Postprint: Power Analysis of Block Cipher Algorithms Based on MLP Neural Networks

Authors: Wang Kai, Cai Juesong, Yan Yingjian

Date: 2020-09-28T00:00:00+00:00

Abstract

With the widespread application of embedded cryptographic devices, Side Channel Analysis (SCA) has become one of their security threats. By analyzing leakage information during the physical implementation of cryptographic algorithms to achieve key recovery, the security of cryptographic algorithm implementations can be evaluated. Multi-layer Perceptron (MLP) is a type of artificial neural network structure. To streamline the MLP network structure for power analysis and reduce the model' s training parameters and training time, research was conducted on MLP neural network models based on Hamming weight (HW) and bit-based approaches, reducing the output categories from 256 classes to 9 classes and 2 classes, respectively. By collecting power traces during the operation of the AES cryptographic algorithm, the proposed MLP neural network was trained and tested. Experimental results demonstrate that this model, while ensuring prediction accuracy, can reduce the training parameters of the MLP neural network by 84% and the training time by 28%, and reduces the number of power traces required in the key recovery phase, requiring only a minimum of 1 power trace to complete the recovery of the full AES algorithm key. Experiments validate the effectiveness of the model, and using this model, the security of block cipher algorithm implementations can be analyzed and evaluated.

Full Text

Preamble

Vol. 38 No. 3
Application Research of Computers
ChinaXiv Cooperative Journal

Research on Side-Channel Analysis of Block Cipher Algorithms Based on MLP Neural Networks

Wang Kai[1],[2], **Cai Juesong**[1], **Yan Yingjian**[1]
(1. Strategic Support Force Information Engineering University, Zhengzhou 450001, China;
2. 32125 Troops of PLA, Jinan 250001, China)

Abstract: With the widespread deployment of embedded cryptographic devices, side-channel analysis (SCA) has emerged as a significant security threat. By analyzing leaked information during the physical implementation of cryptographic algorithms, attackers can recover secret keys and evaluate the security of these implementations. Multi-Layer Perceptron (MLP) is a fundamental artificial neural network architecture. To streamline MLP networks for power analysis and reduce training parameters and time, this paper investigates MLP-based models using Hamming weight (HW) and bit-level representations, reducing output classes from 256 to 9 and 2 categories respectively. Power traces were collected during AES algorithm execution to train and test the proposed MLP models. Experimental results demonstrate that the proposed models reduce training parameters by 84% and training time by 28% while maintaining prediction accuracy, and significantly decrease the number of power traces required for key recovery. In optimal cases, only a single power trace is sufficient to recover the complete AES key. The effectiveness of these models is validated experimentally, confirming their applicability for analyzing and evaluating the security of block cipher implementations.

Keywords: side-channel analysis; deep learning; multi-layer perceptron (MLP); cryptographic chips; AES

0 Introduction

Embedded devices such as smart cards, RFID tags, and various IoT devices have become ubiquitous in modern life. These devices employ cryptographic algorithms to perform encryption and decryption operations, thereby protecting data security. However, during cryptographic algorithm execution, secret information processed by the device leaks through power consumption [1], timing [2], electromagnetic emanations [3], and other side channels. Since this leakage depends on both the processed data and the operations performed, sensitive information can be recovered through analysis of these side-channel signals.

Side-channel analysis was first introduced by Kocher in 1996 [2], who demonstrated key recovery through timing analysis and subsequently proposed a more powerful and general technique [1] known as Differential Power Analysis (DPA). Later, Mulder et al. extended differential analysis to Differential Electromagnetic Analysis (DEMA) [4,5] and applied it to other cryptographic algorithms such as ECC and RSA. Beyond simple and differential analysis, template attacks are considered the most effective approach. They assume the adversary possesses an identical target device and has full control over it, enabling precise

modeling of leakage signal statistical characteristics to extract secret information hidden in the device' s side-channel leakage.

In recent years, the cryptographic community has explored novel cryptanalytic methods based on machine learning and deep learning, posing new threats to the security of cryptographic implementations. Liu et al. [6] employed Support Vector Machines (SVM) to analyze electromagnetic leakage for key recovery. Lerman et al. [7] experimentally verified that SVM can recover keys from masked AES implementations.

As a branch of machine learning, deep learning utilizes deep neural networks to learn features from complex data and make decisions on new datasets, offering powerful feature extraction and classification capabilities. It has become a research hotspot, with numerous studies demonstrating the effectiveness of deep neural networks in SCA. Maghrebi et al. [8] pioneered the application of deep learning to cryptographic implementations, applying MLP and Convolutional Neural Networks (CNN) to side-channel analysis. In [9], the authors proposed a neural network-based SCA method to break the masked AES implementation from the DPA Contest V4. Reference [10] evaluated CNN performance under protected implementations and misaligned trace conditions. Benadjila et al. [11] provided experimental guidance on hyperparameter selection for MLP models, further demonstrating the power of deep learning in profiled SCA.

This paper extends the MLP neural network model proposed in [11] by addressing its shortcomings: excessive parameters, complex layer connections, and long training times. We optimize and streamline the model architecture by reducing the output layer from 256 classes to 9 and 2 classes, proposing the Hamming weight-based MLP model (HW-MLP) and the bit-based MLP model (bit-MLP).

1.1 Profiled Power Analysis

Profiled power analysis operates under the following assumptions: the adversary possesses two identical cryptographic devices—a profiling device and a target device. The analyst has full control over the profiling device, enabling precise acquisition and characterization of its power consumption during operation. The target device runs the encryption algorithm with an unknown key $k^* \in K$, and the objective is to recover the key byte k^*. Profiled side-channel analysis thus proceeds in two phases: the profiling phase and the analysis phase, corresponding to training and testing in deep learning.

1) Profiling Phase: Power traces are collected from the profiling device to build a specific leakage model using key-dependent leakage information. For each possible $k \in K$, the analyst collects N power traces $P_k = \{T_i(k)|i = 1, \ldots, N\}$ to form the training set and estimates the probability distribution function ξ as e_k:

$$e_k = \hat{\xi}[T|P, K = p, k]$$

where T represents the collected power trace dataset, and trace t_i is obtained under known plaintext p_i and key k_i. The estimate e_k is computed from the training set $D_p = \{t_i, p_i, k_i\}, i = 1, \ldots, S_p$.

2) Analysis Phase: Power traces are collected from the target device and classified based on leakage information. The test set $D_a = \{t_i, p_i\}, i = 1, \ldots, S_a$ is used to recover the correct key. The ultimate goal is to recover key k^* by identifying the highest probability distribution estimate e_k from the test set. Therefore, the model must correctly distinguish among estimates e_k for all possible keys $k \in K$. In practice, the maximum likelihood function is used to compute e_k by calculating the log-likelihood estimate $L(K)$ for each candidate key $k \in K$:

$$L(K) = \sum_{i=1}^{S_a} \log \hat{\xi}[t_i | p_i, k]$$

where $L(K)$ is the log-likelihood probability corresponding to candidate key k. The value $k \in K$ that maximizes $L(K)$ for traces collected from the target device is identified as the correct key.

1.2 Non-Profiled Power Analysis

Non-profiled power analysis employs statistical methods to detect correlations between leakage information and sensitive variables. Power analysis of cryptographic algorithms exploits the fact that a device' s instantaneous power consumption depends on both the data being processed and the operations being executed. Non-profiled methods primarily include Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Correlation Power Analysis (CPA) [12]. This type of SCA corresponds to a weaker adversary model where the analyst only has access to physical leakage captured from the target device for key recovery.

1.3 MLP Neural Networks

Multi-Layer Perceptron (MLP) is a neural network composed of multiple perceptron units [14,15], as shown in [Figure 1: see original paper]. All perceptrons in each layer are connected to all perceptrons in the next layer. An MLP consists of an input layer, an output layer, and a series of intermediate layers (hidden layers), with each layer comprising one or more perceptron units. MLP weights and biases are trainable parameters updated during stochastic gradient descent.

This paper focuses on MLP neural networks, which can be represented as a function F composed of multiple linear functions and nonlinear activation functions. This function offers high computational efficiency, bounded derivatives, and ease of differentiation. In summary, MLP can be expressed as:

$$F(x) = s(\lambda_n(\sigma_{n-1}(\lambda_{n-1}(\cdots \sigma_1(\lambda_1(x))))))$$

where λ_n represents fully connected layers, σ_i are activation functions, and s is the Softmax function.

The MLP neural network forms a grid-like structure of neurons divided into multiple connected layers. The value of each neuron is:

$$f_j^l = f\left(\sum_i w_{ij}^{l-1} f_i^{l-1} + b_j^l\right)$$

where w represents connection weights between adjacent layers, b is the neuron' s bias value, and f is the activation function. Neurons in the current layer are functions of the output values of all connected neurons in the previous layer. Common activation functions include ReLU, Sigmoid, Tanh, and Softmax.

The core principle of the MLP network algorithm is to compute errors through forward propagation and then adjust weights w through backpropagation to obtain an optimal model. During backpropagation, stochastic gradient descent is typically used to modify weights. Gradient descent computes gradients of the loss function with respect to all internal variables and performs backpropagation. Internal variables are typically weights adjusted according to the steepest descent direction of the loss function surface [16].

Neural networks are computationally intensive technologies that require significantly more time and memory resources compared to traditional methods like DPA and CPA. However, they offer distinct advantages for side-channel power analysis [17]:

a) No manual feature point selection is required; convolutional and fully connected layers automatically extract and identify features relevant to the traces.

b) Convolutional layers can extract features independent of their position in the data, enabling deep neural networks to counter unstable clock cycles and jitter introduced by random delay countermeasures.

c) As highly parameterized models, deep neural networks can optimize classification accuracy through hyperparameter tuning, thereby enhancing side-channel analysis performance.

d) Deep neural networks can implement highly complex functions, offering some capability to analyze widely adopted countermeasures such as masking and shuffling strategies.

1.4 Hamming Weight Model

The Hamming weight model counts the number of 1 bits in data stored in registers at a given time, characterizing register power consumption based on this count [12]. For a k-bit binary data value, the Hamming weight model is formally expressed as:

$$HW(V) = \sum_{i=0}^{k-1} v_i$$

where $V = (v_{k-1} \ldots v_1 v_0)$ and $v_i \in \{0, 1\}$. This model is typically used to characterize power consumption in devices with pre-charge logic, where all register bits are set to 0 or 1 before data changes.

Power consumption models simulate actual power consumption; higher simulation accuracy enables stronger capability to recover sensitive or key information. The Hamming weight model can be applied when the analyst has little or no knowledge of the cryptographic chip's netlist. In this model, the analyst assumes a linear relationship between power consumption and the number of active bits in processed data. However, in practice, Hamming weight is not perfectly correlated with power consumption, being applicable only in specific scenarios.

In AES side-channel analysis, the output of an S-box is typically selected as the analysis point. The Hamming weight of the output value at this point usually exhibits a linear relationship with the leakage value, as shown in [Figure 2: see original paper], where power trace colors transitioning from dark to light indicate this linear relationship.

2.1 Power Analysis Flow

The overall flow of MLP neural network-based power analysis is illustrated in [Figure 3: see original paper]. The specific steps are as follows:

a) **Data Acquisition and Preprocessing:** In traditional template attacks and machine learning-based SCA, data preprocessing primarily involves dimensionality reduction. Raw power trace data typically exhibits high noise and dimensionality, requiring preprocessing to reduce computational complexity. However, deep learning-based SCA can omit dimensionality reduction and directly preprocess the dataset. Collected power traces are divided into training and test datasets: the training set is used for model training and profiling, while the test set is used for model evaluation and key recovery.

b) **Model Training:** The model is trained using the training data, typically employing cross-validation to assess training effectiveness and compare different deep neural network or machine learning algorithms. In our experiments, we compare against the Support Vector Machine (SVM) algorithm.

c) **Model Evaluation:** Multiple evaluation strategies are used to assess model performance or select optimal parameters for parameterized models. In power analysis, primary evaluation metrics include: prediction accuracy, key guessing entropy, training time, number of traces required for key recovery, and computational resource consumption.

d) **Key Recovery:** The key recovery process typically employs a "divide-and-conquer"strategy, recovering each of the 16 AES key bytes individually. For Hamming weight-based models, possible key values can be calculated from predicted Hamming weights and known plaintext data. Using multiple traces with different Hamming weights and plaintexts, the key space is gradually narrowed through intersection operations on candidate values until the subkey is recovered.

2.2 Analysis Model Selection

Similar to traditional side-channel analysis methods such as DPA, CPA, and template attacks, deep learning-based power analysis requires defining a leakage model. MLP neural network-based analysis operates under supervised conditions, so training and test traces must be labeled according to the leakage model. The number of neural network output classes varies depending on the selected label values. When analyzing block ciphers, the S-box output is typically chosen as the analysis point because this nonlinear transformation consumes significant power and provides clear distinctions between different data values.

If the leakage model targets the S-box output during the first round of AES encryption, as shown in [Figure 4: see original paper], the neural network output categories can be defined as:
a) **Identity (ID) model:** $2^8 = 256$ classes;
b) **Hamming weight (HW) model:** 9 classes;
c) **Single-bit model:** 0 or 1, totaling 2 classes.

In our experiments, we use the ID, HW, and bit models for MLP neural network training, referred to as ID-MLP, HW-MLP, and bit-MLP models respectively.

2.3 ID-MLP Model

The ID-MLP model is designed to adapt the MLP model from Benadjila et al. [11] to our experimental platform, optimizing its parameters for optimal analysis performance. This model serves as a reference for comparison with our proposed models.

Since each collected power trace is a time series, the MLP input layer size equals the number of sampling points in the trace. The output layer contains 256 neurons corresponding to possible S-box output values. The detailed structure is shown in . During training, labels are the S-box output values (256 possibilities), enabling direct key byte recovery through a divide-and-conquer approach for all 16 key bytes.

2.4 HW-MLP Model

Compared to ID-MLP, the HW-MLP model reduces the number of hidden layers and neurons per layer, increases the learning rate, and uses the tanh activation function. The layer design employs progressively decreasing neuron counts, creating an inverted trapezoidal structure. Experiments demonstrate that this architecture significantly improves training and test accuracy while substantially reducing training time and parameters.

The HW-MLP model predicts the Hamming weight corresponding to an input power trace. Through a key recovery algorithm, the initial key bytes can be recovered using few traces. The algorithm pseudocode is shown in Algorithm 1, which is applicable to any Hamming weight-based profiled power analysis method, not limited to specific neural network or machine learning algorithms.

Algorithm 1: Key Byte Recovery from Hamming Weight Values
Input: HW_{value} $(HW_i)_{1\leq i\leq N}$, plaintexts $(p_i)_{1\leq i\leq N}$
Output: key

```
for i   N do
    for k   K do
        HW_{guess} = S_{box}(k   p_i)
        if HW_{guess} == HW_i then
            set HW_{guess} to the HW_{set}[i]
            HW_{set}[i] = HW_{set}[i]   HW_{set}[i-1]
            if element number of HW_{set}[i] == 1 then
                key = HW_{set}[i]
            end if
        end if
    end for
end for
return key
```

2.5 bit-MLP Model

To further streamline the network model and reduce training parameters, we employ the bit model. With only 2 output classes compared to the HW model, this approach further reduces network layers and neuron counts. The structure is detailed in . By constructing 8 bit models for each key byte, individual key bits can be recovered directly, enabling complete key recovery.

3.1 Data Collection

To ensure experimental validity, all experiments were conducted on identical hardware configurations. Power traces were collected using the ChipWhisperer Lite platform, a complete open-source toolchain for side-channel power analysis and fault injection. The target chip was an XMEGA128D4 microcontroller running AES-128 (128-bit block and key length). A total of 60,000 power traces

were collected using random plaintexts and keys, with 50,000 traces allocated to the training set and 10,000 to the test set.

A single power trace is shown in [Figure 5: see original paper]. Leveraging knowledge of the AES algorithm, the approximate locations of the 16 S-boxes in the first round can be easily identified—the regularly shaped region on the right corresponds to these S-box operations. [Figure 6: see original paper] shows a magnified view of the first two S-box traces, where maxima or minima indicate leakage points. Observation and calculation reveal that each S-box operation spans 72 sampling points, so subsequent experiments use an input size of 1×72 per trace.

3.2 Experimental Evaluation Metrics

Experimental results are analyzed using the following metrics:

a) **Model Accuracy and Loss Function:** Accuracy is the most common metric for neural network training and evaluation, defined as the probability of successful classification on a dataset. Training accuracy corresponds to the maximum accuracy achieved per epoch, while test accuracy represents maximum validation accuracy. Training accuracy is crucial for determining whether the model achieves proper fitting and generalization, indicating whether backpropagation converges to correct weights and biases. The loss function evaluates the discrepancy between predicted and actual values; smaller loss values indicate better model robustness.

b) **Traces Required per Key Byte Recovery:** Model performance is compared by the number of traces needed to recover a single key byte—fewer traces indicate better performance.

c) **Traces Required for Full Key Recovery:** Performance is also compared by the number of traces needed to recover the entire key—fewer traces indicate better performance.

d) **Model Parameters and Computation Time:** Given equivalent prediction accuracy, models with fewer parameters, simpler structures, and shorter training times are superior.

3.3 Experimental Results

ID-MLP Model Experiments

Using the ID-MLP model for key byte recovery, we tested the 0th byte as an example. The model was trained for 200 epochs with a batch size of 100, achieving a final training accuracy of 98.00% and average test accuracy of 98.89%. Compared to the Benadjila-MLP model, this shows significant improvement in training accuracy, faster loss reduction, and requires fewer traces for key recovery, as shown in [Figure 7: see original paper].

Test accuracy and trace counts for recovering other key bytes are presented in . The average training accuracy across all S-boxes reached 97.88%, with an average test accuracy of 95.51%, requiring an average of 1.050 traces per key byte recovery.

We conducted 1,000 full key recovery experiments. The distribution of required trace counts is shown in [Figure 8: see original paper], with an average of 1.563 traces needed for complete key recovery. In 487 experiments, only a single trace was sufficient. These results demonstrate that our improved Benadjila-MLP model achieves excellent prediction performance on our experimental platform.

HW-MLP Model Experiments

The HW-MLP model was used to recover Hamming weights of key bytes, again testing the 0th byte. Trained for 200 epochs with batch size 100, the model achieved 98.00% training accuracy and 97.11% test accuracy, as shown in [Figure 9: see original paper]. Using the Hamming weight key recovery algorithm, recovering the 0th key byte required an average of 4.045 traces. Compared to SVM, the HW-MLP model shows significantly improved accuracy, requires fewer traces, and has shorter training time ().

Results for other key bytes are shown in , with average training accuracy of 97.75%, test accuracy of 96.13%, and an average of 4.185 traces per key byte recovery.

For full key recovery using HW-MLP, 1,000 experiments averaged 6.46 traces, with the distribution shown in [Figure 10: see original paper]. While HW-MLP requires more traces for key recovery than ID-MLP, its simplified architecture yields substantial reductions in training time and parameters.

bit-MLP Model Experiments

The bit-MLP model recovers 8 individual bits per key byte. Testing the 0th bit of the 0th key byte for 200 epochs with batch size 100 achieved 94.56% training accuracy and 95.20% test accuracy ([Figure 11: see original paper]). Results for all bits of the 0th key byte are shown in , with average training accuracy of 94.70%, test accuracy of 93.39%, and 1.469 traces required to recover the 0th key byte.

Results for other S-boxes are presented in , averaging 93.64% training accuracy, 92.56% test accuracy, and 1.540 traces per key byte recovery.

For full key recovery using bit-MLP, 1,000 experiments averaged 3.837 traces, with the distribution shown in [Figure 12: see original paper].

3.4 Experimental Analysis

Model Parameter and Performance Comparison

Both bit-MLP and ID-MLP can recover keys with few traces, but differ significantly in structure and parameters. The bit-MLP model is simpler, trains faster, and achieves the same accuracy ($ $95%) with fewer training samples and less time—reducing parameters by 84% and training time by 28% compared to ID-MLP. The HW-MLP model offers moderate training time and parameters but requires more traces during key recovery.

We scored each model by calculating $1/(N \times P)$, where N is traces needed for full key recovery and P is the number of model parameters, normalizing the highest score to 100. The comparison is shown in .

In practical power analysis scenarios, model selection should consider trace input dimension, signal-to-noise ratio, number of available traces, and computational resources to achieve optimal performance.

Countermeasure Analysis for MLP Models

Current countermeasures against block cipher analysis include: (1) information hiding techniques (random dummy operations, operation shuffling, noise injection); and (2) masking techniques, where each intermediate value is concealed with a random "mask"to prevent information leakage. Jiang et al. [18] proposed a low-area-complexity, low-entropy masking scheme effective against offset-based CPA attacks.

We evaluated bit-MLP against a platform with random dummy operation insertion. Training and testing on the 0th bit of the 0th S-box showed slower accuracy improvement, with final test accuracy of only 64.12%—slightly above random guessing ([Figure 13: see original paper]). This demonstrates that dummy operation insertion provides effective defense.

4 Conclusion

Integrating MLP neural networks into power analysis leverages their feature extraction capabilities to uncover deep relationships between power leakage and processed sensitive information or key data, providing a novel approach for power analysis. Experimental results show that optimizing the MLP model from [11] reduces training parameters by 84% and training time by 28% while improving test accuracy and reducing traces needed for key recovery. In optimal cases, only a single trace is required to recover the full 128-bit AES key.

The proposed models offer advantages of fewer parameters and shorter training times, achieving high prediction accuracy and strong robustness with increasing iterations. These models can also be applied to side-channel power analysis and security evaluation of other block ciphers (such as DES and SM4).

References

[1] Kocher P, Jaffe J, Jun B. Differential power analysis [C]// Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1999: 388-397.
[2] Kocher P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems [C]// Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1996: 104-113.
[3] Gandolfi K, Mourtel C, Olivier F. Electromagnetic analysis: Concrete results [C]// International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2001: 251-261.
[4] De Mulder E, Buysschaert P, Ors S B, et al. Electromagnetic analysis attack on an FPGA implementation of an elliptic curve cryptosystem [C]// EUROCON 2005-The International Conference on "Computer as a Tool" . IEEE, 2005, 2: 1879-1882.
[5] De Mulder E, Örs S B, Preneel B, et al. Differential power and electromagnetic attacks on a FPGA implementation of elliptic curve cryptosystems [J]. Computers & Electrical Engineering, 2007, 33(5-6): 367-382.
[6] Liu Biao, Feng Huamin, Yuan Zheng, et al. Learning to attack from electromagnetic emanation [C]// 2012 6th Asia-Pacific Conference on Environmental Electromagnetics (CEEM). IEEE, 2012: 202-205.
[7] Lerman L, Bontempi G, Markowitch O. A machine learning approach against a masked AES [J]. Journal of Cryptographic Engineering, 2015, 5(2): 123-139.
[8] Maghrebi H, Portigliatti T, Prouff E. Breaking cryptographic implementations using deep learning techniques [C]// International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Cham, 2016: 3-26.
[9] Gilmore R, Hanley N, O' Neill M. Neural network based attack on a masked implementation of AES [C]// 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 2015: 106-111.
[10] Cagli E, Dumas C, Prouff E. Convolutional neural networks with data augmentation against jitter-based countermeasures [C]// International Conference on Cryptographic Hardware and Embedded Systems. Springer, Cham, 2017: 45-68.
[11] Benadjila R, Prouff E, Strullu R, et al. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database [EB/OL]. ANSSI, France & CEA, LETI, MINATEC Campus, France. Online available at https://eprint.iacr.org/2018/053.pdf.
[12] Feng Dengguo, Zhou Yongbin, Liu Jiye, et al. Power Analysis Attack [M]. Beijing: Science Press, 2010: 84-105.
[13] Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model [C]// International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2004: 16-29.
[14] Cagli E, Dumas C, Prouff E. Convolutional neural networks with data augmentation against jitter-based countermeasures [C]// International Conference on Cryptographic Hardware and Embedded Systems. Springer, Cham, 2017: 45-68.

[15] Bishop C M. Neural networks for pattern recognition [J]. Agricultural Engineering International the CIGR Journal of Scientific Research & Development Manuscript PM, 1995, 12(5): 1235-1242.
[16] Ruder S. An overview of gradient descent optimization algorithms [J]. arXiv preprint arXiv:1609.04747, 2016.
[17] Maghrebi H, Portigliatti T, Prouff E. Breaking cryptographic implementations using deep learning techniques [C]// International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Cham, 2016: 3-26.
[18] Jiang Jiuxing, Hou Jiao, Huang Hai, et al. Research on area-efficient low-entropy masking scheme for AES [J]. Journal on Communications, 2019(5): 201-210.

Note: Figure translations are in progress. See original paper for figures.