model generally cannot serve as a generative model, because the marginal distribution $\Pr(X)$ is intractable to estimate. The FI parameter space, however, has a distinctive property that we exploit to turn a discriminative model into a generator. Let $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ be the domains of the individual dimensions; in EMFI, for example, these are the pulse voltage, the timing delay, and the spatial coordinates. The complete parameter space is their Cartesian product, $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 \times \cdots \times \mathcal{P}_n$. Unlike natural data, whose valid samples occupy a negligible-volume manifold within the ambient space, every vector $p \in \mathcal{P}$ is a valid FI parameter. Because the set of valid parameters

coincides with the entire Cartesian product, no region is a priori more plausible than another, and we therefore adopt a uniform prior over $\mathcal{P}$, i.e., $\Pr(P = p) = 1/|\mathcal{P}|$ for all $p \in \mathcal{P}$. Under this uniform prior, the likelihood and the posterior are proportional, as stated in Property 1.

Property 1. Let P and V denote the random variables for the FI parameter and the verdict, respectively. For any fixed verdict v, the posterior $\Pr(P = p \mid V = v)$ is proportional, as a function of p, to the likelihood $\Pr(V = v \mid P = p)$.

Proof. By Bayes' theorem,

$$\Pr(P = p \mid V = v) = \frac{\Pr(V = v \mid P = p)\Pr(P = p)}{\Pr(V = v)}.$$

Since $\Pr(P = p) = 1/|\mathcal{P}|$ is constant in p and $\Pr(V = v)$ does not depend on p, we have

$$\Pr(P = p \mid V = v) = \alpha \cdot \Pr(V = v \mid P = p), \qquad \alpha = \frac{1}{|\mathcal{P}| \cdot \Pr(V = v)},$$

where α is constant in p. □

For a parameter p, a discriminative classifier m outputs a probability vector $m(p)$ over the verdict classes. We write $m(p)_v$ for the component corresponding to verdict v, which approximates the likelihood $\Pr(V = v \mid P = p)$. By Property 1, for a fixed verdict v, its output is proportional, as a function of p, to the posterior $\Pr(P = p \mid V = v)$. The model thus acts as an unnormalized score for the posterior, which we use to generate parameters that are likely to induce the `intended fault` via rejection sampling.

Rejection sampling is a standard technique for drawing samples from a target distribution that is known only up to a normalizing constant, using a simpler proposal distribution [Bis07]. Let v_{int} denote the `intended fault` verdict, so that the target distribution is the posterior $\Pr(P \mid V = v_{\text{int}})$. Given a proposal distribution Q over $\mathcal{P}$ with probability mass $\Pr(Q = p)$, the standard procedure is as follows:

1. Choose a constant M such that $\Pr(P = p \mid V = v_{\text{int}}) \leq M \cdot \Pr(Q = p)$ for all $p \in \mathcal{P}$.
2. Draw a sample p from Q and a sample u from the uniform distribution $\mathcal{U}(0, 1)$.
3. Accept p if the following criterion holds; otherwise, reject it:

$$u < \frac{\Pr(P = p \mid V = v_{\text{int}})}{M \cdot \Pr(Q = p)}. \tag{1}$$

Each accepted sample is an exact draw from the target posterior. With a uniform proposal, this criterion simplifies to a single condition, stated in Property 2.

Property 2. If the proposal distribution is uniform over $\mathcal{P}$, the acceptance criterion in Equation (1) simplifies to $u < m(p)_{v_{\text{int}}}$.

Proof. By Property 1, the target posterior is proportional to the likelihood,

$$\Pr(P = p \mid V = v_{\text{int}}) = \alpha \cdot \Pr(V = v_{\text{int}} \mid P = p),$$

for the constant α of Property 1. Take the uniform proposal $\Pr(Q = p) = 1/|\mathcal{P}|$ and the constant $M = \alpha \cdot |\mathcal{P}|$. The envelope condition in Step 1 then holds, since $\Pr(V = v_{\text{int}} \mid P = p) \leq 1$ gives

$$\Pr(P = p \mid V = v_{\text{int}}) = \alpha \cdot \Pr(V = v_{\text{int}} \mid P = p) \leq \alpha = M \cdot \Pr(Q = p).$$

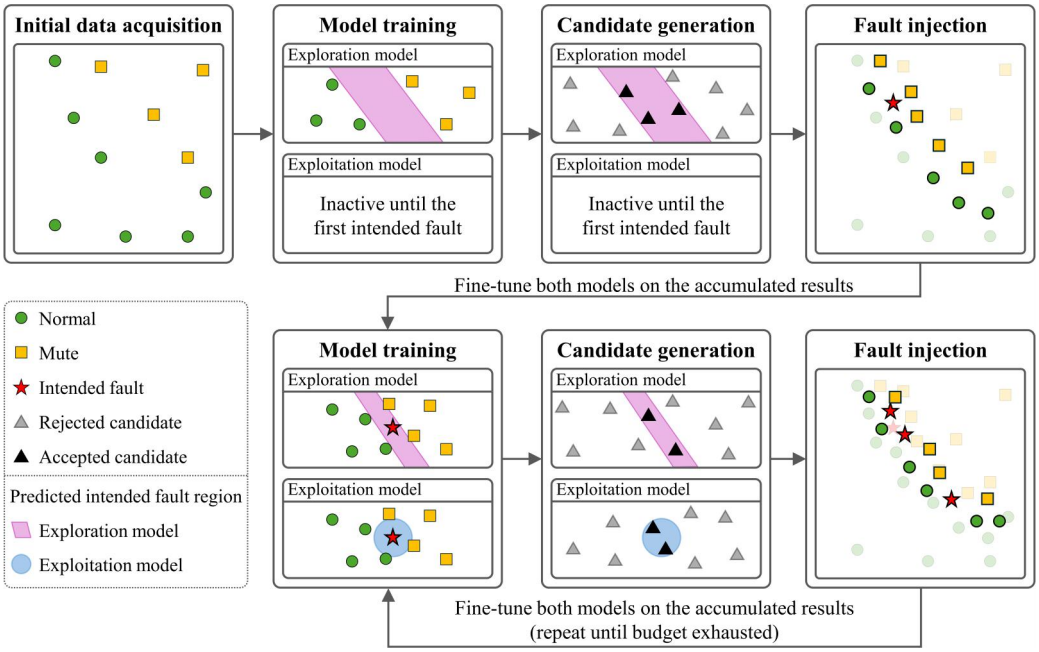


Figure 1: Schematic overview of the proposed framework, illustrating the cycle of candidate generation and model fine-tuning.

Substituting M and Q into Equation (1),

$$u < \frac{\Pr(P = p \mid V = v_{\text{int}})}{M \cdot \Pr(Q = p)} = \frac{\alpha \cdot \Pr(V = v_{\text{int}} \mid P = p)}{\alpha \cdot |\mathcal{P}| \cdot (1/|\mathcal{P}|)} = \Pr(V = v_{\text{int}} \mid P = p).$$

Since $m(p)_{v_{\text{int}}} \approx \Pr(V = v_{\text{int}} \mid P = p)$, the criterion becomes $u < m(p)_{v_{\text{int}}}$. □

The envelope in Property 2 uses the loose bound $\Pr(V = v_{\text{int}} \mid P = p) \leq 1$. When the maximum score $M_m = \max_p m(p)_{v_{\text{int}}}$ is much smaller than 1, this bound is overly conservative and most proposals are rejected. Rescaling the envelope by M_m raises the acceptance rate by a factor of $1/M_m$ without changing the target distribution, since any valid envelope leaves the accepted samples distributed according to the target distribution: the acceptance criterion simplifies to $u < m(p)_{v_{\text{int}}}/M_m$ with $u \sim \mathcal{U}(0, 1)$, or equivalently $u < m(p)_{v_{\text{int}}}$ with $u \sim \mathcal{U}(0, M_m)$. To tighten the envelope further, we replace this single global bound with a piecewise-uniform proposal: we partition $\mathcal{P}$ into cells and bound each cell by the minimum of its empirical per-axis score maxima, which upper-bounds the cell maximum. Cells are then sampled in proportion to their volume times this bound, so that low-scoring cells receive few proposals. This affects only sampling efficiency, not the target distribution.

3.2 Discriminative-Model-based Parameter Search

As established in Section 3.1, a discriminative model can generate promising parameters via rejection sampling. Figure 1 gives an overview of the proposed parameter search framework.

3.2.1 Initial Data Acquisition

The procedure begins by collecting a baseline dataset for training the discriminative models. Specifically, a small number of fault injections are performed with randomly

sampled parameters. Prior studies have shown that FI parameters of the same verdict class tend to form contiguous regions in the parameter space [CPB+13, PBJC14, KBV24]. Owing to this clustering, a budget of a few thousand injections is generally sufficient to obtain a coarse estimate of the boundary between the `normal` and `mute` regions. Acquiring `intended fault` samples at this stage is helpful but not strictly required.

3.2.2 Discriminative Models

We employ two complementary models, an *exploration model* and an *exploitation model*, which target the `normal`–`mute` boundary and the neighborhoods of previously discovered `intended faults`, respectively.

Exploration Model. The exploration model is designed to capture the `normal`–`mute` boundary. Its backbone is a standard network that maps an input FI parameter to a scalar output x. We place three fixed anchors $c_0 = -1$, $c_1 = 0$, $c_2 = 1$, corresponding to the `mute`, `intended fault`, and `normal` classes, respectively, and convert x into a verdict distribution $\mathbf{o}$ by applying a softmin to the fourth powers of the distances between x and the anchors:

$$\mathbf{z}_i = T \cdot (x - c_i)^4, \quad \mathbf{o}_i = \frac{e^{-\mathbf{z}_i}}{\sum_{j=0}^{2} e^{-\mathbf{z}_j}}, \quad \text{for } i \in \{0,\, 1,\, 2\},$$

where $T > 0$ controls the sharpness of the softmin. We set $T = 10$, for which the maximum attainable `intended fault` probability is approximately 1. The exploration model is trained with a cross-entropy loss, using only the `normal` and `mute` samples. This prevents the estimated boundary from being biased toward the regions of discovered `intended fault` parameters.

The intuition is as follows. Parameters that consistently induce a `mute` verdict shift x in the negative direction, driving $\mathbf{o}_0 \to 1$, while parameters yielding a `normal` verdict shift it in the positive direction, driving $\mathbf{o}_2 \to 1$. Parameters in the boundary region, in contrast, produce conflicting supervisory signals, since nearby parameters yield a mixture of `normal` and `mute` verdicts. These opposing signals drive x toward the origin ($x \approx 0$), where the `intended fault` score is maximized. To formalize this, consider a parameter whose injections yield a `mute` verdict with probability π and a `normal` verdict with probability $1-\pi$, so that its expected label vector is $y = (\pi,\, 0,\, 1-\pi)$; the `intended fault` component is zero, as that class is absent from the training labels. The expected cross-entropy loss is then

$$L = -\pi \ln \mathbf{o}_0 - (1 - \pi) \ln \mathbf{o}_2.$$

Setting $\partial L/\partial x = 0$, with $\partial L/\partial \mathbf{z}_i = y_i - \mathbf{o}_i$ and $\partial \mathbf{z}_i/\partial x = 4T(x - c_i)^3$, gives

$$\begin{aligned}\frac{\partial L}{\partial x} = \sum_{i=0}^{2} \frac{\partial L}{\partial \mathbf{z}_i} \frac{\partial \mathbf{z}_i}{\partial x} &= -4T\big[(\mathbf{o}_0 - \pi)(x+1)^3 + \mathbf{o}_1 x^3 + (\mathbf{o}_2 - 1 + \pi)(x-1)^3\big] \\ &= -4T\big[(1 + \mathbf{o}_0 - \mathbf{o}_2 - 2\pi)(3x^2+1) - 3\mathbf{o}_1 x\big] = 0.\end{aligned}$$

Expanding the cubic terms and using $\sum_{i=0}^{2} \mathbf{o}_i = 1$ eliminates the cubic term, which yields π as a function of x and the network outputs:

$$\pi = \frac{1}{2}\left[1 + \mathbf{o}_0 - \mathbf{o}_2 - \frac{3x}{3x^2+1}\mathbf{o}_1\right]. \tag{2}$$

Since $\mathbf{o}$ is itself a function of x, Equation (2) defines π as a function of x; inverting it numerically gives the stationary point x, and hence the `intended fault` score $\mathbf{o}_1$, for each π. Figure 2 shows the resulting relationship. The score is symmetric about $\pi = 0.5$, where

`normal` and `mute` verdicts are equally likely, and exhibits a broad plateau around this point. In contrast, for parameters that yield only `normal` ($\pi = 0$) or only `mute` ($\pi = 1$) verdicts, the optimum is driven toward $x \to +\infty$ or $x \to -\infty$, respectively, forcing $\mathbf{o}_1 \to 0$. The output layer therefore sustains a high `intended fault` score for parameters near the boundary, where both `normal` and `mute` verdicts are observed, while the score decays sharply for parameters that fall squarely within a single verdict class. By design, the exploration model directs candidate generation toward the boundary.

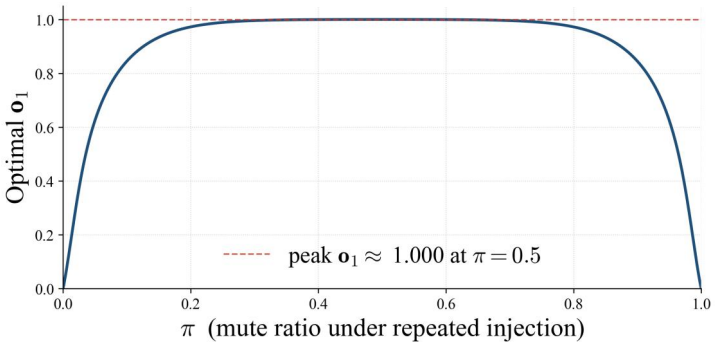


Figure 2: Optimal `intended fault` score $\mathbf{o}_1$ as a function of the mute ratio π, obtained from Equation (2).

Exploitation Model. The exploitation model performs an intensive search in the vicinity of the known `intended fault` parameters. Unlike the exploration model, which encodes the boundary hypothesis in a specialized output layer, the exploitation model learns directly from the observed parameters of all four verdict classes. We therefore employ a standard classifier with a softmax output layer, trained with a cross-entropy loss. Because a classifier cannot model a class with no training samples, the exploitation model becomes active only once at least one `intended fault` parameter has been identified.

3.2.3 Parameter Generation via Rejection Sampling and Fine-Tuning

By Property 2, the exploitation model, despite being a discriminative model, can generate samples from the posterior $\Pr(P \mid V = v_{\text{int}})$, concentrating candidates near the discovered `intended fault` parameters. The exploration model does not estimate this likelihood: by design, it follows the `normal`–`mute` boundary, and its `intended fault` score is a surrogate for the likelihood. For the exploration model, we apply rejection sampling to draw parameters in proportion to this score, which concentrates candidates along the boundary where the score is highest. With a uniform proposal, this yields the same acceptance criterion $u < m(p)_{v_{\text{int}}}$. The two models share an identical sampling procedure, differing only in the score that drives it.

The verdicts obtained by injecting faults with the generated parameters provide fine-grained observations near the discovered `intended fault` parameters and along the boundary, which we use to fine-tune the models. Our framework repeats a cycle of fault injection, model fine-tuning, and generation of a new batch of parameters, with a batch size of a few thousand parameters. In each batch, 10% of the candidates are sampled uniformly at random, independently of the models, so that the models retain global information about the parameter space and do not overfit to the regions explored so far. Once the exploitation model becomes active, the remaining parameters are split evenly between the two models, balancing exploration and exploitation. In this way, the proposed method

explores the broad `normal`–`mute` boundary while refining the search around the discovered `intended fault` parameters.

3.3 Refinement-Oriented Successive Halving Algorithm (RSHA)

The parameter search framework of Section 3.2 efficiently discovers a broad set of `intended fault` parameters. One goal of FI parameter search, however, is to identify a single parameter that induces the `intended fault` most reliably. To this end, we introduce the Refinement-Oriented Successive Halving Algorithm (RSHA), a variant of SHA. SHA was originally proposed for the multi-armed bandit problem and has since been adopted for hyperparameter optimization [KKS13]. It evaluates all candidate arms with a small initial budget, repeatedly discards the lower-performing fraction, and allocates larger budgets to the survivors. Prior work applying SHA to FI parameter search [WMP21] considered a black-box calibration setting initialized with random parameters, focusing on how many parameters to start with, trading this number off against the trials allocated to each. RSHA, in contrast, is applied after the search stage: its input is the set of candidates discovered during the search, each of which has induced the `intended fault` at least once. RSHA focuses on securing enough fault injections per surviving candidate to estimate its success rate to a target precision, thus acting as a post-processing refinement stage that returns the empirically best parameter for subsequent reliability validation.

To make this notion of precision concrete, we model each injection as a Bernoulli trial. For a fixed parameter p, the success probability is $s_p = \Pr(V = v_{\text{int}} \mid P = p)$, and we let $\hat{s}_p$ be the empirical success rate over n_{eval} independent trials. For sufficiently large n_{eval}, the half-width of the normal-approximation confidence interval at level $1 - \alpha$ satisfies

$$\epsilon = Z_{\alpha/2}\sqrt{\frac{\hat{s}_p(1-\hat{s}_p)}{n_{\text{eval}}}} \leq \frac{Z_{\alpha/2}}{2\sqrt{n_{\text{eval}}}}, \tag{3}$$

since $\hat{s}_p(1-\hat{s}_p) \leq 1/4$; here $Z_{\alpha/2}$ is the upper $\alpha/2$ critical value of the standard normal distribution. Given a target precision ϵ, this bound is met by setting $n_{\text{eval}} \geq Z_{\alpha/2}^2/(4\epsilon^2)$.

RSHA has three hyperparameters: the reduction factor η, the target cumulative number of evaluations per surviving candidate n_{eval}, and the minimum number of evaluations per candidate $n_{\min}$. Here n_{eval} sets the precision of the final estimate through Equation (3), η controls the aggressiveness of pruning, and $n_{\min}$ ensures that the first pruning decision is based on a sufficient number of observations. Let $\mathcal{C}$ be the initial candidate set, so that $r = \lceil \log_\eta |\mathcal{C}| \rceil$ is the maximum number of reduction rounds. We set the initial per-candidate budget n_0 so that a candidate evaluated over all r rounds would accumulate n_{eval} evaluations in total:

$$n_0 = \max\left(n_{\min},\ \left\lceil \frac{n_{\text{eval}}(\eta-1)}{\eta^r - 1} \right\rceil\right).$$

Ignoring rounding, round j (for $j = 0, 1, \dots$) retains about $|\mathcal{C}|/\eta^j$ candidates and evaluates each $n_0\eta^j$ times, so every round costs approximately $n_0|\mathcal{C}|$ fault injections, regardless of j. The total number of fault injections is therefore about $n_0|\mathcal{C}| \cdot r^*$, where r^* is the number of rounds the loop runs, which depends on the size of the candidate set. When the candidate set is small, n_0 is not clamped to $n_{\min}$, and the loop runs the full $r = \lceil \log_\eta |\mathcal{C}| \rceil$ rounds, reducing the candidate set to a single parameter. When the candidate set is large, n_0 is clamped to $n_{\min}$, and the cumulative evaluations reach n_{eval} before a single parameter remains; the loop then stops early, after $r_{\text{early}} = \lceil \log_\eta ((\eta-1)n_{\text{eval}}/n_0 + 1) \rceil$ rounds. The full procedure is given in Algorithm 1.

We note that Equation (3) is not a fixed-confidence guarantee that RSHA returns the truly most reliable parameter: because RSHA prunes candidates adaptively from finite,

Algorithm 1 Refinement-Oriented Successive Halving Algorithm (RSHA)

Require: Candidate set $\mathcal{C}$, target cumulative evaluations n_{eval}, minimum evaluations per candidate $n_{\min}$, reduction factor η
Ensure: Refined parameter p^*

Initialize success score $S(p) \leftarrow 0$ for all $p \in \mathcal{C}$
$r \leftarrow \lceil \log_\eta |\mathcal{C}| \rceil$
$n_0 \leftarrow \max\left(n_{\min}, \lceil n_{\text{eval}}\,(\eta - 1) \,/\, (\eta^r - 1) \rceil\right)$
$n_{\text{test}} \leftarrow n_0$
$n_{\text{cum}} \leftarrow 0$
while $|\mathcal{C}| > 1$ and $n_{\text{cum}} < n_{\text{eval}}$ **do**
  for each $p \in \mathcal{C}$ **do** ▷ Evaluate each surviving candidate n_{test} times
    for $i = 1$ to n_{test} **do**
      $v \leftarrow \text{FaultInjection}(p)$
      if $v = v_{\text{int}}$ **then**
        $S(p) \leftarrow S(p) + 1$
      end if
    end for
  end for
  $n_{\text{cum}} \leftarrow n_{\text{cum}} + n_{\text{test}}$
  $n_{\text{test}} \leftarrow \eta \cdot n_{\text{test}}$
  $k \leftarrow \lceil |\mathcal{C}|/\eta \rceil$
  $\mathcal{C} \leftarrow$ top-k parameters of $\mathcal{C}$ by $S(p)$ ▷ Retain the top $1/\eta$ fraction
end while
return $\text{argmax}_{p \in \mathcal{C}}\, S(p)$

noisy observations, a high-reliability candidate may still be discarded after a few unfavorable early outcomes. RSHA therefore serves as a budget-efficient refinement heuristic rather than an exact selection procedure.

4 Experimental Evaluation

4.1 Experimental Environment

Experimental Setup. We evaluated the proposed framework across four distinct FI scenarios. Our testbed comprises two physical fault sources—Voltage Glitching (VG) and Electromagnetic Fault Injection (EMFI)—targeting two commercial off-the-shelf microcontrollers: the 8-bit ATXmega128D4 (AVR XMEGA architecture) [Newb] and the 32-bit SAM4S2A (Arm Cortex-M4 architecture) [Newc]. The goal of FI is to subvert the PIN verification routine of Listing 1, making it return `SUCCESS` for an incorrect input. We performed VG with the ChipWhisperer-Husky [Newa] and EMFI with the ChipSHOUTER [Newd]. Table 1 lists the configurable parameters and their search spaces.

```
1 // ... (variable initialization)
2 for (i = 0; i < 4; i++)
3     if (pin[i] == input[i])
4         digits_ok++;
5 if (digits_ok == 4)
6     return SUCCESS;
7 else
8     return FAIL;
```

Listing 1: Target code for PIN verification.

Table 1: Configurable fault injection parameters and their search spaces.

Type	Parameter	Description	Search Space	Resolution
Timing	External offset	Delay from trigger to glitch injection.	$[0, 72]$ (ATXmega) $[0, 168]$ (SAM4S)	1 cycle
	Internal offset	Phase offset from the rising clock edge.	$(-50\%, 50\%)$	1/2800 cycle
VG	Width	Pulse width as a fraction of the clock period.	$[0\%, 50\%)$	1/2800 cycle
	Repeat	Number of consecutive glitches per trigger.	$[1, 10]$	1 cycle
EMFI	Voltage	Pulse voltage applied to the probe.	$[150, 500]$	1 V
	Position (x, y)	Spatial coordinates of the probe.	$[0, 330{,}000]$ (ATXmega) $[0, 300{,}000]$ (SAM4S)	1 step

Implementation Specifics. We instantiated the proposed framework (Section 3.2) with a sampling batch size of 2,000 and ran it for 50 iterations, giving a total budget of 100,000 injections per experiment; we chose this budget so that a single Random Search campaign completes within a day. Both discriminative models used a standard Multilayer Perceptron (MLP) backbone with Leaky-ReLU activations. We tested four standard configurations, varying the depth and width (three layers of 128 units versus four layers of 256 units) and the learning rate (10^{-3} and 10^{-4}). Training used the Adam optimizer for 100 epochs with a batch size of 64. For the RSHA phase, we set the minimum evaluation count to $n_{\min} = 2$, the target cumulative evaluation count per candidate to $n_{\text{eval}} = 700$, and the reduction factor to $\eta = 3$. To assess the reliability of the refined parameter, we performed 10,000 independent validation injections. By Equation (3), at the 99% confidence level ($\alpha = 0.01$) the worst-case estimation error is 4.87 percentage points for the final RSHA decision ($n_{\text{eval}} = 700$) and 1.29 percentage points for the final validation (10,000 injections).

Baseline Methodologies. We benchmarked our method against eight parameter search strategies: seven state-of-the-art techniques and Random Search (RS). To ensure a fair comparison, we enforced an identical budget of 100,000 injections across all methods. Unless otherwise stated below, every baseline used the hyperparameter values reported in its original publication. For AZB [CPB+13], we performed three injections per parameter to enable the detection of `changing` behavior, and evaluated three grid initialization resolutions of 5, 7, and 9 subdivisions per dimension. For the Genetic Algorithm (GA) [PBJC14], we likewise performed three injections per candidate evaluation, and evaluated population sizes of 30 and 300 and mutation rates of 0.1 and 0.3. The Memetic Algorithm (MA) [PBBJ15] and GridMA [KO24] were configured identically to the GA. Since GridMA was originally designed to partition the x–y space, for VG we instead partitioned the width–repeat space, where the `normal`–`mute` boundary is clearly visible. For the CGAN-based approach [KBV24], we aligned the network architecture and training hyperparameters with those of our framework. For SHA and SMAC [WMP21], we used the reference configurations of the original authors. The outcome of a fault injection is itself stochastic, so the result of a single run may be affected by randomness. RS, SHA, and SMAC used a single setting, whereas the other baselines were run over several hyperparameter settings, which averages out this effect. We therefore ran RS, SHA, and SMAC three times each and report the averaged results.

Table 2 reports the mean execution time of each method. These times should not be read as a direct comparison of algorithmic efficiency, as they are dominated by factors

specific to the experimental setup rather than by the algorithms themselves. For the non-learning-based methods—RS, AZB, GA, MA, GridMA, and SHA—whose computation is negligible, the running time is governed almost entirely by physical operations of the equipment, such as stage movements and device resets, and thus varies widely with the regions each method explores. For the learning-based methods—SMAC, CGAN, and the proposed method—the running time additionally includes model training, whose duration depends on the computing hardware. The proposed method trains neural networks, yet its running time remains comparable to that of the other methods.

Table 2: Mean execution time (hours) of each method, for each fault source and target.

Source	Target	RS	AZB	GA	MA	GridMA	SHA	SMAC	CGAN	Ours
VG	ATXmega	4.20	5.45	6.50	6.68	5.57	3.75	8.17	11.56	11.89
	SAM4S	4.09	6.42	6.34	6.63	5.44	5.12	8.97	11.66	12.17
EMFI	ATXmega	16.76	16.58	21.63	37.59	14.26	16.10	28.72	23.72	23.60
	SAM4S	20.18	12.47	20.01	19.78	12.94	10.34	28.12	17.98	22.10

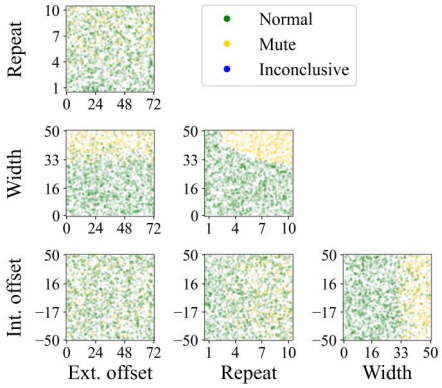
4.2 FI Parameter Search

Search Process on VG. Figure 3 illustrates the evolution of the parameter search for VG on the ATXmega. Because the parameter space is high-dimensional, we project the samples onto two-dimensional planes, one for each pair of parameters. Figure 3a shows the initial data acquisition phase of 2,000 random injections. Consistent with prior studies [CPB+13, PBJC14, KBV24], the `normal` and `mute` samples form clusters, indicating that a budget of a few thousand injections suffices to coarsely delineate the boundary between them. The internal and external offset parameters show little correlation with this separation, whereas the width and repeat parameters define the boundary clearly, as seen in the central plot of Figure 3a. Under the boundary hypothesis, `intended faults` are expected to lie within this transition zone.

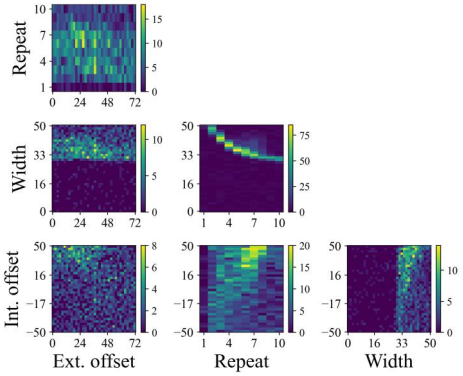
Figure 3b shows the distribution of the first candidate batch. Since no `intended fault` has yet been found, 90% of the parameters are generated by the exploration model and the remaining 10% are sampled uniformly at random. The exploration model locates the boundary effectively and concentrates the generated parameters along it. As discussed in Section 3.2.3, the initial random search data carry only coarse information, so a model trained on them estimates a broad region as the boundary; the newly generated parameters carry fine-grained information about the boundary and are therefore well suited to refining the model there. Figure 3c shows the parameters generated by this refined model, which predicts the boundary more precisely than before. Injecting faults with these parameters yields the first `intended fault` samples, overlaid as red points in Figure 3d.

With `intended fault` samples now available, the exploitation model can learn their distribution and contribute to parameter generation. Figure 3d shows the resulting batch: since both models now contribute equally, the search covers the broad boundary of Figure 3c while concentrating on the neighborhoods of the discovered `intended faults`. This batch yields new `intended faults` along the boundary and additional ones around those found previously, as shown in Figure 3e. This illustrates how the proposed method balances exploration and exploitation. After the full budget is exhausted (Figure 3f), `intended faults` are distributed across the boundary, indicating that the method explores the broad boundary effectively.

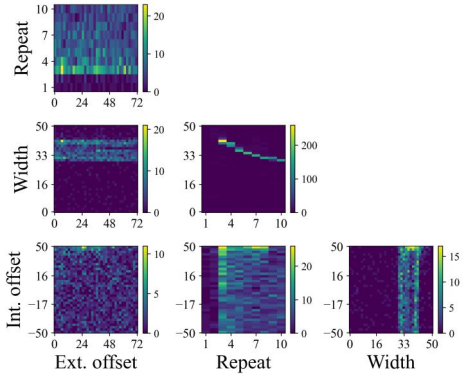
Search Process on EMFI. Figure 4 shows the search for EMFI on the SAM4S. The `normal`–`mute` boundary is less distinct than in VG. Whereas the VG boundary is determined by the width and repeat parameters and stands out clearly in the corresponding



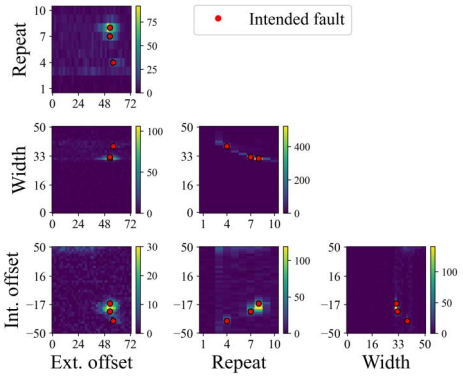
(a) Initial random search.



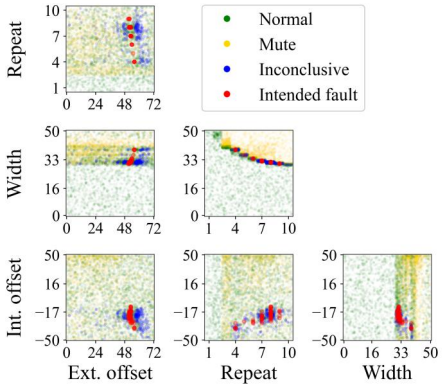
(b) First candidate batch.



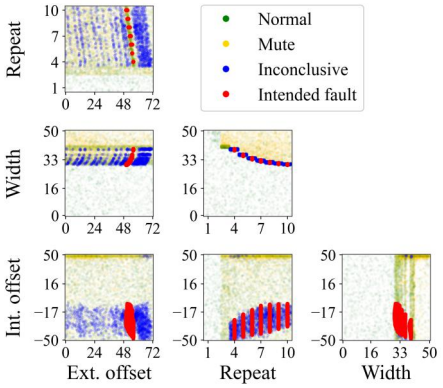
(c) Second candidate batch.



(d) Third candidate batch.



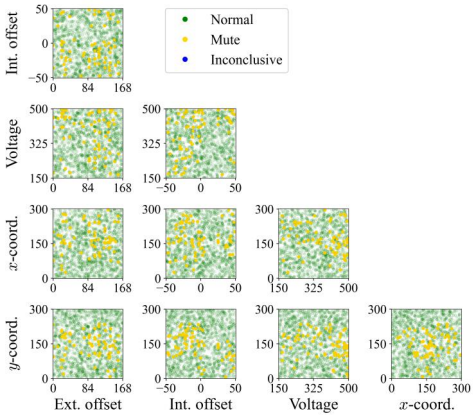
(e) Cumulative faults after four batches.



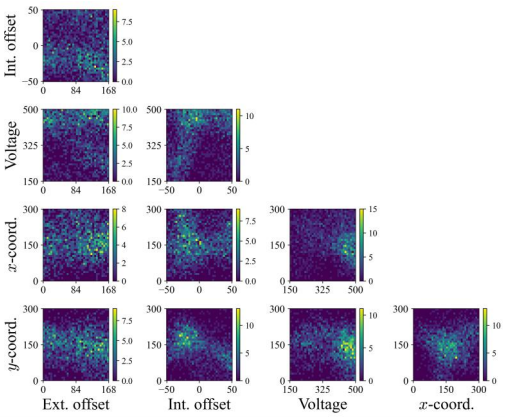
(f) Cumulative faults after the full budget.

Figure 3: Evolution of the parameter search for VG on the ATXmega. Each panel projects the samples onto the six planes spanned by pairs of parameters. All candidate batches include 10% uniformly random parameters.

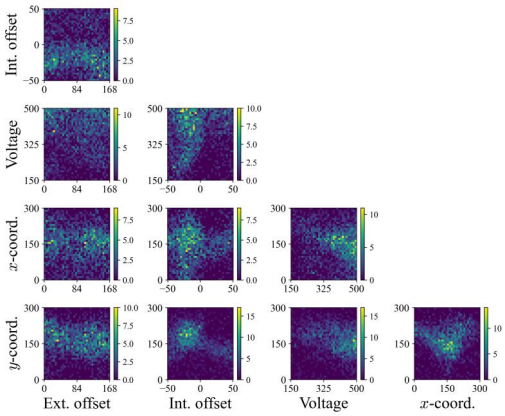
2D projection, the EMFI boundary is governed by a combination of parameters, so each 2D projection superimposes the remaining ones and the `normal` and `mute` regions overlap substantially. For instance, a high pulse voltage alone does not induce a `mute` verdict: a `normal` verdict is still likely when the probe is not positioned over a component involved in the computation, so the two classes mix at high voltage. The boundary is therefore only



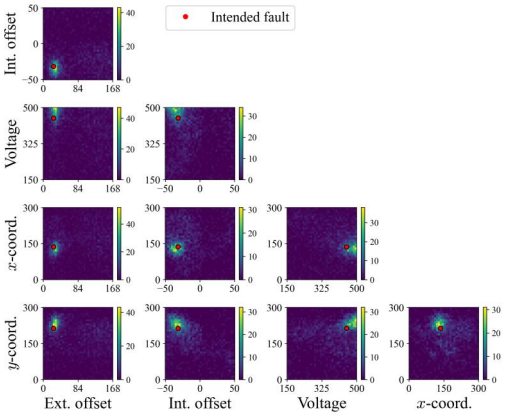
(a) Initial random search.



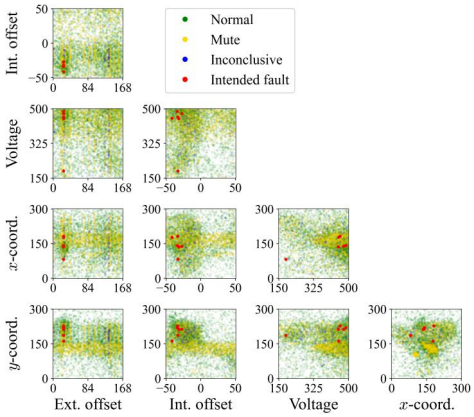
(b) First candidate batch.



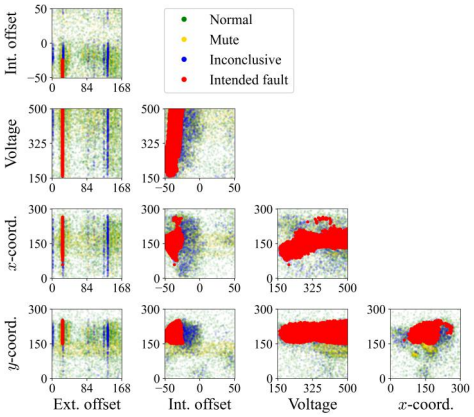
(c) Second candidate batch.



(d) Sixth candidate batch.



(e) Cumulative faults after seven batches.



(f) Cumulative faults after the full budget.

Figure 4: Evolution of the parameter search for EMFI on the SAM4S. Each panel projects the samples onto the ten planes spanned by pairs of parameters. All candidate batches include 10% uniformly random parameters.

faintly visible, appearing in different regions across the projections—on the left and top of the internal offset–voltage plane, as an inverted triangle at the center of the x–y plane, and along the vertical center of the right side of each voltage–coordinate plane.

The search proceeds as in the VG case, though `intended faults` are rarer and take longer to appear. The exploration model generates parameters along the boundary (Figure 4b) and localizes it more precisely after fine-tuning (Figure 4c); in the x–y projection, the generated parameters closely follow the boundary. No `intended fault` is found in the first four candidate batches; the first one appears only in the fifth. From the next batch onward (Figure 4d), the exploitation model joins parameter generation, concentrating on the neighborhood of the discovered `intended fault` while the exploration model continues to cover the boundary. Injecting the resulting parameters discovers faults both near the previously found one and along the broad boundary (Figure 4e). After the full budget is exhausted (Figure 4f), the `intended faults` are distributed across the boundary.

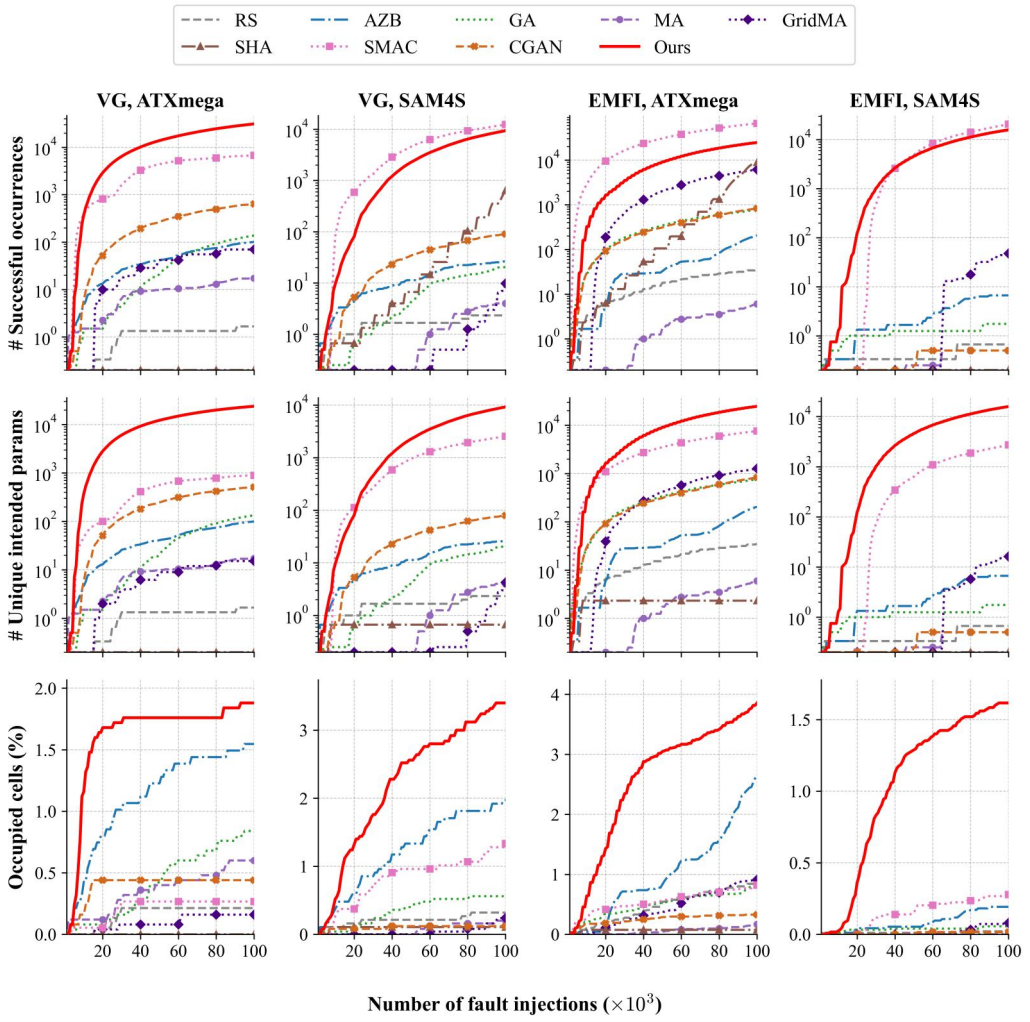


Figure 5: Comparison of the proposed method against the state-of-the-art methods on the four scenarios, as a function of the number of fault injections. The rows report the number of `intended fault` occurrences, the number of *unique* `intended fault` parameters, and the occupied cell ratio.

Comparison with State-of-the-Art Methods. Figure 5 compares the search behavior of all methods using three quantitative metrics as a function of the number of injections. The first is the number of `intended fault` occurrences, which reflects the success rate during the search. On this metric, SMAC performs best in several scenarios: as a Bayesian optimizer, it repeatedly attacks the parameter with the highest observed success rate, which inflates the count. As argued in Section 1, however, the second objective of FI parameter search is to discover `intended fault` parameters spread across the parameter space. We therefore compare two further metrics.

The second metric is the number of unique `intended fault` parameters, which excludes repeated measurements of the same parameter. Here the advantage of SMAC disappears: because it spends most of its budget on a few high-success parameters, its unique count falls well below its total, whereas the proposed method attains the most unique parameters beyond roughly 22,000 injections, reaching up to 26.7× that of the best baseline within the full budget. The third metric is the occupied cell ratio, which partitions each parameter range into five bins—yielding 5^4 cells for VG and 5^5 for EMFI—and measures the fraction of cells containing at least one `intended fault`, thus quantifying how broadly the discovered faults are spread across the parameter space. On this metric, the proposed method covers the largest region beyond roughly 10,000 injections, up to 5.8× that of the best baseline within the full budget, followed by AZB. The exception is EMFI on the SAM4S, where `intended faults` are so rare that AZB does not surpass the coverage of SMAC within the budget; its steeper growth suggests that it would do so given a larger budget, as in the other scenarios. This is consistent with the discussion in Section 1.1: the exploitation-biased methods, such as GA, MA, and CGAN, discover many `intended fault` parameters during the search but attain a low occupied cell ratio, whereas the exploration-biased AZB shows the opposite, covering a broad region but finding comparatively few faults. These results show that the proposed method discovers not only many but also diverse `intended fault` parameters, spread broadly across the parameter space.

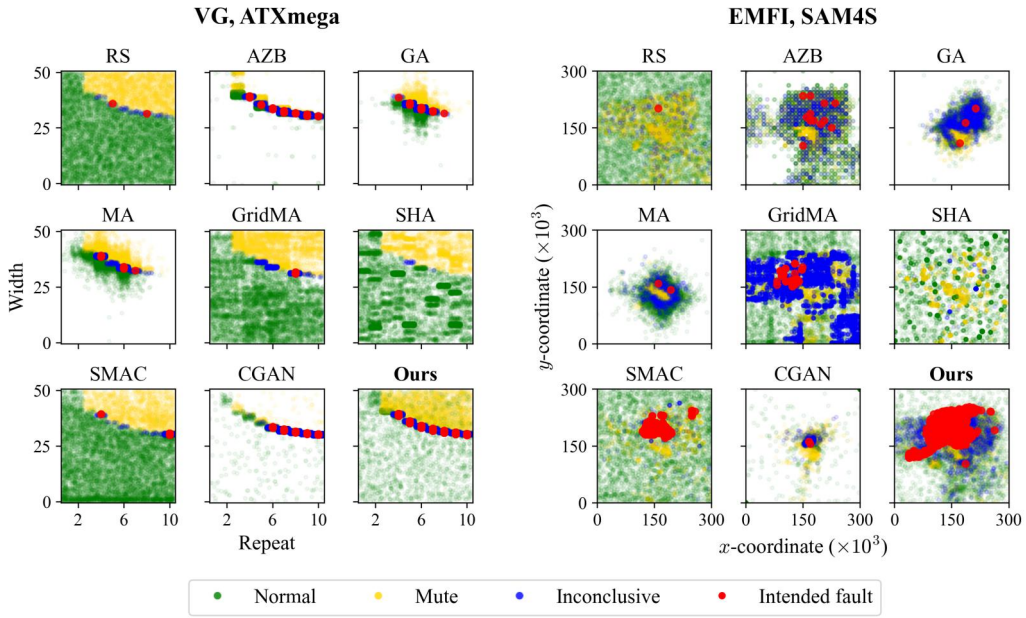


Figure 6: Discovered `intended fault` parameters (red) after the full budget, for VG on the ATXmega (left, width–repeat plane) and EMFI on the SAM4S (right, spatial coordinates).

To illustrate these search behaviors, Figure 6 shows the discovered parameters after the full budget for VG on the ATXmega and EMFI on the SAM4S. Parameters are projected onto the width–repeat plane, where the `normal`–`mute` boundary is most distinct for VG, and onto the spatial coordinates for EMFI. Consistent with the analysis above, the exploration-biased AZB spreads its `intended faults` broadly along the boundary. The exploitation-biased GA, MA, and CGAN, in contrast, remain confined to a narrow region: once an `intended fault` is found, they concentrate the search around it, so they explore less broadly than the other methods. GridMA covers diverse regions, but, as noted in Section 1.1, it also spends part of its budget on regions unlikely to yield an `intended fault`—such as those producing only `normal` verdicts—and thus attains a low overall success rate. SHA, whose candidates are less diverse than those of RS, fails to discover `intended faults` altogether. The proposed method, by balancing exploration and exploitation, covers the broadest region of all while still concentrating on the discovered `intended faults`.

4.3 Reliability of the Refined Parameter

For an adversary, the practical value of finding diverse `intended fault` parameters lies in whether they yield a parameter that reliably induces the fault. We therefore evaluate the reliability of the single parameter that each method produces after refinement. For each search strategy, we applied RSHA to the set of `intended fault` parameters that the strategy discovered, obtaining a single parameter, and then performed 10,000 independent validation injections with it to measure its success rate empirically.

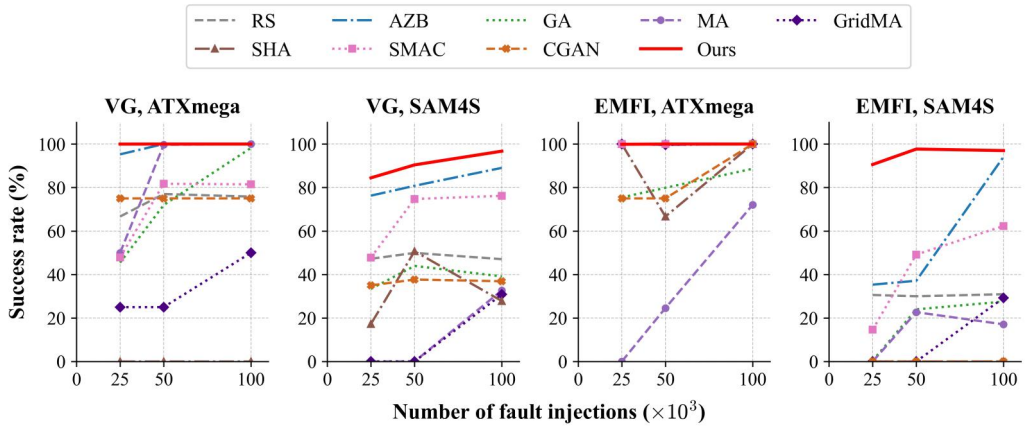


Figure 7: Success rate of the single parameter refined by RSHA from the `intended fault` parameters discovered by each method, as a function of the total budget. Each success rate is measured over 10,000 independent validation injections.

Figure 7 reports the success rate of the refined parameter as a function of the total budget. The proposed method achieves the highest success rate in all scenarios. The advantage is most pronounced on the SAM4S. For VG-SAM4S, it exceeds the best baseline by 8.2, 9.6, and 7.6 percentage points at budgets of 25,000, 50,000, and 100,000 injections, respectively. For EMFI-SAM4S, the gap is far larger at smaller budgets: it exceeds the best baseline by 55.3 and 48.5 percentage points at 25,000 and 50,000 injections, respectively, and by 2.9 percentage points at 100,000 injections. Among the baselines, AZB performs best. This is consistent with the preceding analysis and suggests why diversity matters for reliability: the broader the discovered set, the larger and more spread the pool from which RSHA selects, and thus the higher the chance that it contains a highly reliable parameter.

On EMFI-SAM4S, AZB is less reliable than SMAC at a budget of 50,000 but reaches 94.1% once the full budget is used; the narrowing gap at 100,000 reflects this. Consistent with Section 4.2, this scenario requires a larger budget because `intended faults` are rare. The exploitation-biased GA, MA, and CGAN, whose discovered parameters are the least diverse, attain comparatively low reliability.

5 Discussion

Intended Faults off the Boundary. The boundary hypothesis underlies both our method and prior search methods [CPB+13, PBJC14, PBBJ15, KBV24]. Our experiments, however, reveal a case in which some `intended faults` arise far from any `normal`–`mute` boundary. In the EMFI experiment on the ATXmega (Figure 8), we analyzed a Random Search of 300,000 injections, in which the `intended faults` form distinct clusters.

Two `normal`–`mute` boundaries are clearly visible in the left panel of Figure 8, as circular regions in the upper-left and lower-right, where `normal` and `mute` verdicts occur in comparable numbers. Under the boundary hypothesis, the `intended faults` would be expected along these boundaries. Instead, they concentrate in a lower-left region ($30{,}000 < x < 150{,}000$, $10{,}000 < y < 120{,}000$), where the `normal`–`mute` boundary is less distinct: this region holds 89.52% of all `intended faults`, yet its verdict distribution is 90.97% `normal`, 1.73% `mute`, 7.04% `inconclusive`, and 0.26% `intended fault`. The next largest share (6.67%) appears in the upper-right region ($200{,}000 < x < 330{,}000$, $200{,}000 < y < 330{,}000$), which is almost entirely `normal` (99.58%), with `mute` (0.008%) even rarer than `intended fault` (0.01%). The upper-left boundary, despite a substantial `mute` fraction (21.71%), yields only 3.81% of the `intended faults`, and the lower-right boundary yields almost none. A boundary-based search can thus locate faults along the distinct upper-left boundary and the weaker lower-left boundary, but those in the upper-right region, far from any boundary, remain difficult to find.

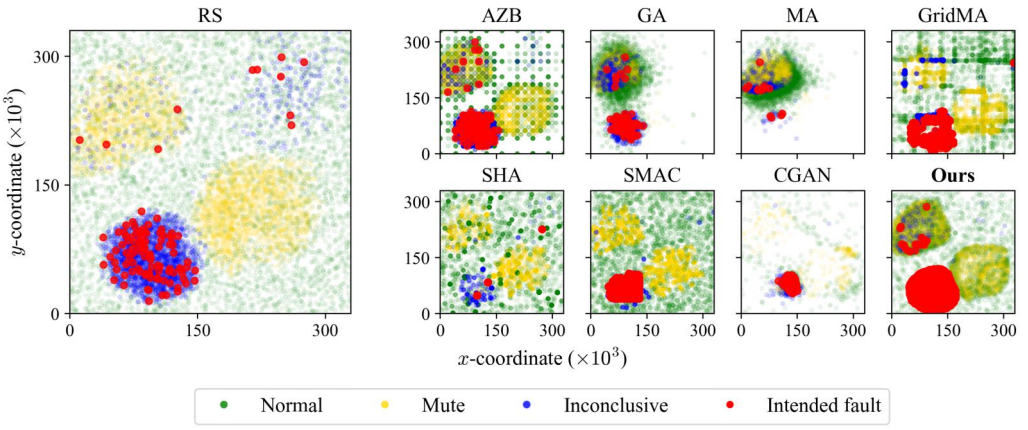


Figure 8: Discovered `intended fault` parameters for EMFI on the ATXmega, projected onto the x–y plane. Left: a Random Search of 300,000 injections. Right: the other methods at the full budget.

We performed EMFI without decapsulating the chip, so these faults may arise because the electromagnetic pulse does not reach the internal circuitry at full strength. Regardless of the cause, the observation shows that, in some settings, a small fraction of `intended fault` parameters can lie far from any `normal`–`mute` boundary. Since the large majority still lie on a boundary, albeit sometimes a weak one, the proposed method remains sufficient for an adversary seeking a highly reliable parameter. For a security evaluator who must

map all `intended fault` regions, however, increasing the proportion of random parameters in the proposed method is advisable: a random parameter that lands in an off-boundary region allows the exploitation model to search its neighborhood.

Inconclusive Verdicts as a Hint. In the scatter plots of Figures 6 and 8, the `inconclusive` verdicts follow a distribution similar to that of the `intended faults`. This resemblance likely arises because an `inconclusive` verdict typically results from corrupting the wrong instruction or data, much as an `intended fault` results from the intended corruption. Our method uses `inconclusive` verdicts only as one of the four training classes of the exploitation model and does not exploit them as a hint toward `intended faults`. Doing so could accelerate the parameter search, since `inconclusive` verdicts occur more frequently than `intended faults` early in the search. We leave this to future work.

6 Conclusion

We presented a discriminative-model-based FI parameter search framework that discovers diverse `intended fault` parameters, together with RSHA, which refines them into a single reliable parameter. The framework couples two models that balance exploration and exploitation: an exploration model that learns the `normal`–`mute` boundary, and an exploitation model that learns the distribution of the discovered `intended faults`. RSHA then selects a highly reliable one among the discovered parameters. Across four attack scenarios, the proposed method finds up to 26.7× as many distinct `intended fault` parameters as the best baseline and attains up to 5.8× the best baseline's occupied cell ratio. Moreover, at the full budget, the parameter that RSHA selects from our discovered set achieves a success rate up to 7.6 percentage points higher than the one it selects from the best baseline's set.

Beyond these results, we observed a case in which the boundary hypothesis does not fully hold: in one hardware configuration, some `intended faults` arose far from any `normal`–`mute` boundary. Although most `intended fault` parameters still lie on a boundary, albeit sometimes a weak one, this suggests that boundary-centric search alone may miss fault regions, which matters for an evaluator aiming to map the `intended fault` space exhaustively. For such cases, we recommend increasing the proportion of random parameters to also capture off-boundary faults. We also found that `inconclusive` verdicts follow a distribution similar to that of `intended faults`; exploiting this similarity is a promising direction for future work.

References

[AK97] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *5th Security Protocols Workshop*, volume 1361 of *LNCS*, pages 125–136. Springer, April 1997.

[BCV13] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013. doi:10.1109/TPAMI.2013.50.

[BFP19] Claudio Bozzato, Riccardo Focardi, and Francesco Palmarini. Shaping the glitch: Optimizing voltage fault injection attacks. *IACR TCHES*, 2019(2):199–224, 2019. URL: https://tches.iacr.org/index.php/TCHES/article/view/7390, doi:10.13154/tches.v2019.i2.199-224.

[Bis07] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007. URL: https://www.worldcat.org/oclc/71008143.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski, Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 513–525. Springer, Berlin, Heidelberg, August 1997. doi:10.1007/BFb0052259.

[CPB+13] Rafael Boix Carpi, Stjepan Picek, Lejla Batina, Federico Menarini, Domagoj Jakobovic, and Marin Golub. Glitch it if you can: Parameter search strategies for successful fault injection. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 236–252. Springer, 2013. doi:10.1007/978-3-319-08302-5_16.

[DDRT12] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In Guido Bertoni and Benedikt Gierlichs, editors, *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012*, pages 7–15. IEEE Computer Society, 2012. doi:10.1109/FDTC.2012.15.

[GPM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014. URL: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

[GZW+25] Xue Gong, Xin Zhang, Qianmei Wu, Fan Zhang, Junge Xu, Qingni Shen, and Zhi Zhang. Practical opcode-based fault attack on AES-NI. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(3):693–716, 2025. URL: https://doi.org/10.46586/tches.v2025.i3.693-716, doi:10.46586/TCHES.V2025.I3.693-716.

[HHH+26] Xiao Huang, Zhuo Huang, Yituo He, Quan Yuan, Chao Sun, Mehdi Tibouchi, and Yu Yu. Faultless key recovery: Iteration-skip and loop-abort fault attacks on less. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(3):71–97, Jul. 2026. URL: https://tches.iacr.org/index.php/TCHES/article/view/13080, doi:10.46586/tches.v2026.i3.71-97.

[JD26] Sönke Jendral and Elena Dubrova. Fault attacks on voleith signature schemes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(1):225–249, Jan. 2026. URL: https://tches.iacr.org/index.php/TCHES/article/view/12675, doi:10.46586/tches.v2026.i1.225-249.

[JDGJ26] Sönke Jendral, Elena Dubrova, Qian Guo, and Thomas Johansson. Correction fault attack on cross under unknown bit flips. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(2):192–217, Apr. 2026. URL: https://tches.iacr.org/index.php/TCHES/article/view/12885, doi:10.46586/tches.v2026.i2.192-217.

[KBV24] Troya Çagil Köylü, Cornelis Christiaan Berg, and Praveen Kumar Vadnala. Cgan-based automated fault injection. In *IEEE European Test Symposium, ETS 2024, The Hague, Netherlands, May 20-24, 2024*, pages 1–6. IEEE, 2024. doi:10.1109/ETS61313.2024.10568027.

[KFP21] Marina Krcek, Daniele Fronte, and Stjepan Picek. On the importance of initial solutions selection in fault injection. In *18th Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2021, Milan, Italy, September 17, 2021*, pages 1–12. IEEE, 2021. doi:10.1109/FDTC53659.2021.00011.

[KGAS25] Anup Kumar Kundu, Shibam Ghosh, Aikata Aikata, and Dhiman Saha. Tofa: Towards fault analysis of GIFT and gift-like ciphers leveraging truncated impossible differentials. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(3):614–643, 2025. URL: https://doi.org/10.46586/tches.v2025.i3.614-643, doi:10.46586/TCHES.V2025.I3.614-643.

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In Scott B. Guthery and Peter Honeyman, editors, *Proceedings of the 1st Workshop on Smartcard Technology, Smartcard 1999, Chicago, Illinois, USA, May 10-11, 1999*. USENIX Association, 1999. URL: https://www.usenix.org/conference/usenix-workshop-smartcard-technology/design-principles-tamper-resistant-smartcard.

[KKS13] Zohar Shay Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1238–1246. JMLR.org, 2013. URL: http://proceedings.mlr.press/v28/karnin13.html.

[KO24] Marina Krcek and Thomas Ordas. Diversity algorithms for laser fault injection. In Martin Andreoni, editor, *Applied Cryptography and Network Security Workshops - ACNS 2024 Satellite Workshops, AIBlock, AIHWS, AIoTS, SCI, AAC, SiMLA, LLE, and CIMSS, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part I*, Lecture Notes in Computer Science, pages 121–138. Springer, 2024. doi:10.1007/978-3-031-61486-6_8.

[KOFP22] Marina Krcek, Thomas Ordas, Daniele Fronte, and Stjepan Picek. The more you know: Improving laser fault injection with prior knowledge. In *Workshop on Fault Detection and Tolerance in Cryptography, FDTC 2022, Virtual Event / Italy, September 16, 2022*, pages 18–29. IEEE, 2022. doi:10.1109/FDTC57191.2022.00012.

[MSPB18] Antun Maldini, Niels Samwel, Stjepan Picek, and Lejla Batina. Genetic algorithm-based electromagnetic fault injection. In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2018, Amsterdam, The Netherlands, September 13, 2018*, pages 35–42. IEEE Computer Society, 2018. doi:10.1109/FDTC.2018.00014.

[Newa] NewAE Technology Inc. ChipWhisperer-Husky. https://chipwhisperer.readthedocs.io/en/latest/Capture/ChipWhisperer-Husky.html.

[Newb] NewAE Technology Inc. CW303 XMEGA Target. https://chipwhisperer.readthedocs.io/en/latest/Targets/CW303%20XMEGA.html.

[Newc] NewAE Technology Inc. CW312T-SAM4S. https://chipwhisperer.readthedocs.io/en/latest/chipwhisperer-target-cw308t/CW312T_ATSAM4S/README.html.

[Newd] NewAE Technology Inc. CW520 ChipSHOUTER. https://chipwhisperer.readthedocs.io/en/latest/ChipSHOUTER/ChipSHOUTER.html.

[NJ01] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 841–848. MIT Press, 2001. URL: https://proceedings.neurips.cc/paper/2001/hash/7b7a53e239400a13bd6be6c91c4f6c4e-Abstract.html.

[PBBJ15] Stjepan Picek, Lejla Batina, Pieter Buzing, and Domagoj Jakobovic. Fault injection with a new flavor: Memetic algorithms make a difference. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2015. doi:10.1007/978-3-319-21476-4_11.

[PBJC14] Stjepan Picek, Lejla Batina, Domagoj Jakobovic, and Rafael Boix Carpi. Evolving genetic algorithms for fault injection attacks. In *37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014, Opatija, Croatia, May 26-30, 2014*, pages 1106–1111. IEEE, 2014. doi:10.1109/MIPRO.2014.6859734.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *CHES 2003*, volume 2779 of *LNCS*, pages 77–88. Springer, Berlin, Heidelberg, September 2003. doi:10.1007/978-3-540-45238-6_7.

[SA03] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski, Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 2–12. Springer, Berlin, Heidelberg, August 2003. doi:10.1007/3-540-36400-5_2.

[SAW25] Kevin Schneider, Lukas Auer, and Alexander Wagner. Fault attacks on ECC signature verification. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(4):1010–1052, 2025. URL: https://doi.org/10.46586/tches.v2025.i4.1010-1052, doi:10.46586/TCHES.V2025.I4.1010-1052.

[SC22] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans): Challenges, solutions, and future directions. *ACM Comput. Surv.*, 54(3):63:1–63:42, 2022. doi:10.1145/3446374.

[VOGT21] Jan Van den Herrewegen, David Oswald, Flavio D. Garcia, and Qais Temeiza. Fill your boots: Enhanced embedded bootloader exploits via fault injection and binary analysis. *IACR TCHES*, 2021(1):56–81, 2021. URL: https://tches.iacr.org/index.php/TCHES/article/view/8727, doi:10.46586/tches.v2021.i1.56-81.

[WMP21] Vincent Werner, Laurent Maingault, and Marie-Laure Potet. Fast calibration of fault injection equipment with hyperparameter optimization techniques. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 121–138. Springer, 2021. doi:10.1007/978-3-030-97348-3_7.

[WMT25] Weizhe Wang, Pierrick Méaux, and Deng Tang. Shortcut2secrets: A table-based differential fault attack framework. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(2):385–419, 2025. URL: https://doi.org/10.46586/tches.v2025.i2.385-419, doi:10.46586/TCHES.V2025.I2.385-419.

[WRBP20] Lichao Wu, Gerard Ribera, Noemie Beringuier-Boher, and Stjepan Picek. A fast characterization method for semi-invasive fault injection attacks. In Stanislaw Jarecki, editor, *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, Lecture Notes in Computer Science, pages 146–170. Springer, 2020. doi:10.1007/978-3-030-40186-3_8.

[ZZZ+26] Jiayu Zhang, Shihui Zheng, Xinming Zhu, Yingyan Chen, and Changhai Ou. Dspfa: Generalized persistent fault analysis based on distributional similarity. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(3):402–425, Jul. 2026. URL: https://tches.iacr.org/index.php/TCHES/article/view/13099, doi:10.46586/tches.v2026.i3.402-425.