Received: 17 May 2020 | Revised: 19 January 2021 | Accepted: 20 January 2021 | CAAI Transactions on Intelligence Technology

DOI: 10.1049/cit2.12027

ORIGINAL RESEARCH PAPER

Differential fault location identification by machine learning

Anubhab Baksi[1] | Santanu Sarkar[2] | Akhilesh Siddhanti[3] | Ravi Anand[4] | Anupam Chattopadhyay[1]

[1]Nanyang Technological University, Singapore

[2]Indian Institute of Technology, Chennai, India

[3]Georgia Institute of Technology, Atlanta, USA

[4]Indian Institute of Technology, Kharagpur, India

Correspondence

Anubhab Baksi, Nanyang Technological University - Yunnan Garden Campus, Singapore 636921.
Email: anubhab001@e.ntu.edu.sg

Abstract

As the fault-based attacks are becoming a more pertinent threat in today's era of edge computing/internet-of-things, there is a need to streamline the existing tools for better accuracy and ease of use, so that we can gauge the attacker's power and a proper countermeasure can be devised in the long run. In this regard, we propose a machine learning (ML) assisted tool that can be used in the context of a differential fault attack. In particular, finding the exact fault location by analysing the output difference (typically the XOR of the nonfaulty and the faulty ciphertexts) is somewhat nontrivial. During the literature survey, we notice that the Pearson's correlation coefficient dominantly is used for this purpose, and has almost become the defacto standard. While this method can yield good accuracy for certain cases, we argue that an ML-based method is more powerful in all the situations we experiment with. We substantiate our claim by showing the relative performances (we choose the commonly used multilayer perceptron as our ML tool) with two variants of Grain-128a (a stream cipher, and a stream cipher with authentication), the lightweight stream cipher LIZARD and the lightweight block cipher SIMON-32 (where the faults are injected at the fifth last rounds). Our results demonstrate that a common ML tool can outperform the correlation with the same training/testing data. We believe that our work extends the state-of-the-art by showing how traditional cryptographic methods can be replaced by a more powerful ML tool.

1 | INTRODUCTION

Fault attacks are a common type of techniques used in the cryptanalysis of primitives. This technique works by injecting a disturbance on a device while it is performing a cryptographic operation [1]. Typically, this disturbance can be induced by means of a power glitch, LASER shot etc. It has been shown in the study that such disturbance can be injected by an inexpensive equipment with a high precision [2]. With rise and spread of the internet-of-things/edge computing, small-scale devices performing cryptographic operations are almost ubiquitous; the fault attacks are becoming an increasing concern for the community.

Among all the fault attack techniques, the differential fault attack (DFA) [1, Section 5.1] is one of the most common techniques in the symmetric key cryptographic community. Under this model, the injected fault is able to flip one bit or more bits of the state of the cipher being carried out on the target device. By taking the XOR of the nonfaulty and the faulty outputs, the attacker can learn information about the secret key of the system. Since its first appearance, DFA is being used extensively to cryptanalyse a variety of ciphers, which are considered secure against the classical attacks.

Generally, it is considered that the exact location of the bits which are affected by the fault is not known to the attacker, while the target round is known. In other words, the attacker can precisely control the fault in the temporal domain but not in the spatial domain. This is termed as the random fault model [1, Section 2.3], and is generally more practical compared to the model where the exact fault location is known directly/chosen by the attacker. The problem of finding the exact location of fault is sometimes solved by carefully analysing the XOR of the

non-faulty and the faulty outputs of the cipher, particularly when the authors show a simulation instead of actually performing the attack (e.g. [3,4]).

Generally, finding the exact fault location is more complicated in case of stream ciphers and stream cipher-based designs where the usual attack can be conceptually thought to be composed of two distinct phases, for example [4,5]. In the first phase, the exact location of the fault injection is determined by analysing the effect of the fault propagation (termed as, signature) typically by using the correlation coefficient. With the information on the flipped bit, some automated tool (typically, a SAT solver, e.g. in [3,5]) is used to model the state, thereby recovering it. A similar attack procedure is applied against block ciphers as well, for example in [6].

Since finding the location of fault typically constitutes the first step of the attack and this is done mostly in an adhoc manner (see, e.g. [5]), we believe that the generalisation and automation of this step will be useful for future researchers. This motivates us to revisit several existing research and look for possible improvements. We observe that, finding the exact location of fault can be done by the machine learning (ML) tools–thus providing an edge compared to the existing correlation-based method; since ML tools are generally readily available, generalised, robust and easy-to-use. We use a multilayer perceptron (MLP) [7] for our choice of the ML tool, which is among the first and commonly used tools.

ML, although has found its application in a lot of areas of computer science, is relatively rare when it comes to cryptanalysis [8,9] (ML has found its application in the context of side channel analysis, for example in [10]). Thus our work is among the earliest that incorporates ML in cryptographic applications.

It can be mentioned that the scope of our work is limited in finding the exact location of the fault in the DFA setting at the state of the cipher (the round of the cipher at which the fault is affected, is assumed known to the attacker). Following the existing literature (e.g. [5]), we only consider that one bit is flipped by the fault, though our method is likely capable to work with multiple bit flip model as well. The next step in DFA is to use this information to mount a state recovery and/or key recovery attack, which is omitted here for brevity (we recommend to use already existing tools for this purpose, e.g. [4,5]). Also, we only consider cryptographic faults, and as such natural faults (e.g. due to ageing) are not within the scope. Furthermore, we keep the discussion on DFA countermeasures out of scope for the interest of brevity. In case DFA countermeasures are solicited; one may refer to, for example, [11] or [1, Section 7].

Our results do not constitute the upper limit for the ML tools. It is likely that with more training data and/or with more sophisticated ML tool (e.g. a deeper network), the performance can be further improved.

1.1 Our contribution

Generally the proponents of DFA assume the round, at which the fault is injected, is known to the attacker, but the exact location (at which a bit is flipped) is not known [12]. This makes the attack model more practical. However, in order to retrieve the secret information from the cipher, it becomes a necessity to learn the exact location of the fault (i.e. which location of the state is flipped because of the fault injection).

Thus, one fundamental problem with respect to DFA is identifying the precise bit which is flipped as a result of the fault injection. The current standard is, to use the correlation coefficient, as can be seen from the existing literature [5,6,12]. While more discussion in this regard is given in Section 2.2, it can be stated that, this method computes the mean of the XOR of the nonfaulty and the faulty outputs over multiple independent runs (the so-called signature) during the offline phase. Thus, the signature basically refers to a matrix, with elements from [0, 1]. At the online phase (i.e. when the exact location is to be found out), the XOR of faulty and nonfaulty output is computed. This gives a vector of binary elements. With each of the rows of the signature matrix, this vector is used to compute the corresponding correlation coefficient. The location (i.e. the row of the signature matrix) corresponding to which the correlation is maximum, is taken as the correct location of fault. It may be noted that this method is somewhat similar to the correlation power analysis (CPA) [13], which is used in the context of power/electromagnetic side channel analysis. In CPA, the exact value is found by locating the case for which the (absolute) correlation coefficient is maximum. The same procedure (i.e. the maximum correlation coefficient) is used in the identification of fault location too.

We propose to replace the correlation-based method (which comes as an adhoc approach) with a suitable supervised ML tools for this problem. Based on simulations of fault, our results (described in Section 3) show that an MLP, which is a common and one of the earliest supervised learning tools, can outperform the correlation-based method (although the same training/testing data are used) for a number of ciphers. We choose two variants of the well-studied GRAIN-128A cipher, one without authentication functionality and the other with authentication [14], the lightweight stream cipher LIZARD [15] and the lightweight block cipher SIMON-32 [16]. For all the ciphers, we show that an ML approach can outperform correlation. Essentially, our work directly improves the results of [5,6,12] among others, thus significantly contributing to the advancement of the state-of-the-art.

Note that, we assume the fault model that flips only one bit of the state (similar to previous works like [4]), and the fault is simulated. The effect of the fault propagation is then computed from the XOR of the faulty and the nonfaulty outputs.

2 | BACKGROUND

2.1 | Context of differential fault attack

Fault attacks have surely gained considerable attention of the cryptographic research community in recent times. New types of fault attack models, specialised countermeasures to thwart

those attacks, are being reported frequently, one may refer to [1] for more details. Most, if not all ciphers are shown vulnerable against fault attacks (unless equipped with a suitable fault protection mechanism).

The DFA model is the earliest in the symmetric key setting [17], and among the most commonly used models. A basic workflow of DFA on symmetric key ciphers is shown in Figure 1. Here the fault toggles a bit at a round (a nonfaulty round is shown by F, and that of the faulty by $F^\star$) near the end of the cipher execution [1, Section 5.1]. This creates a difference at the output (the difference is computed by $C \oplus C^\star$, where C is the nonfaulty output and $C^\star$ is the faulty output), from which the secret information can be retrieved.

As noted already, generally it is assumed that the attacker is not able to choose/decide the exact location for fault (i.e. which particular bit will be flipped). Therefore, the attacker attempts to find out the exact location of fault by analysing the XOR of the nonfaulty and the faulty outputs. Thus our model can be summarised by the following points:

- The attacker can inject a one-bit fault, thereby flipping that particular bit of the state. Such precision of fault location can be achieved, for example, by LASER shot [2].
- Each bit of the state is equally likely be flipped as a result of the fault, and the location of the fault injection is unknown to the attacker.
- The attacker has precise control over the round of fault injection.

2.2 | Correlation-based method for identifying fault location

Correlation-based method to find fault location is the practical standard for finding the location of the fault, as can be seen from several research works [4,5,12,18]. For GRAIN-128A, LIZARD and SIMON-32, previous results on correlation-based fault location identification are reported, respectively in [5,6,12].

A summarised description for the correlation-based differential fault location identification is given here for completeness, one may refer to, for example, [12] for more information. The following notations are used:

- the fault-free key-stream sequence of length ℓ which the adversary has access to: $z_0, z_1, \ldots, z_{\ell-1}$;
- the fault location, f;
- the ℓ-length key-stream obtained after injecting a fault (faulty key-stream): $z_0^{(f)}, z_1^{(f)}, \ldots, z_{\ell-1}^{(f)}$.

The fault identification procedure can be roughly classified into two phases, namely offline and online. Overall, the attacker (Eve) at first computes the offline phase, which can be done through simulation. We assume she can control the target device so that she can perform the fault injection with precision. Being equipped with the information from this phase, the attacker moves to the actual online phase of the attack.

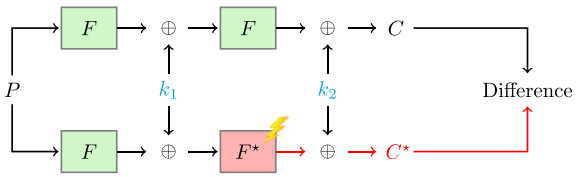


FIGURE 1 Basic idea of DFA

2.2.1 Offline phase

The attacker precomputes the signature vector $\mathscr{O}^{(f)}$ for each fault location f of the cipher. The signatures are prepared by observing the probability of fault-free key-stream bits being not equal to faulty key-stream bits over several randomly generated keys and nonces: $\mathscr{O}^{(f)} = \{q_0^{(f)}, q_1^{(f)}, \ldots, q_{\ell-1}^{(f)}\}$ where $q_i^{(f)} = \Pr(z_i \neq z_i^{(f)})$.

2.2.2 Online phase

The attacker injects a fault in an unknown location g, and calculates the trail $\Gamma^{(g)}$ of the fault location as follows: $\Gamma^{(g)} = \{\gamma_0^{(g)}, \gamma_1^{(g)}, \ldots, \gamma_{\ell-1}^{(g)}\}$ where $\gamma_i^{(g)} = \Pr(z_i \neq z_i^{(g)})$.

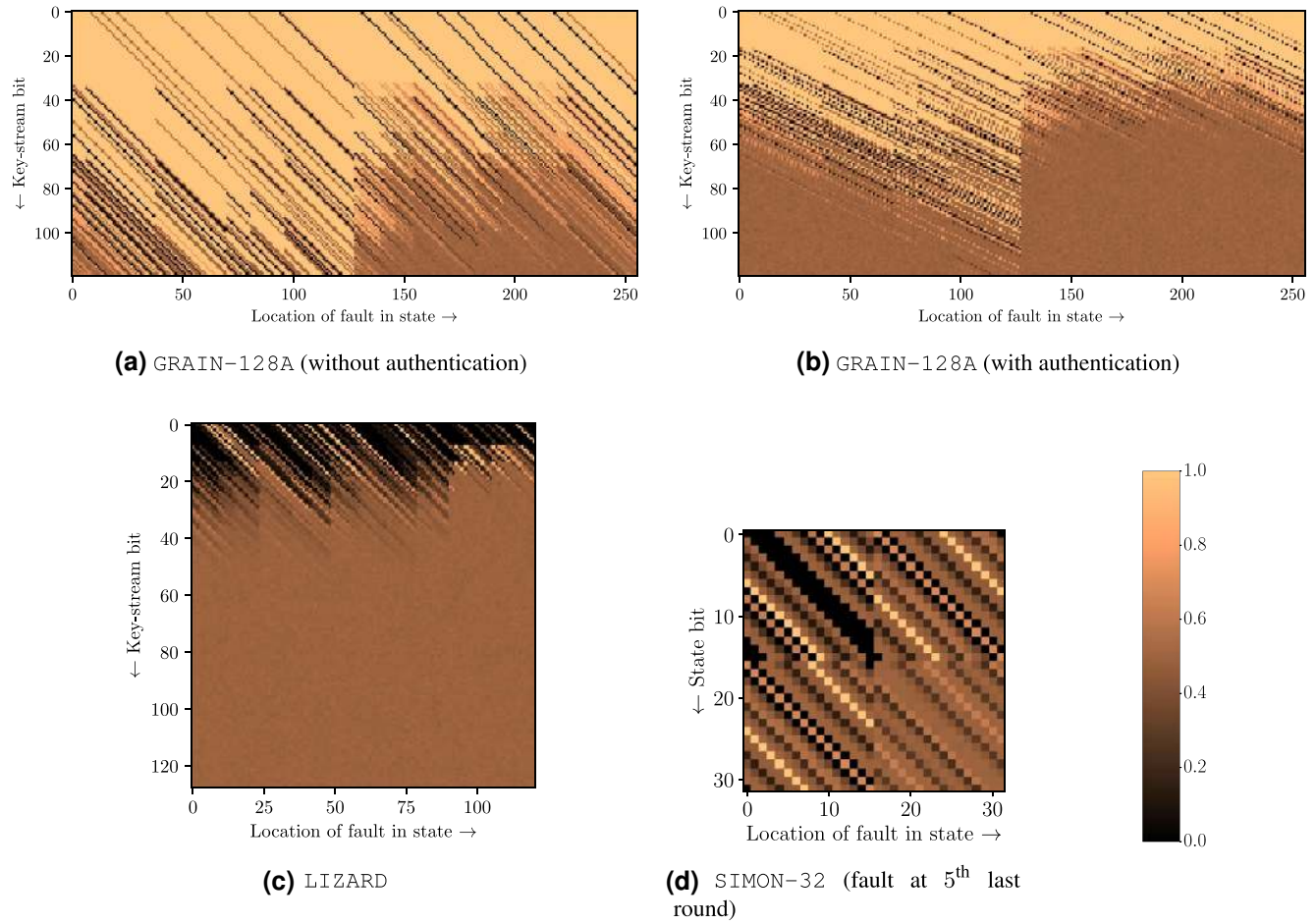
Hence, the fault signature is a matrix of values in [0, 1]. The number of columns of the matrix is the same as the number of key-stream bits and that of rows is the same as the number of fault locations (typically the entire state).

The final goal for the attacker is to identify g. The value of f, for which $\mathscr{O}^{(f)}$ best matches the trail $\Gamma^{(g)}$ obtained, and corresponds to the correct fault location. For checking this, correlation coefficient is shown to work with a good accuracy [12]. The attacker calculates the correlation between the signature $\mathscr{O}^{(f)}$ and trail $\Gamma^{(g)}$ for all possible values of f. This algorithm provides the value of g with a reasonably high accuracy. The same algorithm is repeated to identify fault location for all faulty key-stream sequences. The equations are then gathered and solved using an automated tool, typically a SAT solver (e.g. [3]).

However, often the correct location does not have a maximum correlation. It may so happen that the correlation for the correct fault location is lower than that of some other location. In this case, the location identified by the correlation-based method would be wrong. The rank metric measures the number of locations where the correlation coefficients of those locations are greater than or equal to the correlation coefficient of the correct location. Hence, if the correlation coefficient of the correct location is maximum, it has rank 1. Hence, if the rank is small (close to 1), then the performance of the method can be considered good. We also extend this notion of rank to ML-based fault location finding to have a comparison of performances.

2.2.3 Visualisation of fault signatures

As signatures for a cipher are indeed a matrix with entries from [0, 1], they can be pictorially represented. It can be



(a) GRAIN-128A (without authentication)

(b) GRAIN-128A (with authentication)

(c) LIZARD

(d) SIMON-32 (fault at 5th last round)

FIGURE 2 Visualisation of signatures

mentioned that a signature works as a visual reference to the level of diffusion the cipher achieves. For the ideal case, the entries in the signature will be (close to) 0.5, significant deviation from that can be taken as an indication of low diffusion.

Figure 2 shows examples with the ciphers. Note that, the case for Figure 2(d), that is SIMON-32with differential fault being applied at the fifth last round is considered in [6].

To the best of our knowledge, the visualisation of the signatures is done with three-dimensional graphs throughout the study (e.g. [4]). This makes it hard to interpret. Instead of that, we use a two-dimensional structure with heatmap so that it is easier to interpret for the reader.

While it is hard to draw a direct conclusion based on the signature, it can still be inferred that the overall nature (i.e. how far or near entries are from 0.5) can play a role in the accuracy of finding the exact fault location. For example, in Figure 2(c) (which shows the case for LIZARD), most of the entries are close to 0.5; in contrast to Figure 2(d) (the case for SIMON-32), where most of the entries are further away from 0.5. Thus the performances for the correlation as well as ML-based methods (described in Section 3) are comparatively poor for LIZARD than that of SIMON-32.

2.3 | Fundamentals of artificial neural network

ML can be loosely defined by a collection of various types of algorithms, of which artificial neural networks (ANNs) [7] are of particular interest. ANNs are algorithms used for fitting a model to a given data that can perform efficiently tasks like classification or regression, which are generally considered difficult for a computer. ANNs are capable of finding inherent characteristics of the training data by iterating through it repeatedly and gradually adjusting its parameters, until these parameters are finally stabilised. Once training is completed, the model is validated against the testing data.

The basic processing unit of an ANN is termed as a neuron, which is inspired from the biological neuron found in brain cells. The neurons are arranged in a series of layers. More depth of layers generally makes the ANN capable of handling more complex data.

Here we use the basic forward propagation ANN. More precisely, we use the MLP, which is among the oldest ANNs. In an MLP, all the neurons at the previous layer are connected to all the neurons at the next layer (those layers are known as, dense layers). A schematic of an ANN can be found in Figure 3.

3 | EVALUATION

3.1 | Details of machine learning model

We use a five-layer neural network with TensorFlow[1] as the back-end and Keras[2] API. The layers can be briefly described as:

- **Layer 1.** The first layer is a dense layer with a dropout rate of 0.2 and activation function as rectifier (ReLU). The number of neurons in the layer equals to the number of output bits available for processing (e.g. the number of key-stream bits).
- **Layers 2, 3, 4.** The second, third and fourth layers are dense layers with rectifier as activation functions. The number of neurons is respectively 252, 202 and 160.
- **Layer 5.** The final layer is a dense layer with softmax activation (one-hot encoding). The number of neurons in the layer equals to the number of possible fault locations, that is the state size in our case. The neuron with the maximum firing rate is taken as the predictor of the class.

We compile the model with the Adam optimiser [19], sparse categorical cross-entropy as the loss function, accuracy as the metric and with eight epochs[3].

The hyperparameters are chosen somewhat arbitrarily, akin to [8,9]. While it is possible to improve the accuracy by fine-tuning the hyperparameters, we leave it open for future research.

3.2 | Results

For our experiment, we take 120 bits of the key-stream from starting of PRGA of GRAIN-128A (for both with and without authentication variants), and 128 bits for LIZARD. The fault in each case is injected at the first round of PRGA. In case of SIMON-32 (unkeyed permutation), we assume that the fault is injected at the fifth last round (similar to [6]).

Relative performances of the ML and correlation-based approaches with maximum and average ranks as the benchmark are presented in Figure 4 for GRAIN-128A (without authentication), in Figure 5 for GRAIN-128A (with authentication), in Figure 6 for LIZARD and in Figure 7 for SIMON-32. Notice from Figure 7(a), the ML-based method works with full (1.0000) accuracy.

Some additional information is also shown in Table 1; namely the size of the training and testing data, the accuracy (ration of predictions that are at rank 1 compared to the total number of predictions), average and maximum ranks and number of wrong classification. For example, it can be seen from Table 1 that; with the $2^{18.000}$ training data, the $2^{14.551}$ testing data, and the accuracy for GRAIN-128A without authentication for ML-based method is 0.9988, whereas the same for the correlation-based method is 0.9970.

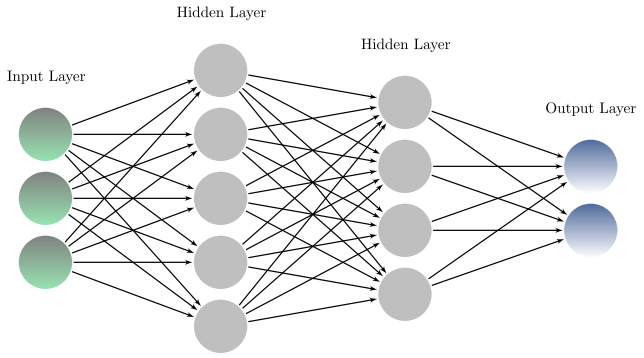


FIGURE 3 Schematic of a multilayer perceptron

From each pair-wise comparison, it can be observed that ML outperforms correlation with the same training and testing data, in each of the metrics. However, for certain ciphers, the advantage of using ML is more apparent than other ciphers. For example, we observe that the ML-based method achieves full accuracy for SIMON-32 (can be seen from Table 1 as well as Figure 7(a)), whereas the same for the correlation-based method is 0.7113.

As for the complexity of the training/testing data, we note the following. The offline (training) phase can be simulated, hence the data required ($<2^{19}$) is not a major limiting factor. The size of the testing data used here ($<2^{15}$) can likely be reduced.

We would like to emphasise that the graphs in Figures 4, 6, and 7 show the ranks at which the correct prediction occurs. For the best case scenario, the rank should be 1 (as in Figure 7 (a) for SIMON-32), which would indicate that all the fault locations are identified at the first predicted class. Higher ranks show the accuracy of the model is worse; for example, a comparison of Figure 7(a) with Figure 7(b) shows that the ranks are higher for the correlation-based method, meaning the correlation-based method performs worse.

4 | CONCLUSION

Here we apply an ML method to the problem of finding the location of a fault in a symmetric key cipher. We consider one of the most common faults, DFA, where the fault effectively flips one bit (location of this flipped bit is unknown to the attacker). Previous research works use a correlation coefficient-based method for this purpose, for example, [5,12], to name a few. Instead of that, we propose to use a supervised ML-based approach. We show that a five-layer (i.e. with three middle layers) MLP-based method which can outperform that of the correlation while both are trained and tested with the same data. The validation of our claim is done through four ciphers, namely for two variants of GRAIN-128A (one as a stream

[1] https://www.tensorflow.org/
[2] https://keras.io/
[3] The trained models (in '.h5' format) and the signatures (in '.npy' format) are available online: https://entuedu-my.sharepoint.com/:f:/g/personal/anubhab001_e_ntu_edu_sg/EmDCnoe-sllKrmLKvQuhwkgBByxY7DDab8h4IdxjVsw6Vg?e=1ptqOX.

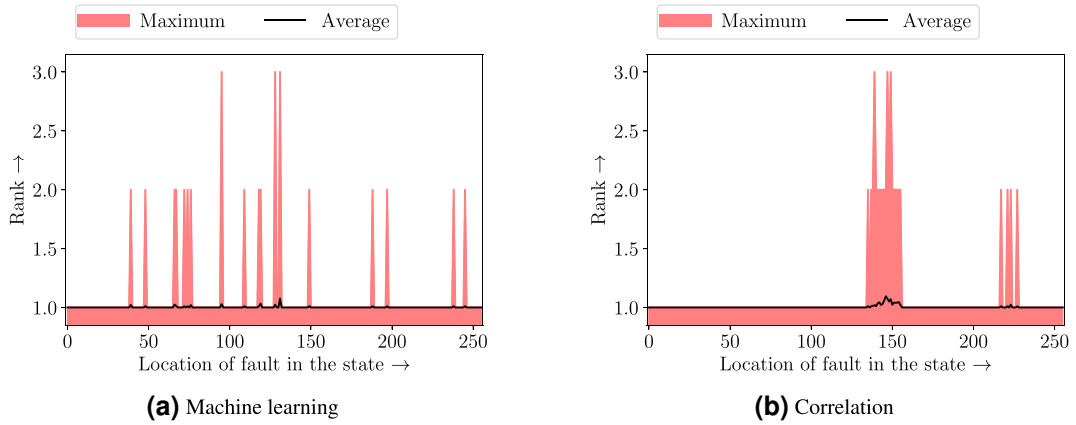


FIGURE 4 Performances for machine learning and correlation-based methods on GRAIN-128A (without authentication)

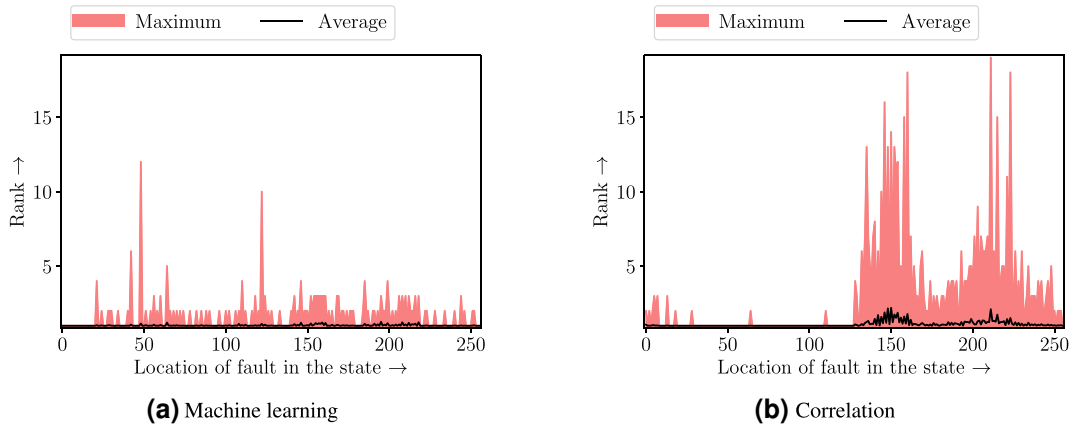


FIGURE 5 Performances for machine learning and correlation-based methods on GRAIN-128A (with authentication)

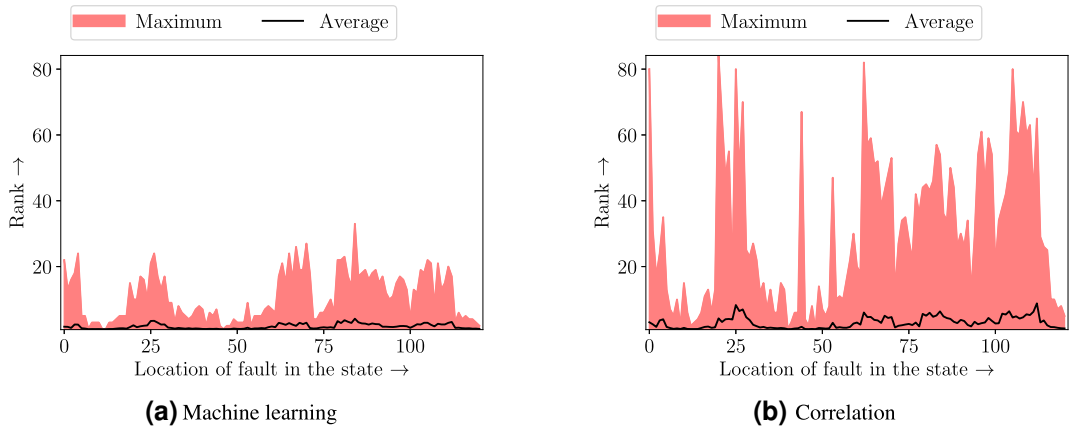


FIGURE 6 Performances for machine learning and correlation-based methods on LIZARD

cipher, while the other as a stream cipher with authentication) [14], the lightweight stream cipher LIZARD [15] and the lightweight block cipher SIMON-32 [16] (the unkeyed permutation variation).

We note that our work does not pose the upper bound of ML-based analysis. It is likely that the performance can be improved by choosing a more sophisticated model (currently the hype-parameters of our model are somewhat arbitrarily

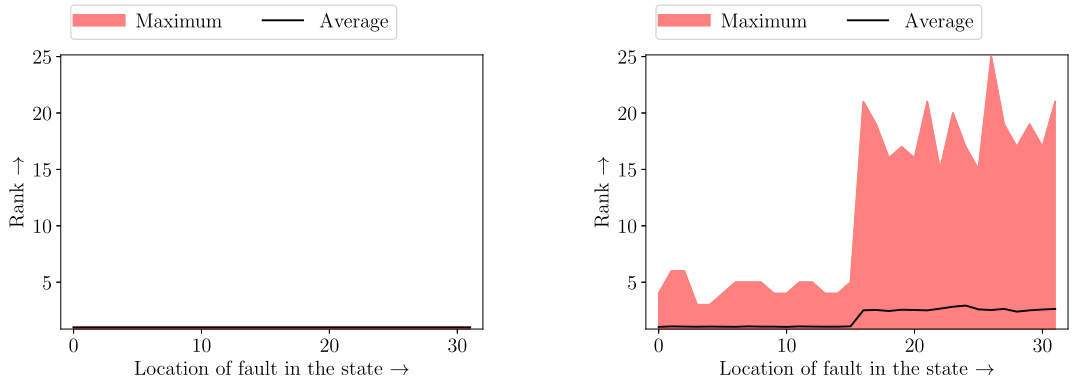


FIGURE 7 Performances for machine learning and correlation-based methods on SIMON-32 (fault at fifth last round)

TABLE 1 Summary of experiments for differential fault location identification

		Size (in Power of 2)		Machine learning (ours)				Correlation [12]			
					Rank*				Rank*		
	Cipher	Training	Testing	Accuracy	Average	Maximum	# Wrong identification	Accuracy	Average	Maximum	# Wrong identification
GRAIN-128A	W/o Auth.	18.000	14.551	0.9988	1.0013	3	27	0.9970	1.0032	3	72
	w/Auth.	18.000	14.551	0.9799	1.0240	12	482	0.9448	1.1024	19	1324
	LIZARD	18.111	14.047	0.7173	1.8128	33	4883	0.6347	2.9512	84	6186
SIMON-32	Fifth last round	18.226	14.180	1.0000	1.0000	1	0	0.7113	1.8271	25	5359

* denotes lower is better, 1 is ideal.

chosen, similar to [8,9]), and/or using more training/testing data. Continuing in this line of work, one may be interested in a multibit fault (where the fault flips more than one state bits), or the sensitivity of one particular ML model with respect to various ciphers.

ORCID

Anubhab Baksi https://orcid.org/0000-0002-5639-7372

REFERENCES

1. Baksi, A., et al.: Fault attacks in symmetric key cryptosystems, IACR Cryptology ePrint Archive (2020). https://eprint.iacr.org/2020/1267
2. Agoyan, M., et al.: How to flip a bit?. IEEE 16th International on-line testing symposium, pp. 235–239. (2010)
3. Dey, P., Adhikari, A.: In: Improved multi-bit differential fault analysis of Trivium, INDOCRYPT 2014, New Delhi, India, Proceedings, pp. 37–52. Corfu, Greece (2014). https://doi.org/10.1007/978-3-319-13039-2_3
4. Maitra, S., et al.: Key recovery from state information of sprout: Application to cryptanalysis and fault attack, IACR Cryptology ePrint Archive (2015). http://eprint.iacr.org/2015/236
5. Siddhanti, A., et al.: Differential fault attack on grain v1, acorn v3 and lizard. In: International conference on security, privacy, and applied cryptography engineering, pp. 247–263 (2017)
6. Anand, R., et al.: Differential fault attack on SIMON with very few faults. In: Progress in cryptology - INDOCRYPT 2018 - 19th International conference on cryptology in India proceedings, pp. 107–119. New Delhi (2018). https://doi.org/10.1007/978-3-030-05378-9_6
7. Haykin, S.: Neural Networks and Learning Machines, 3rd ed. Pearson, USA (2008)
8. Baksi, A., et al.: Machine learning assisted differential distinguishers for lightweight ciphers. IACR Cryptology ePrint Archive. (2020). https://eprint.iacr.org/2020/571
9. Baksi, A., et al.: Following-up on machine learning assisted differential distinguishers. SILC workshop – security and implementation of lightweight cryptography (2021). https://www.esat.kuleuven.be/cosic/events/silc2020/wp-content/uploads/sites/4/2020/10/Submission4.pdf
10. Perin, G., Picek, S.: On the influence of optimizers in deep learning-based side-channel analysis (2020). https://eprint.iacr.org/2020/977. Cryptology ePrint Archive, Report 2020/977
11. Baksi, A., Saha, D., Sarkar, S.: To infect or not to infect: A critical analysis of infective countermeasures in fault attacks, IACR Cryptology ePrint Archive (2019). https://eprint.iacr.org/2019/355
12. Sarkar, S., et al.: Probabilistic signature based generalized framework for differential fault analysis of stream ciphers. Cryptogr. Commun. 9(4), 523–543 (2017). https://doi.org/10.1007/s12095-016-0197-2
13. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. Cryptographic hardware and embedded systems - CHES 2004, pp. 16–29.6th International workshop, Cambridge (2004). https://doi.org/10.1007/978-3-540-28632-5_211-13

14. Ågren, M., et al.: Grain-128a: a new version of grain-128 with optional authentication. Intl. J. Wireless Mobile Comput. 5(1), 48–59 (2011). https://doi.org/10.1504/IJWMC.2011.044106
15. Hamann, M., Krause, M., Meier, W.: *LIZARD* - a lightweight stream cipher for power-constrained devices, IACR Cryptology ePrint Archive (2016). http://eprint.iacr.org/2016/926
16. Beaulieu, R., et al.: SIMON and SPECK: Block ciphers for the internet of things, IACR Cryptology ePrint Archive (2015). http://eprint.iacr.org/2015/585
17. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski, J., Burton, S. (eds.) Advances in Cryptology - CRYPTO '97. Vol. 1294 of Lecture Notes in Computer Science, pp. 513–525. Springer Berlin Heidelberg (1997). http://dx.doi.org/10.1007/BFb0052259
18. Maitra, S., Siddhanti, A., Sarkar, S.: A differential fault attack on plantlet. IEEE Trans. Comput. 66(10), 1804–1808 (2017)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). arXiv preprint arXiv:14126980

How to cite this article: Baksi A, Sarkar S, Siddhanti A, Anand R, Chattopadhyay A. Differential fault location identification by machine learning. *CAAI Trans. Intell. Technol.* 2021;6:17–24. https://doi.org/10.1049/cit2.12027