


Research Article

Power Analysis Attack Based on Hamming Weight Model without Brute Force Cracking

Xiaohong Fan, Jianmin Tong, You Li , Xiaoyi Duan , and Yu Ren

Beijing Electronic Science and Technology Institute, Beijing 100070, China

Correspondence should be addressed to Xiaoyi Duan; duanxiaoyi@besti.edu.cn

Received 8 November 2021; Revised 14 April 2022; Accepted 23 April 2022; Published 16 June 2022

Academic Editor: Shahram Babaie

Copyright © 2022 Xiaohong Fan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Power analysis attack is an attack method to obtain the key in cryptographic chip by analyzing the power information of the cryptographic chip. Machine learning has been widely used in power analysis attacks in recent years. Machine learning can effectively establish the model between the power traces and the SBOX output value or the HM (Hamming) weight of the SBOX output value so that the SBOX output value or the HM weight of the SBOX output value can be obtained through the power traces. HM weight model is widely used because it has less classification of nine and can achieve better machine learning accuracy. However, in the HM weight model, the key cannot be obtained directly by obtaining the median HM weight; instead, the key needs to be deduced by brute force cracking of the median HM weight. Usually, the brute force cracking of a byte key requires 51 enumeration times on average. The HM weight distribution of the SBOX output value is unbalanced, so the power analysis attack based on the HM weight model without brute force cracking is proposed in this paper. Based on the HM weight of the SBOX output value, the method selects the best plaintext for the next power analysis attack, and Euclidean distance is chosen as the optimal plaintext selection judgment algorithm. It makes the HM weight distribution of the SBOX output value more evenly, thus reducing the possible key space and confirming the key more easily. This scheme does not require brute force cracking. It only needs to input 3.332 plaintexts on average and up to 4 plaintexts to determine the unique key, which effectively improves the efficiency of the power analysis attack. In this paper, the authors test the DPA competition V4 data set and Kizhvatov's data set with random defense. Experiments show that this scheme enjoys the high accuracy of the HM weight model in machine learning. Compared with the Whole Byte scheme, the accuracy based on this scheme can be increased by about 360%. Compared with the brute force cracking HM weight scheme, the guessing entropy can be decreased by about 1700%.

1. Introduction

Power analysis attack is one of the most powerful means in side-channel attack, which has the advantages of simple equipment and easy implementation. Power analysis attack is a method of attacking the power consumption information consumed by the cipher chip, which has a certain correlation with the key information during the operation of the cipher chip.

In 1999, Paul Kocher proposed the DPA (Differential Power Analysis) to recover the key of DES [1]. In 2003, Chari S [2] collected many power traces to establish statistical information and used TA (template attack) to obtain the key. In 2003, Eric Brier [3] proposed a method of using correlation power analysis.

1.1. Related Work. In recent years, with the continuous development of machine learning, more and more scholars have applied machine learning to power analysis attacks. In 2011, Hospodar et al. applied machine learning techniques to the power analysis attacks of side-channel attack for the first time. For the data set with obvious HM weight leakage, they successfully attacked some software implementation of the AES (Advanced Encryption Standard) by using LS-SVM (Least Squares Support Vector Machine) [4]. In 2012, He et al. used SVM to attack the DES (Data Encryption Standard) algorithm running on 8-bit smart cards [5]. In [6–8], relevant scholars used machine learning to attack unprotected cryptographic algorithms. In 2014, machine learning was first introduced to attack the AES implementation with mask countermeasure. In 2017, Eleonora

Cagli [9] and others used machine learning and data enhancement technology to attack chips with delay protection. In 2018, Benadjila et al. [10] used CNN (Convolutional Neural Networks) to attack the algorithm implementation on a single-chip computer with mask and disturbance defense. Timon B proposed a side-channel attack based on deep learning in the Non-Profiled scenario [11]. This method maps the Whole Byte classification of the intermediate value or the HM weight model of the intermediate value to the assumed power consumption value as the label and uses the accuracy and loss value as the key discriminator. AES algorithm key can be obtained successfully after the training and the prediction by using MLP (Multilayer Perceptron) algorithm and CNN algorithm. In 2019 Kim et al. proposed a new Convolutional Neural Network to analyze side-channel [12], and Mathieu et al. used deep learning to evaluate secure RSA implementation [13]. In 2020, Xiaoyi Duan and others used data enhancement to solve the imbalance problem of the HM weight of SBOX output value in machine learning [14].

Machine learning has been widely used in power analysis attacks in recent years. Machine learning can effectively establish the model between the power traces and the SBOX output value or the HM weight of the SBOX output value, which is respectively called the Whole Byte model and HM model in this paper so that the SBOX output value or the HM weight of the SBOX output value can be obtained through the power traces. HM model is widely used because it has less classification of nine and can achieve better machine learning accuracy. However, in the HM model, the key cannot be obtained directly by obtaining the median HM weight; instead, the key needs to be deduced by brute force cracking of the median HM weight. This cracking method is called the Force-HM scheme in this paper. Usually, the brute force cracking of a byte key requires 51 enumeration times on average, which is not very efficient.

1.2. Our Contribution. The SBOX output HM weight value distribution is unbalanced, so the power analysis attack based on the HM model without brute force cracking is proposed in this paper, which is called the Non-Force-HM scheme. Based on the HM model, the scheme selects the best plaintext in the power analysis attack of the next attack, and Euclidean distance is chosen as the optimal plaintext selection judgment algorithm. It makes the SBOX output HM weight value distribution more evenly, thus reducing the possibility of determining the key space and key uniqueness. This scheme does not require brute force cracking. It only needs to input 3.332 plaintexts on average and up to 4 plaintexts to determine the unique key, which effectively improves the efficiency of the power analysis attack. Experiments show that this scheme enjoys the high accuracy of the HM model in machine learning. Compared with the Whole Byte scheme, the accuracy based on this scheme can be increased greatly. Compared with the brute force cracking HM weight scheme, the guessing entropy can be decreased substantially. The comparison of the Force-HM scheme and the Non-Force-HM scheme is shown in Figure 1.

1.3. Structure of This Paper. The structure of this paper is as follows: In Section II, the power analysis attack based on machine learning and its related computation are introduced. In Section III, the Non-Force-HM scheme proposed in this paper is introduced. The principle of Euclidean distance is introduced, which can be applied to choose the optimal plaintext. The mapping table of plaintext, HM weight value of SBOX output, and key is given. In Section IV, experimental verification and discussion are carried out. In Section V, conclusion and future work are presented.

2. Power Analysis Attack

2.1. Power Analysis Attack Based on Machine Learning. Power analysis attack is a kind of attack method which uses the correlation between the power traces of the cipher chip and the key when executing the cryptographic algorithm. It is one of the powerful attack means in the side-channel attack, which has the advantages of simple implementation equipment and easy implementation. In recent years, with the development of artificial intelligence, the combination of machine learning and power analysis attack has become more and more, which greatly improves the success rate and attack efficiency of attacks.

The power analysis attack based on machine learning can be expressed as establishing a possible template for every possible class $c \in \{1, \dots, C\}$, in which the number of class C depends on the assumed leakage model. Assuming that, for each class $c \in \{1, \dots, C\}$, the attacker obtains the power consumption trace vector $\{l_c^i\}_{i=1}^{N_c}$, in which N_c is the number of power consumption trace vectors of class C . Since the template attack depends on the multivariate Gaussian noise model, the power trace vector is considered to be drawn from the multivariate distribution. Formulas (1) and (2) give more precise expressions.

$$N(l_c | \mu_c, \Sigma_c) = \frac{1}{(2\pi)^N 1/2 |\Sigma_c|^{1/2}} \exp \left\{ -\frac{1}{2} (l_c - \mu_c)^T \sum_x^{-1} (l_c - \mu_c) \right\}, \quad (1)$$

$$\tilde{\mu}_c = \frac{1}{N_c} \sum_{n_c=1}^{N_c} l_{n_c}, \tilde{\Sigma}_c = \frac{1}{N_c} \sum_{n_c=1}^{N_c} (l_{n_c} - \tilde{\mu}_c)(l_{n_c} - \tilde{\mu}_c)^T. \quad (2)$$

These templates are built based on the estimation of expectations $\tilde{\mu}_c$ and covariance matrix $\tilde{\Sigma}_c$. The key recovery in the attack phase is performed by maximum likelihood estimation or equivalent log-likelihood rule as shown in the following formula:

$$\log L_{k^*} \equiv \log \prod_{i=1}^{N_2} P(l_i | c) = \sum_{i=1}^{N_2} \log N(l_i | \mu_c, \Sigma_c), \quad (3)$$

where class C is calculated according to the leakage model based on given key guess k^* and input.

High additional computational complexity is required to recover the entire key. There are various methods to reduce the computational complexity of key recovery. The first

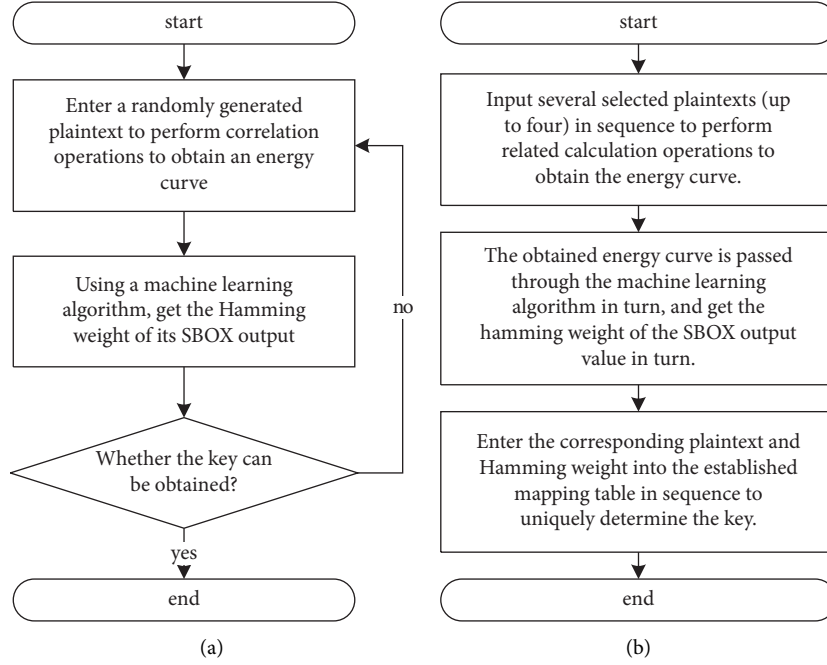


FIGURE 1: The comparison of (a) the Force-HM scheme and (b) the Non-Force-HM scheme.

method is to extend the leakage model to the HM weight leakage model. HM weight leakage model assumes the whole middle value instead of just a few bits of the middle value. The second method is to extend the attack to multiple power traces. Logarithmic maximum likelihood estimation for every possible key k^* is shown in the following formula:

$$\log L_{k^*} \equiv \log \prod_{i=1}^N P(l_i|c) = \sum_{i=1}^N \log P(l_i|c). \quad (4)$$

Select the key to maximize the possibility as shown in the following formula:

$$\arg \max_{k^*} \log L_{k^*}. \quad (5)$$

2.2. HM Weight Model. HM weight [15] refers to the number of nonzero elements in a string. For a common binary string, it is the number of “1” in the string. In power analysis attacks, the HM model is generally used to represent the power consumption model of the running chip [16].

2.3. HM Weight Characteristics of AES SBOX Output. Due to the high nonlinearity of SBOX in AES, for power analysis attack, this is the best attack point. Therefore, in the power analysis attack, the output value of SBOX is often used as an attack point. When using the HM model, the HM weight of the SBOX output value is generally attacked.

For each byte of plaintext input, the output values of SBOX are in the interval of [0,255] and are not equal. HM weight refers to the number of “1” in a binary value, so there are 9 cases of HM weight of the SBOX output [17]. The “01” balance of SBOX output leads to the imbalance of its HM weight. The HM weight values of AES SBOX output conform

to normal distribution. After each byte of plaintext passes through SBOX, the HM weight distribution of output and its distribution probability are shown in Table 1.

2.4. Power Analysis Attack Based on HM Weight Model.

When using machine learning and the HM model for the power analysis attack, the general attack process is shown in Figure 1(a). First, a 9-class model is trained according to the HM weight values. When the power trace is input, the HM weight of the SBOX output value can be obtained through machine learning, and then the enumeration attack can be started.

When the HM weight of the SBOX output value is known, the average number of attempts required to obtain a one-byte key can be estimated using formula (6), which is expressed as θ .

$$\theta = \sum_{i=0}^8 p_i \times G_i. \quad (6)$$

In the formula, G_i refers to the HM weight distribution value output by SBOX, and p_i refers to its corresponding distribution probability.

According to Table 1 and (6), we can calculate the number of times needed to enumerate a byte, as shown in the following formula:

$$\begin{aligned} \theta &= \frac{1}{256} \times 1 + \frac{8}{256} \times 8 + \frac{28}{256} \times 28 + \frac{56}{256} \\ &\times 56 + \frac{70}{256} \times 70 + \frac{56}{256} \times 56 + \frac{28}{256} \times 28 + \frac{8}{256} \\ &\times 8 + \frac{1}{256} \times 1 = 50.27344. \end{aligned} \quad (7)$$

TABLE 1: Corresponding table of HM weight values and number.

HM weight	0	1	2	3	4	5	6	7	8
Number	1	8	28	56	70	56	28	8	1
P_i	1/256	8/256	28/256	56/256	70/256	56/256	28/256	8/256	1/256

Therefore, when obtaining the HM weight of the SBOX output value, the brute force cracking of a one-byte key requires 51 enumeration times on average.

3. Power Analysis Attack Based on HM Weight Model without Brute Force Cracking

The HM weight distribution of the SBOX output value is unbalanced, so the Non-Force-HM scheme is proposed in this paper. Based on the HM model, the scheme selects the best plaintext in the power analysis attack of the next attack, and Euclidean distance is chosen as the optimal plaintext selection judgment algorithm. It makes the HM weight distribution of the SBOX output value more evenly, thus reducing the possible key space and confirming the key more easily.

3.1. Euclidean Distance. Euclidean distance measures the absolute distance between two points in multidimensional space. It can also be understood as the true distance between two points in multidimensional space or the natural length of the vector. The Euclidean distance in two-dimensional and three-dimensional space is the actual distance between two points [18]. The schematic diagram of the Euclidean distance between point A and point B is shown in Figure 2.

The Euclidean distance between point $A(x_1, y_1)$ and point $B(x_2, y_2)$ in the two-dimensional plane is shown in the following formula:

$$\text{dist}(A, B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \quad (8)$$

The Euclidean distance between point $A(x_1, y_1, z_1)$ and point $B(x_2, y_2, z_2)$ in the three-dimensional plane is shown in the following formula:

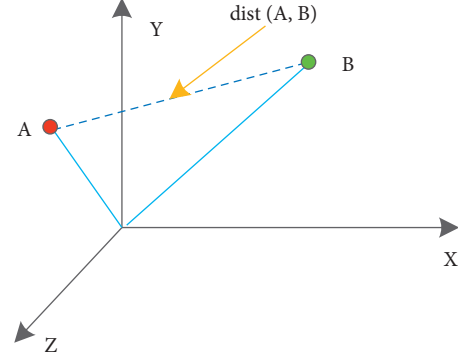


FIGURE 2: Schematic diagram of Euclidean distance.

$$\text{dist}(A, B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}. \quad (9)$$

The Euclidean distance between points $A(x_{11}, x_{12}, \dots, x_{1n})$ and $B(x_{21}, x_{22}, \dots, x_{2n})$ in the n -dimensional space is shown in the following formula:

$$\text{dist}(A, B) = \sqrt{\sum_{k=1}^n (x_{1k} - x_{2k})^2}. \quad (10)$$

3.2. How to Select Best Plaintext. Suppose that the key set is $[0, 255]$. As shown in Table 2, when a plaintext is an input for calculation, the HM weight of the SBOX output value can be obtained. Each HM weight can correspond to multiple keys, which form the key subset K . Assuming that the number of keys in subset K is t , traversing plaintext $m_i (0 \leq i \leq 255)$, a mapping as shown in formula (11) can be defined for each plaintext m_i .

$$\psi_{m_i}: (Y, K) \mapsto Y \{y_{ij} \in Y | y_{ij} = HM[Sbox(m_i \oplus k_j)], k_j \in K\}, \quad 0 \leq i \leq 255, 0 \leq j \leq t. \quad (11)$$

That is, each plaintext m_i is executed with each key k_j in the key subset K , and $256 \times t$ $y_{ij} = HM[Sbox(m_i \oplus k_j)]$, $(0 \leq i \leq 255, 0 \leq j \leq t)$ $(0 \leq y_{ij} \leq 8)$ can be obtained as shown in Table 2.

Define count set $Z_i = \{z_{i0}, z_{i1}, z_{i2}, z_{i3}, z_{i4}, z_{i5}, z_{i6}, z_{i7}, z_{i8} | 0 \leq i \leq 255\}$, Z_i is the number of times that HM weight values 0, 1, ..., 8 appear in line i . Define the ideal count set of key subset $T = \{t/9, t/9, t/9, t/9, t/9, t/9, t/9, t/9, t/9\}$.

The Euclidean distance between each Z_i and the ideal count set T can be calculated. Z_i corresponding to the minimal Euclidean distance is the ideal counting set, and the corresponding m_i is the best plaintext. According to this idea, the key can be determined by the least plaintext.

The average number of the power traces required to attack a byte key is shown in the following formula:

$$p = \frac{\sum_{i=0}^{255} x_i}{256}, \quad (12)$$

where x_i represents the number of the power traces required to determine any possible key $k_i \in \{0, 1, \dots, 254, 255\}$.

3.3. Plaintext Selection for AES Cipher Algorithm. According to the method in 3.2, calculate the chosen plaintext of the AES cipher algorithm. Figure 3 shows the

TABLE 2: Subset table.

	k_1	...	k_j	...	k_t
m_0	γ_{01}	...	γ_{0j}	...	γ_{0t}
...
m_i	γ_{i1}	...	γ_{ij}	...	γ_{it}
...
m_{255}	γ_{255_1}	...	γ_{255_j}	...	γ_{255_t}

key pool reduction diagram for the plaintext selection process.

The pseudocode based on the chosen plaintext algorithm in this paper is shown in Figure 4.

Table 3 is the mapping table of partial plaintext, the HM weight of the SBOX output value, and key calculated by the method in this paper. The complete mapping table is shown in the Supplementary Materials (available here).

Substitute the experimental results in the attachment mapping table into formula (12), and get the average number of power traces needed to attack one byte key, as shown in the following formula:

$$p \approx 3.32. \quad (13)$$

Using the Non-Force-HM scheme proposed in this paper, only at least one plaintext input, up to four plaintext inputs, with an average of 3.332 plaintext inputs, can complete a successful attack.

4. Experimental Results and Analysis

In order to better verify the advantages of the solution proposed in this paper, two machine learning models are established, that is, HM model and Whole Byte model. This paper analyzes these two models. Compared with the Whole Byte scheme, the HM model scheme has greater advantages in accuracy and guessing entropy.

This experiment selected two data sets: one is a data set without delay, and the other is a data set with random delay protection. In the experiment, the nondelay data set is from the DPA contest V4 [19], named DS1. A total of 10000 power traces were obtained, including 9000 for training and 1000 for testing. This experiment used MLP, SVM, RF (Random Forest), and CNN to attack the set. The data set with random delay was running in 8-bit AVR in [20], named DS2. There are 10000 traces, 9000 for training and 1000 for testing.

4.1. Evaluation Indicator. In this paper, three different indicators are used to evaluate the performance of the model: accuracy, NGE (new guessing entropy), and computational complexity. Accuracy reflects the accuracy of classification, that is, the probability of successful attack. This paper will not elaborate on it, but the NGE and computational complexity defined in this paper will be explained, respectively.

4.1.1. NGE . Guessing entropy is a common index to evaluate attack performance in SCA (side-channel attacks) [2]. The definition is as follows: g is the decreasing order of

the probabilities of all possible keys in each experiment, and i is the index of the correct key. Through s experiments, a matrix $[g_1, g_2, \dots, g_s]$ and the corresponding vector $[i_1, i_2, \dots, i_s]$ are obtained. The guessing entropy is the average position of the correct key, as shown in the following formula:

$$GE = \frac{1}{s} \sum_{x=1}^s i_x. \quad (14)$$

In other words, the guessing entropy describes the average number of guesses required to recover the actual key.

However, in practice, the success rate of attack is difficult to reach 100%. When the HM model is used to attack, the HM weight of the SBOX output value will be obtained. In order to obtain the key, an enumeration attack is still required. The traditional guessing entropy does not take into account the fact that the HM weight cannot directly attack the key, and it cannot express the actual number of attacks. Therefore, this paper defines a new guessing entropy, which is calculated by the following formula:

$$NGE = n \times GE. \quad (15)$$

Among them, n represents the number of attacks that still need to be enumerated after obtaining the HM weight of the SBOX output value. The chosen plaintext attack scheme proposed in this paper needs an average of 3.32 attacks to achieve 100% attack success rate. When we input selected plaintext, through the machine learning algorithm, we will obtain the average guess times required for the attack, that is, the traditional guessing entropy GE . Therefore, the NGE based on Non-Force-HM scheme is $NGE = 3.32 \times GE$. The NGE based on Force-HM scheme is $NGE = 51 \times GE$, when selecting the Whole Byte model, the attack is successful when the Whole Byte of SBOX output is obtained, and there is no need to enumerate. Therefore, the NGE based on the Force-HM scheme is $NGE = GE$.

4.1.2. Computational Complexity. Computational complexity is a mathematical method to quantitatively analyze the consumption of various resources required in the calculation. In order to highlight the advantages of the Non-Force-HM scheme in computational complexity, it is compared with the Force-HM scheme and the Whole Byte scheme. We define the computational complexity of the AES 16-byte key attacked by three schemes as follows:

- (1) Suppose that the computational complexity of an energy analysis attack is Pa . Define that the guessing entropy of the Force-HM energy attack is HM_GE and the guessing entropy of the Whole Byte energy attack is $Whole_byte_GE$.
- (2) According to Section 3.3, the Non-Force-HM energy attack scheme proposed in this paper needs an average of 3.32 attacks to achieve 100% attack success rate. Therefore, the computational complexity based on this scheme is

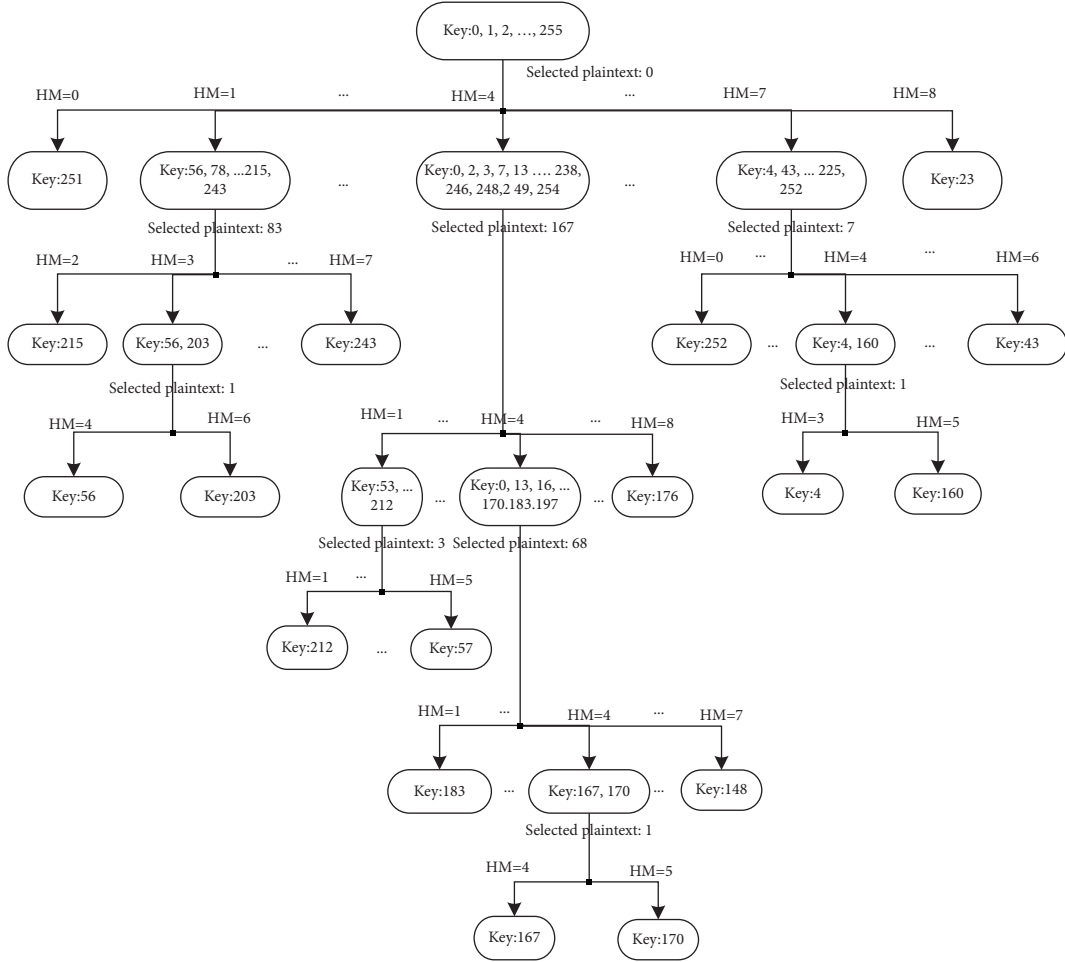


FIGURE 3: Key pool reduction diagram for the plaintext selection process.

$$Pa \times HM_GE \times 3.32 \times 16. \quad (16)$$

- (3) As can be seen from Section 2.4, the Force-HM energy analysis attack requires 51 attack curves. Therefore, the computational complexity based on this scheme is

$$Pa \times HM_GE \times 51 \times 16. \quad (17)$$

- (4) The computational complexity of the Whole Byte model is

$$Pa \times Whole_byte_GE \times 16. \quad (18)$$

4.2. Voltage Characteristics of Data. Using the HM model, the relationship between the voltage value of the trace and the HM value can be plotted in Figure 5 [14]. At the same time, if the Whole Byte model is adopted, the relationship between the voltage value of the trace and the classification value of the Whole Byte can be plotted in Figure 6. These two figures show that there is an obvious linear relationship between the HM weight and the voltage value; that is, the voltage is proportional to the HM weight, but there is no obvious linear relationship between the Whole Byte

classification value and the voltage. Because HM weight is proportional to voltage, it is easier to classify by machine learning.

4.3. Attacks on DS1. With the help of artificial intelligence algorithms such as SVM, RF, CNN, and MLP, this paper uses the Whole Byte scheme, HM scheme, and Non-Force-HM scheme to attack the nondelay data sets, and the results were compared from the guessing entropy and accuracy. The experimental results show that the two indexes of the Non-Force-HM scheme are fine. The accuracy comparison between the HM model and the Whole Byte model with different machine learning algorithms is shown in Figure 7. The HM model has only 9 categories, and the Whole Byte model has 256 categories, so the voltage value of the HM weight is more differentiated and has better accuracy than the Whole Byte model in learning algorithms.

For DS1, the guessing entropy of the Non-Force-HM scheme, the Force-HM scheme, and the Whole Byte scheme is shown in Figure 8. Compared with the Force-HM scheme, the guessing entropy of the Non-Force-HM scheme is much smaller.

The computational complexity of the three attack schemes is shown in Table 4.

Algorithm 1: Select plaintext attack

Input: Sbox Hamming weight output characteristic table sboxoutlist [256]
Output : Key space nine-pronged search tree root structure pointer
 *KeySpaceTreeRoot

```

1 CREATE SearchQueue = [] ;
2 CREATE TreeRootNode =
3 {
4   *(NextNode[9]) = NULL ;
5   NextPlaintext = INF ;
6   KeySpaceSize = 256 ;
7   KeySpace[KeySpaceSize] = [0,255] ;
8 } ;
9 CREATE *TreeNode = &TreeRootNode ;
10 SearchQueue.in( *TreeNode ) ;
11 while( SearchQueue is not EMPTY ) :
12 {
13   *TreeNode = SearchQueue.out('head') ;
14   if(*TreeNode.KeySpaceSize != 1) :
15     {
16       for TEMPplaintext in [0,255] :
17         Distance[TEMPplaintext]= Euclid (SboxOutList
18         (TEMPplain^*text TreeNode.KeySpace),*TreeNode.KeySpaceSize/9) ;
19       end
20       *TreeNode.NextPlaintext = where( Distance == min(Distance) ) ;
21       for i in [0,8] :
22         CREATE newNode =
23         {
24           *(NextNode[9]) = NULL ;
25           NextPlaintext = INF ;
26           KeySpace[KeySpaceSize]=*TreeNode.KeySpace[(SboxOut
27           List(TreeNode.NextPlaintext^*TreeNode.KeySpace) == i) ] ;
28           KeySpaceSize = sizeof( KeySpace ) ;
29         }
30         *TreeNode.NextNode[i] = &newNode ;
31         if( *TreeNode.NextNode[i].KeySpaceSize != 0 ) :
32           SearchQueue.in(*TreeNode.NextNode[i]) ;
33         end
34       }
35     }
36 }
37 }
38 save ALL ;

```

FIGURE 4: Pseudocode of the algorithm.

TABLE 3: Mapping table of partial plaintext, the HM weight of SBOX output value, and key.

First round			Second round			Third round			Fourth round			Result
Key subset	Selected plaintext	HM weight	Key subset	Selected plaintext	HM weight	Key subset	Selected plaintext	HM weight	Key subset	Selected plaintext	HM weight	The key
0		0	251									251
1			56		2	215						215
2			78		3	56	1	4	56			56
3			115		3	203	1	6	203			203
4		1	146	83	4	115	1	4	158			158
5			158		5	158		5	115			115
6			203		5	78						78
7			215		6	146						146
8			243		7	243						243
9			6		0	22						22
10			11		1	163	2	1	213			213
11			22		1	213	2	5	163			163
12			27		2	11	1	3	11			11
13			37		2	230	1	5	230			230
14			54			95		1	206			206
15	0		61		3	135	5	3	135			135
16			64		3	180	5	4	95			95
17			68			206		5	180			180
18			95			27		2	234			234
19		2	101	237		64		3	64			64
20			110		4	182	12	5	222			222
21			129			222		7	182			182
22			132			234		8	27			27
23			135			6		2	68			68
24			163			54		3	165			165
25			165			68		4	54	1	3	101
26			180		5	101	121	5	101		4	54
27			182			110		5	204			204
28			195			165		6	6			6
29			204			195		7	195			195
30			206			204		8	110			110

TABLE 4: Comparison of computational complexity of the three schemes for DS1.

Scheme	Non-force-HM	Force-HM	Whole byte
Computational complexity	$48.512 \times Pa$	$843.776 \times Pa$	$46.752 \times Pa$

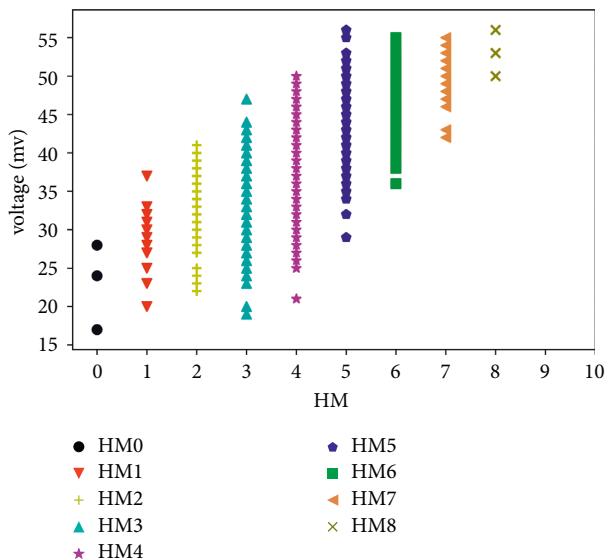


FIGURE 5: HM-two-dimensional distribution of power consumption.

As can be seen from Table 4, when the 16-byte AES algorithm is attacked simultaneously, the advantages of the Non-Force-HM scheme are more prominent. Compared with the Force-HM scheme, this scheme has a very low order of magnitude of computational complexity. The computational complexity of this scheme is almost the same as that of the Whole Byte model and better than the Force-HM model.

This paper compares the attack success rate of the Whole Byte model and HM model with different machine learning algorithms under different number of training traces, as shown in Figures 9 and 10. As can be seen from Figure 9, during the training stage, as the number of power traces increases, the attack success rate of the Whole Byte model increases slowly until it reaches the maximum. The attack success rate of the four machine learning algorithms is about 0.15 when the number of training traces is about 1000, and the attack success rate is the highest 0.557 when the number of training traces is 9,000. As can be seen from Figure 10, in the HM model,

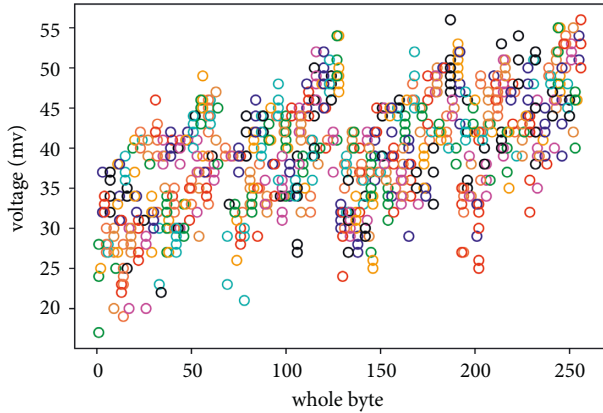


FIGURE 6: The Whole Byte classification-two-dimensional distribution of power consumption.

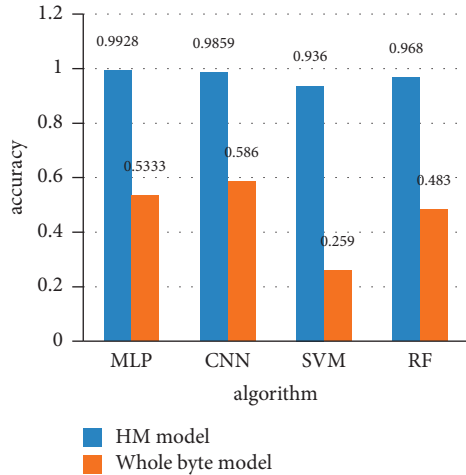


FIGURE 7: Accuracy of the HM model and Whole Byte model with different algorithms for DS1.

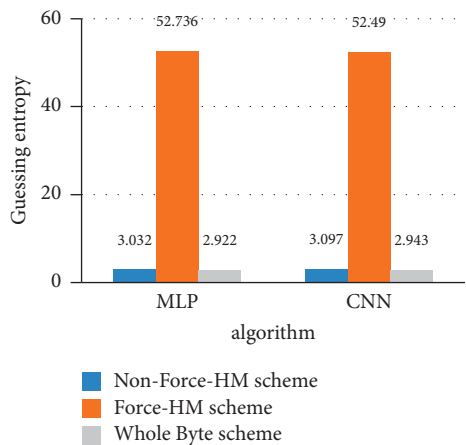


FIGURE 8: Guessing entropy of the three schemes with different algorithms for DS1.

TABLE 5: Comparison of computational complexity of the three schemes for DS2.

Scheme	Non-Force-HM	Force-HM	Whole byte
Computational complexity	$101.9 \times Pa$	$1566.72 \times Pa$	$232.84 \times Pa$

with the increase of the number of power traces, the success rate of the attack will increase sharply until it reaches the maximum. When the CNN algorithm is selected, the success rate of the attack reaches 0.941 when the number of training traces is about 1000. When the MLP algorithm is selected, the attack success rate is as high as 0.996 when the number of training traces is 9000.

4.4. Attacks against DS2. With the help of artificial intelligence algorithms such as SVM, RF, CNN, and MLP, this paper uses Whole Byte scheme, HM scheme, and Non-Force-HM scheme to attack the data sets with delay, and the results were compared from the guessing entropy and accuracy. Since the data set with delay adds delay protection, the attack on such data sets becomes more difficult. As Figure 11 shows, the accuracy of the four algorithms is reduced, but the reduction of the HM model is relatively low compared with the Whole Byte model.

For DS2, the guessing entropy of the Non-Force-HM scheme, the Force-HM scheme, and the Whole Byte scheme is shown in Figure 12. Compared with the Force-HM scheme, the guessing entropy of the Non-Force-HM scheme is much smaller.

The computational complexity of the three attack schemes is shown in Table 5.

As can be seen from Table 4, when the 16-byte AES algorithm is attacked simultaneously, the advantages of the Non-Force-HM scheme are more prominent. Compared to the Force-HM scheme, this scheme has a very low order of magnitude of computational complexity. Its computational complexity is significantly lower than that of the Whole Byte scheme too.

For DS2, this paper compares the attack success rate of the Whole Byte model and HM model with different machine learning algorithms under different number of training traces, as shown in Figures 13 and 14, respectively. The success rate of 9 attacks is recorded successively with power traces ranging from 1000 to 9000. As can be seen from Figures 13 and 14, in the training stage, as the number of power traces increases, the attack success rate of the two models gradually increases and steadily reaches the maximum. If the Whole Byte model is used as the classification label, when the number of training traces is about 4000, the attack success rate reaches the maximum and stabilizes at about 0.004. By contrast, if the HM model is used as the classification label when the number of training traces is about 1000, the attack success rate reaches the maximum and stabilizes at about 0.25. Therefore, compared with the Whole Byte model, the HM model has a higher attack success rate and requires fewer training power traces numbers.

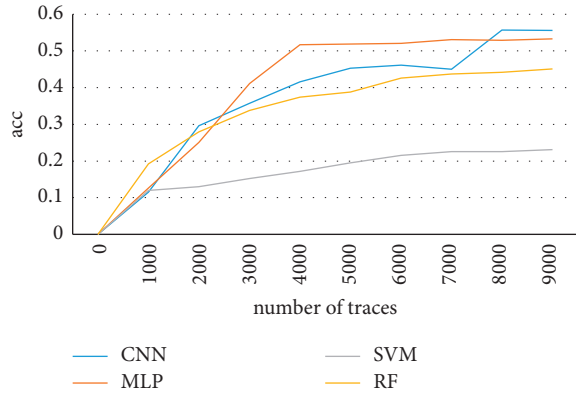


FIGURE 9: Accuracy of different algorithms in Whole Byte model for DS1.

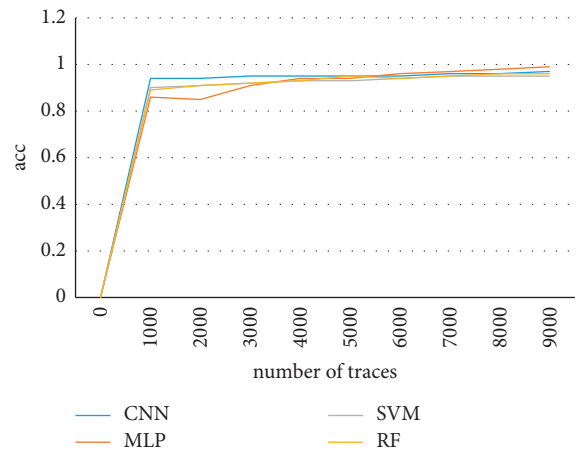


FIGURE 10: Accuracy of different algorithms in HM model for DS1.

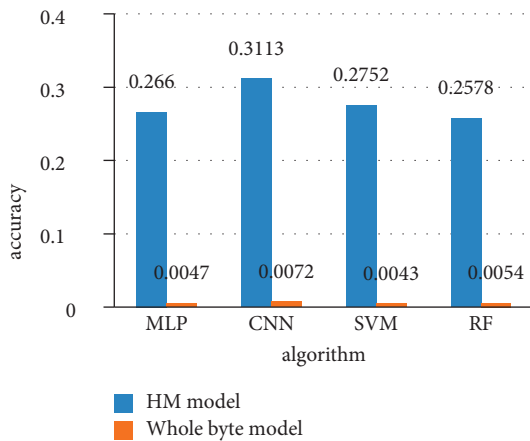


FIGURE 11: Accuracy of the HM model and Whole Byte model with different algorithms for DS2.

5. Summary

This paper proposes a power analysis attack method of the None-Force-HM scheme that uses the HM model and does not require brute force cracking. Based on the HM weight of the SBOX output value, the method selects the best plaintext for the

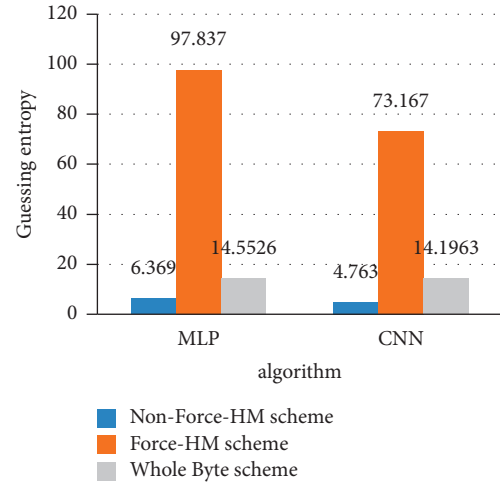


FIGURE 12: Guessing entropy of the three schemes with different algorithms for DS2.

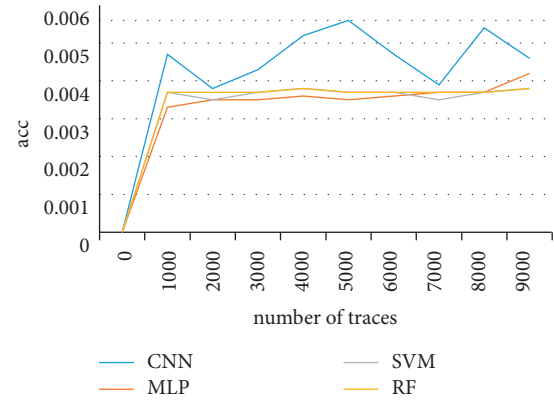


FIGURE 13: Accuracy of different algorithms with Whole Byte model for DS2.

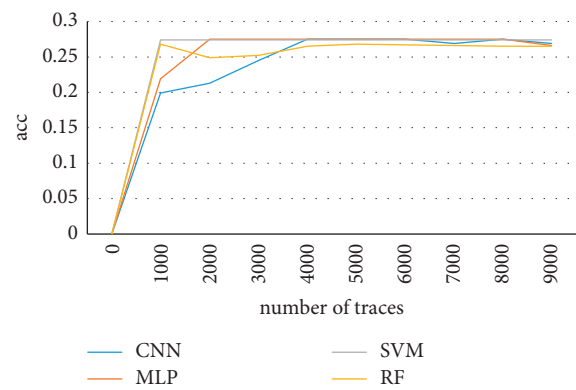


FIGURE 14: Accuracy of different algorithms with HM model for DS2.

next power analysis attack, and Euclidean distance is chosen as the optimal plaintext selection judgment algorithm. It makes the HM weight distribution of the SBOX output value more evenly, thus reducing the possible key space and confirming the key more easily. This scheme does not require brute force cracking. It only needs to input 3.332 plaintexts on average and up to 4

plaintexts to determine the unique key, which effectively improves the efficiency of the power analysis attack.

The Non-Force-HM scheme is based on the HM model, and it has the following characteristics:

- (1) It still uses the HM model, so there are 9 classes when using machine learning for classification. Compared with the 256 classes of the Whole Byte model, the model is easier to train successfully and has better accuracy.
- (2) Since the average number of traces required for a successful attack is 3.332 and the maximum is 4, it also has an advantage in guessing entropy compared with the Whole Byte model.

Non-Force-HM scheme is based on the attack of selecting plaintext. In view of this, the following research points can be considered in the future [17]:

- (1) How to select and attack the plaintext for the algorithm with mask defense.
- (2) How to carry out plaintext selection and attack for the algorithm with disturbance defense.

Data Availability

The data used to support the findings of this study are from the DPA Contest v4 database, which is available at http://www.dpacontest.org/v4/42_traces.php.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the High-Tech Discipline Construction Funds of China (no. 20210032Z0401; no. 20210033Z0402).

Supplementary Materials

The following table is the mapping table of plaintext, the HM weight of SBOX output value and key calculated by the method in this paper. (*Supplementary Materials*)

References

- [1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis, advances in cryptology-CRYPTO'99," *Proc. 19th Annual International Cryptology Conf.* vol. 1666, pp. 388–397, 1999.
- [2] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Germany, February, 2002.
- [3] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 3156, Springer, Berlin, Germany, August, 2004.
- [4] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 293–302, 2011.
- [5] H. He, J. Jaffe, and L. Zou, *Side Channel Cryptanalysis Using Machine Learning*, pp. 1–392CS229 Project, 2012, <http://cs229.stanford.edu/proj2012/HeJaffeZou-SideChannelCryptanalysisUsingMachineLearning.pdf>.
- [6] T. Bartkewitz and K. Lemke-Rust, "Efficient template attacks based on probabilistic multi-class support vector machines," vol. 7771, pp. 263–276, in *Proceedings of the International Conference on Smart Card Research and Advanced Applications*, vol. 7771, Springer, Berlin, Germany, November, 2012.
- [7] A. Heuser and M. Zohner, "Intelligent machine homicide," vol. 7275, pp. 249–264, in *Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design*, vol. 7275, Springer, Berlin, Germany, May, 2012.
- [8] L. Lerman, R. Poussier, G. Bontempi, O. Markowich, and F. X. Standaert, "Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis)," vol. 9064, pp. 20–33, in *Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design*, vol. 9064, Springer, Cham, July, 2015.
- [9] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," vol. 10529, pp. 45–68, in *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*, vol. 10529, Springer, Taipei, Taiwan, August, 2017.
- [10] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *Journal of Cryptographic Engineering*, vol. 10, no. 2, pp. 163–188, 2020.
- [11] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2, pp. 107–131, 2019.
- [12] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2, pp. 148–179, 2019.
- [13] M. Carbone, V. Conin, M. A. Cornelié et al., "Deep learning to evaluate secure RSA implementations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 3, pp. 132–161, 2019.
- [14] X. Duan, D. Chen, X. Fan, X. Li, D. Ding, and Y. Li, "Research and Implementation on Power Analysis Attacks for Unbalanced Data," *Security and Communication Networks*, vol. 2020, Article ID 5695943, 10 pages, 2020.
- [15] M. L. Akkar, R. Bevan, P. Dischamp, and D. Moyart, "Power analysis, what is now possible," vol. 1976, pp. 489–502, in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, vol. 1976, Springer, Berlin, Germany, October, 2000.
- [16] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Vol. 31, Springer Science & Business Media, Berlin, Germany, 2008.
- [17] J. Daemen, "aes proposal: Rijndael," *Aes Algorithm Submission*, 1999.
- [18] B. Lei, G. Xu, M. Feng et al., *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*, John Wiley & Sons, Hoboken, NJ, USA, 2017.
- [19] <http://dpacontest.org/home/>.
- [20] J.-S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," vol. 5747, pp. 156–170, in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 5747, Springer, Berlin, Germany, September, 2009.