# Vulnerability Assessment of Ciphers To Fault Attacks Using Reinforcement Learning

Hao Guo[1], Sayandeep Saha[2], Satwik Patnaik[1], Vasudev Gohil[1], Debdeep Mukhopadhyay[2] and Jeyavijayan (JV) Rajendran[1]

[1] Texas A&M University, College Station, Texas, USA
{guohao2019,satwik.patnaik,gohil.vasudev,jv.rajendran}@tamu.edu
[2] Indian Institute of Technology, Kharagpur, India
{sayandeep.iitkgp,Debdeep.mukhopadhyay}@gmail.com

**Abstract.** A fault attack (FA) is one of the most potent threats to cryptographic applications. Implementing a FA-protected block cipher requires knowledge of the exploitable fault space of the underlying crypto algorithm. The discovery of exploitable faults is a challenging problem that demands human expertise and time. Current practice is to rely on certain predefined fault models. However, the applicability of such fault models varies among ciphers. Prior work discovers such exploitable fault models individually for each cipher at the expanse of a large amount of human effort. Our work completely replaces human effort by using reinforcement learning (RL) over the huge fault space of a block cipher to discover the effective fault models automatically. Validation on an AES block cipher demonstrates that our approach can automatically discover the effective fault models within a few hours, outperforming prior work, which requires days of manual analysis. The proposed approach also reveals vulnerabilities in the existing FA-protected block ciphers and initiates an end-to-end vulnerability assessment flow.

**Keywords:** Reinforcement Learning · Fault Attack

## 1 Introduction

A fault attack (FA) is a class of active implementation-based attacks in which an adversary deliberately perturbs the computation to extract secrets through the faulty system response or bypass some security mechanism. FAs are applicable over a wide range of computing platforms starting from small embedded systems (e.g., RFID tags) running 8-bit processors [BBB+10, SJR+20], medium-scale 32-bit ARM processors [RRB+19, HPP21a], to high-end cloud applications running Intel chips and GPUs [SFK20, BM16, QWLQ19, MOG+20]. Hardware circuits such as FPGA and ASIC are also vulnerable [SBR+20, BLMR19, CML+11] to FAs. Some popular and recent exploits of FAs include Sony Play-stations [Lu19], bitcoin wallets [bit22], Starlink User Terminal [att22], deep neural nets [HFK+19], and many more. The that FAs can be launched on such commercial products at very low cost makes them extremely potent threats.

While applicable to symmetric and public-key cryptosystems, FAs have seen significant progress for keyed symmetric-key primitives (e.g., block ciphers). This is attributed to the widespread applicability of such primitives, even in modern post-quantum schemes [HPP21b]. Therefore, protecting and testing symmetric-key cryptosystems for potential FA threats remains one of the most active research areas in hardware security.

Even though FAs in block ciphers are well-explored, developing protected implementations (even for well-known attacks) remains challenging. This is partly because FAs are often algorithm-specific. Secondly, FAs require logical abstraction of physical faults,

called fault models, to perform the attacks. In general, the attacks have to be discovered separately for each block-cipher algorithm and each fault model, which requires (mathematically intense) expertise and time. For instance, it took roughly 8 years (after the introduction of faults attacks) to find out the most optimal approach to attacking AES [TMA11]. While finding out a single attack instance is sufficient for an adversary, a defender has to find all of them. Such a process is extremely tedious and error-prone, and depends on the considered fault models. Research in this area has seen numerous failures, mainly for not considering proper fault models. For example, Statistical Ineffective Fault Analysis (SIFA) [DEG+18, DEK+18] in 2018 and Fault Template Attacks (FTA) in 2020 have rendered most of the existing FA countermeasures ineffective [SBR+20]. In summary, the interaction between a fault model and a cipher is often complex and cipher-specific, which could only be explored with manual effort to date.

Quite evidently, designing and especially certifying FA countermeasures is not something that can be performed manually. Given that the attacks are complex, the first step is to understand the exploitable fault space (and, therefore, the exploitable fault models) in an unprotected cipher algorithm. Then, the design and testing of the countermeasures should proceed based on the exploitable faults in the unprotected version. Also, in the end, one should verify whether or not the inclusion of countermeasures adds some vulnerability. Surprisingly, only a handful of work have addressed this problem to date. Existing tools in this regard aim to discover vulnerabilities in either unprotected or protected algorithms. However, most of them work on a given fault-list based on some well-known fault model (byte/bit). The main problem with this approach is that it can be highly specific to certain fault models and may eventually result in a false sense of security. In particular, there is no guarantee that if a fault model is unexploitable for one cipher, it will not be exploitable for the other one.

## 1.1 Our Goals and Contributions

This work aims to address the problem of automated exploitable fault model identification in a block cipher or its protected implementation. This bridges one of the major gaps in the automation efforts for fault analysis. In order to mimic human expertise in this regard, we propose using reinforcement learning (RL) combined with a $t$-test [SKP+19]. While ALAFA maybe effective in analyzing the leakage due to a fault, the identification of the effective fault space remains an open problem. We note that the proposed approach is based on simulation data, and therefore, easy to realize for any block cipher without spending much time behind formalization (this is clearly an advantage over developing a formal approach in this regard). The simplicity of a simulation-based approach also leaves the scope for integrating several implementation-based constraints, eventually leading to application-specific security guarantees. As a potential example, we demonstrate that all the known exploitable fault models for AES (including the target rounds) can be identified automatically by the RL agent. Finally, the RL-based framework scales for analyzing protected implementations without significant changes. We establish this by exploring vulnerabilities in an FA countermeasure. The major contributions of this work are as follows.

1. We bridge the gap in state-of-the-art FA automation by devising an automated fault-model discovery methodology.

2. To the best of our knowledge, for the first time, RL has been utilized in the context of FAs. We note that RL is an appropriate choice in this, given its exploration power through complex search spaces. We note that one of the major challenges in FA testing is the size and complexity of the search space, which can be efficiently explored with RL.

3. We present two distinct use cases: unprotected AES and one protected version of AES. The success of RL on two use cases establishes that FA testing for both protected and unprotected implementations is feasible under a unified framework.

## 2 Methodology

In this section, we formulate the intricacies of the fault space exploration problem as a Markov decision process and introduce its five critical components $(\mathcal{X}, \mathcal{A}, P, R, \gamma)$. $\mathcal{X}$ is the set of states. **State** $s_t$ represents the current state of AES with injected faults. $\mathcal{A}$ is the set of actions. **Action** $a_t$ is the bit location where RL selected to inject a fault according to policy $\pi$. For **State transition** $P$, we can identify the next state $s_{t+1}$ in a deterministic manner by flagging the bits in the current state $s_t$ according to action $a_t$. Thus, the transition probability function reduces to a deterministic mapping $P\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. The reward function is $r_{t+1} = R(x_t, a_t)$. **Reward** $r_t$ relies on the t-test statistic (denoted as $t$), which is based on the current faulty state of AES. The t-test examines whether a faulty state is distinguishable from the uniform distribution [SMD18]. If t-test value is larger than a given threshold $t_{th}$, $r_t$ is 1, otherwise, it's 0. **Discount rate** $\gamma$, $0 \leq \gamma \leq 1$, discounts future rewards to their present value.

State vector $s_0$ is initialized to zeros. At time step $t$, the RL agent makes the current action $a_t$ of selecting $i_{th}$ bit in AES state to inject a fault. This changes the $i_{th}$ entry of next state vector $s_{t+1}$ to be flagged as a 1. Once fault injection is completed, we collect 10,000 faulty ciphertexts and their correct ciphertexts and calculate the t-test value based on these faulty ciphertexts, followed by obtaining reward $r_t$. In this way, the RL agent explores the fault space by taking actions and acquiring the corresponding reward. After certain interactions, the RL agent can obtain an optimal policy $\pi^*$ in the end.

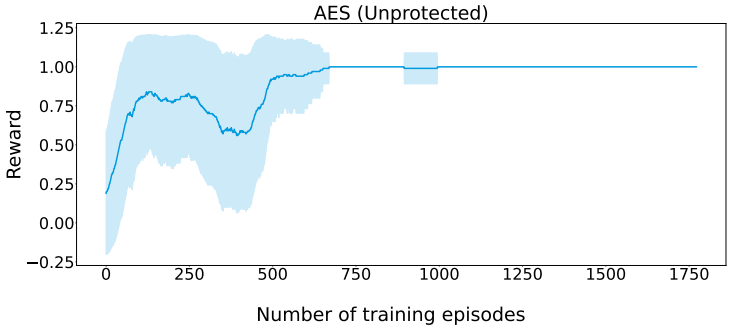


**Figure 1:** Rewards (moving average) vs. number of training episodes for evaluation on unprotected AES. The shaded region represents the standard deviation.

| Bit selected by RL | Location |
|---|---|
| 21, 23 | Byte 2 |
| 60, 62, 63 | Byte 7 |
| 66 | Byte 8 |
| 104, 106, 107, 108, 109 | Byte 13 |

**Table 1:** Bits selected by RL

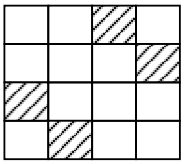

**Figure 2:** Fault Induced in $8^{th}$ round of AES

Figure 1 illustrates the reward recorded during the training of RL. As the training episodes increase, the RL agent learns to make a sequence of actions that favor the t-test statistic and finally converges to an optimal policy. We can get a t-test statistic of around

267 for the discovered fault model. Table 1 lists the fault model derived according to the converged optimal policy, and Figure 2 depicts the fault model presenting in the $8^{th}$ round of AES. This fault model is called the diagonal fault model, which was found by Saha *et al.* [SMC09]. Furthermore, the RL agent also discovers the bit fault model and byte fault model while exploring the fault space. This demonstrates that the RL agent can figure out all fault models researchers have discovered previously in the context of DFA of AES.

# 3 Experimental Results

## 3.1 Experiments Setup

We implement our work using Python 3.8.10 and conduct experiments on a 20.04 Ubuntu machine with an AMD 32-core 3.5GHz CPU processor and an NVIDIA A5000 24GB GPU. We select Proximal Policy Optimization [SWD+17] as our reinforcement learning algorithm. The custom RL environment and PPO algorithm were implemented based on the framework provided by `Stable-Baselines3` and `PyTorch1.6` packages. We employ vectorized environment in this package to compute multiple episodes of the RL pipeline in parallel, drastically reducing our training time. We set the parameter $t_{th}$ as 4.5.

## 3.2 Evaluation on AES block Cipher with Redundancy Countermeasure

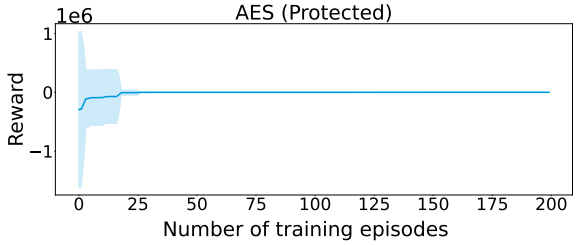


**Figure 3:** Rewards (moving average) vs. number of training episodes for evaluation on AES. The shaded region represents the standard deviation.

We first evaluated the proposed framework on unprotected AES. Figure 3 shows the reward plot. It demonstrated that the proposed RL architecture could tackle an unprotected AES. We then investigated RL's capability of fault space exploration on AES with one countermeasure in [SKP+19]. The countermeasure utilizes a redundant cipher computation creating two computational branches. The ciphertexts in these two branches are compared at the end of the computation. If they are different, the leakage can be observed by examining those ciphertexts. To evade this countermeasure, RL learned to inject the same set of faults sampled at those two computational branches automatically. The reward plot in Figure 3 demonstrates that our proposed framework can be extended to protected AES.

# 4 Conclusion

To the best of our knowledge, we report the first work in utilizing RL for fault space exploration in the context of FA of block cipher. Instead of relying on a human expert, our RL can achieve the same outcome by prior work but automatically. We demonstrated our proposed framework on two distinct use-cases: unprotected AES and AES with one countermeasure. This advancement can greatly reduce design effort for exploring effective fault space for FA of block cipher.

## References

[att22] Researcher hacks elon musk's starlink system with $25 homemade device, 2022. https://www.business-standard.com/article/international/researcher-hacks-elon-musk-s-starlink-system-with-25-homemade-device-122081300454_1.html.

[BBB+10] Alessandro Barenghi, Guido M Bertoni, Luca Breveglieri, Mauro Pellicioli, and Gerardo Pelosi. Low voltage fault attacks to AES. In *Proceedings of 3rd IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 7–12, Anaheim, CA USA, June 2010. IEEE.

[bit22] Glitching the keepkey hardware wallet, 2022. https://www.riscure.com/blog/glitching-the-keepkey-hardware-wallet.

[BLMR19] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: lightweight tweakable block cipher with efficient protection against dfa attacks. *IACR Transactions on Symmetric Cryptology*, 2019(1):5–45, 2019.

[BM16] Sarani Bhattacharya and Debdeep Mukhopadhyay. Curious case of rowhammer: flipping secret exponent bits using timing analysis. In *Proceedings of 18th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 602–624, Santa Barbara, USA, Aug 2016. Springer.

[CML+11] Gaetan Canivet, Paolo Maistri, Régis Leveugle, Jessy Clédière, Florent Valette, and Marc Renaudin. Glitch and laser fault attacks onto a secure AES implementation on a SRAM-based FPGA. *Journal of Cryptology*, 24(2):247–268, 2011.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Hannes Gross, Stefan Mangard, Florian Mendel, and Robert Primas. Statistical ineffective fault attacks on masked AES with fault countermeasures. In *Proceedings of 24th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 315–342, Brisbane, QLD, Australia, Dec 2018. Springer.

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):547–572, 2018.

[HFK+19] Sanghyun Hong, Pietro Frigo, Yiğitcan Kaya, Cristiano Giuffrida, and Tudor Dumitraș. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 497–514, 2019.

[HPP21a] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. Fault-enabled chosen-ciphertext attacks on kyber. In *International Conference on Cryptology in India*, pages 311–334. Springer, 2021.

[HPP21b] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. Fault-enabled chosen-ciphertext attacks on kyber. In *International Conference on Cryptology in India*, pages 311–334. Springer, 2021.

[Lu19] Yifan Lu. Attacking hardware AES with DFA. *arXiv preprint arXiv:1902.08693*, 2019.

[MOG+20] Kit Murdock, David Oswald, Flavio D Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based fault injection attacks against Intel SGX. In *Proceedings of 41st IEEE Symposium on Security and Privacy (S&P)*, pages 1466–1482, San Francisco„ USA, May 2020. IEEE.

[QWLQ19] Pengfei Qiu, Dongsheng Wang, Yongqiang Lyu, and Gang Qu. Voltjockey: Breaking sgx by software-controlled voltage-induced hardware faults. In *Proceedings of 4th Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6, Xi'an, P.R. China, Dec 2019. IEEE.

[RRB+19] Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Number "not used" once-practical fault attack on pqm4 implementations of nist candidates. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 232–250. Springer, 2019.

[SBR+20] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In *Proceedings of 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 612–643, Zagreb, Croatia, May 2020. Springer.

[SFK20] Majid Sabbagh, Yunsi Fei, and David Kaeli. A novel GPU overdrive fault attack. In *Proceedings 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, San francisco, USA, Sep 2020. IEEE.

[SJR+20] S. Saha, D. Jap, D. Basu Roy, A. Chakraborty, S. Bhasin, and D. Mukhopadhyay. A framework to counter statistical ineffective fault analysis of block ciphers using domain transformation and error correction. *IEEE Transactions on Information Forensics and Security*, 15:1905–1919, 2020.

[SKP+19] Sayandeep Saha, S Nishok Kumar, Sikhar Patranabis, Debdeep Mukhopadhyay, and Pallab Dasgupta. ALAFA: Automatic leakage assessment for fault attack countermeasures. In *Proceedings of the 56th ACM/IEEE Design Automation Conference (DAC)*, pages 136–142, Las Vegas, USA, June 2019. ACM.

[SMC09] Dhiman Saha, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. A diagonal fault attack on the advanced encryption standard. IACR Cryptology ePrint Archive, 2009.

[SMD18] Sayandeep Saha, Debdeep Mukhopadhyay, and Pallab Dasgupta. ExpFault: an automated framework for exploitable fault characterization in block ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(2):242–276, 2018.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[TMA11] Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. Differential fault analysis of the advanced encryption standard using a single fault. In *Proceedings of 5th Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication (WISTP)*, pages 224–233, Crete, Greece, June 2011. Springer.