Research on Data Preprocessing in Side Channel Attack Based on Multi-Layer Perception

Tiankai Yu1, Min Wang1, Yi Wang1, Zhen Wu1 and Xi Wei2

1 Department of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China.
2 Southern Power Grid Science Research Institute Co., Ltd, Guangzhou 510080, China.
Email: yuhaitan@126.com

Abstract. The data encryption process in the encryption chip will be leaked by means of timing operation time, probe collection operation power and electromagnetic signal interception, which makes side channel attack possible. The actual power consumption data usually contains a lot of noise, which reduces the success rate of side channel attack. This paper studies the preprocessing of FPGA power data, proposes two indexes -- MSNR (Maximum Signal Noise Ratio) and GE (Guessing Entropy), and the variance and mean curves of Hamming weight assist modelling to summarize three reliable preprocessing methods based on dynamic powers.

1. Introduction

With the popularization of Internet of Things technology in intelligent city, the protection of hardware encryption based on FPGA has become one of the important protective measures to protect sensitive data. The algorithm used in the hardware implementation of the encryption process has a reliable security theory proof, but for the attacker, the acquisition of a large amount of power leakage during hardware operation recovers sensitive data (keys, etc.). At present, the simple power analysis (SPA, simple power analysis) [1] cannot fully recover the key in the FPGA encryption algorithm. Differential power analysis (DPA), although statistical rules can be used for side channel analysis, a sufficient power curves are required. In 2002, Suresh Chari proposed a template attack (TA) to speed up the side channel attack [2]. It used noise as model in accordance with the multivariate gaussian distribution. It is not necessary for Multi-Layer Perceptions (MLP) objected to noise gaussian distribution [3]; it also resists jitter problems in the power curve. The method of side channel attack, the preprocessing of power consumption curve is indispensable. At present, the work in this area is divided into three categories:

1) Converting power consumption data into other forms of data, for example, using regularization and wave transformation for preprocessing;
2) Dimension reduction of the power consumption curve, for example, averaging multiple curves in the same clock cycle to reduce the amount of calculation;
3) Alignment of the power consumption curve, for example, uses the correlation coefficient method to determine the time and then align [4].

DPA CONTEST V2 [5] data will be processed according to the preprocessing method, key information leakage, variance and mean comparison, and MLP-based channel attack. The effects of different pre-processing methods on side channel attacks are quantified by means of MSNR and GE.
2. Multi-Layer Perception

In recent years, MLP research has made great progress, and the connection weights between hidden layers are corrected by the Back Propagation (BP) algorithm. The steps include:

Step 1: Initialize the connection weight between neurons $\text{Weight}(k)_{m,n}$ is a small random value, where $(m, n)$ indicates that the $m$th node of the $k$th layer is connected to the $n$th node of the $(k + 1)$th layer.

Step 2: Training $(x, y)$, where $x$ is the input data and $y$ is the real data. The loss function describes the difference between $y$ and the prediction result $\hat{y}$.

Step 3: The BP algorithm is used to calculate the gradient of the loss function relative to the weight in the neural network:

$$\nabla \text{Loss} = \frac{\partial \text{Loss}}{\partial \text{Weight}(k)_{m,n}} \tag{1}$$

Step 4: Update the weight and reduce the loss function. Change the learning rate $\theta$ to quickly find the best;

$$\Delta \text{Weight}(k)_{m,n} = \theta \nabla \text{Loss} \tag{2}$$

Step 5: Repeat the three steps until the training result reaches the preset threshold or the training is completed [6].

3. FPGA Hardware Implementation

FPGAs belong to the wide family of programmable logic components; An FPGA is defined as a matrix of configurable logic blocks, linked to each other's by an interconnection network which is entirely reprogrammable [7]. The memory cells control the logic blocks as well as the connections. Several configurable technologies exist. FPGA hardware encryption uses parallel encryption compared to traditional encryption soft implementations.

![Figure 1. Generic Architecture of an FPGA](image)

4. Preprocessing of Power Consumption Curve

4.1. Regularization

Prior to the actual attack, the collected power consumption curves are aligned, and the curves are normalized so that each curve has a mean value of 0:

$$\text{Align} \ i_n(x_i, k_s) = i_n(x_i, k_s) - \text{mean}(i_n(x_i, k_s)) \tag{3}$$
4.2. Butterworth Filter
The power consumption is different from the frequency band occupied by noise. The experiment performs filtering preprocessing from the frequency domain. In the actual attack, different parameters are selected according to the data and the experimental environment (filter sampling rate, etc.).

\[
|H(\tau T)|^2 = A(T^2) = \frac{1}{1 + \left(\frac{T}{T_c}\right)^{2N}} 
\]

Where \( T \) is the cutoff frequency of the Butterworth low-pass filter, and \( N \) is the order [8].

\[
N = \log \left( \frac{10^{R_f/10} - 1}{10^{R_p/10} - 1} \right) 
\]

\[
N = \log \left( \frac{T_f/T_p} \right) 
\]

4.3. Wave Transforms
In actual attacks, the acquired signals are usually analyzed in the time domain. Wave transforms can reveal details related to encryption in the frequency domain.

4.3.1. Discrete Fourier Transform
The collected data is not continuous, there is an extremely tiny time difference between the sampling points, so the frequency domain analysis uses the discrete Fourier transform:

\[
\text{DFT}(\ ) = \sum_{n=-N}^{N} (\ ) e^{-2\pi i n} 
\]

Although the Fourier transform has a bit of information leakage in the frequency domain, it cannot analyze power consumption in the time domain [9].

4.3.2. Short-time Discrete Fourier Transform
In order to overcome the shortcomings of the Fourier transform not recognizing the time domain information, the fast Fourier transform came into being. The fast Fourier transform uses a sliding window function \( s(t) \) of fixed time \( \tau \), where * represents the conjugate factor.

\[
\text{DSFT}(\ , \ ) = \sum_{n=-N}^{N} (\ ) * (\ - \ ) e^{-2\pi i n} 
\]

\( s(t) \) is limited by the W. Heisenberg uncertainty criterion, and the area of the time-frequency window is not less than 2.[10] This also shows from the side that the time and frequency resolution of the short-time Fourier transform window function cannot be optimized at the same time.

5. Side Channel Attack Based on MLP

5.1. Experimental Data
In the DPA Contest V2 data set, each power consumption curve is 3253 sample points, the bandwidth is 5 GHz, and the sampling rate is 5G sample/s. The FPGA device runs the AES-128 encryption algorithm in parallel at 24 MHz. The Public data set is a training data set, including 32 random keys, and each round of keys is 20000 random plaintexts. The Template data set is attack data including 32 fixed keys.
5.2. Signal-to-Noise Ratio
Signal-to-Noise Ratio (SNR), \(SNR = \frac{Var(E(signal))}{E(Var(signal))}\). The lower the signal-to-noise ratio is, the less features in the energy consumption are included, which denotes the leaking of the power curve [11]. Among them, MSNR represents the sampling point with the largest signal to noise ratio.

5.3. Mean and Variance Curve of Power Noise
In side channel attack, the noise is usually modeled. But if you build 256 templates, it will bring huge time costs. If you use Hamming weight to model the noise, the time complexity is greatly reduced. The initial target value of 0-255 is replaced with 9 Hamming weights of 0, 1, 2, 3, 4, 5, 6, 7, 8 when modeling. Only need to find the key sequences of the noise distribution (based on the mean and the variance curve) in the data, and select them as the training data set and the attack data set respectively. Figure 1. denotes the mean-variables curve of Template origin data, it leaks nothing to assaults; Figure 2. denotes mean-variable curve of origin data with Regularization, and each key can be adapted to attack; Figure 3. represents

![Figure 2. Mean-Variable Curve of Template Data (Origin)](image)

Mean-variable curve of origin data after butterworth filter, it is clear that key1 and key10 is the same; Figure 4. denotes mean-variable curve of origin data after Discrete Fourier transform, each key can be used to attack except key2, 4, 6, 10;

5.4. Attack Process
The power consumption curve was preprocessed using regularization, butterworth filter, and flourier transform algorithms to improve the signal-to-noise ratio. In order to reduce the computational complexity, 2500-3000 sampling points (initial data is 0-3253 sampling points) are selected for training. In the process of neural network training, after calculating the intermediate value of the AES-128 algorithm, it compares with 256 (0-255) target values to define the loss function. The validity of the side channel attack is checked by the guessing entropy of the deep perceptron:

1) Stable reduction of entropy represents an effective side channel attack. The partial fluctuations and anomalies of the guessing entropy in the attack are acceptable. The training of the MLP has certain anti-shaking to the DPAV2 data.
2) Using mean and variance to assist model based on noise, noise distributions and the parameters in the pre-processing method are not same [12]. The optimized parameters given in the
paper are not rule of thumb, but attack results in the previous experiments, and there are certain fluctuations.

6. Result
The power consumption curve is filtered by algorithms such as normalization, Butterworth filter and discrete Fourier transform, and the pre-processed SNR and noise mean-variance curves are compared. Attack object: Template data set (attack data containing 32 fixed keys).

Attack result:
- Table 1, Table 2, Table 3 and Table 4 denotes enhancement of the consumption curves, meanwhile, wave transform always change the origin shape of the curve, and we ignore its enhancement;
- It is not that all the preprocessing of each Key can improve the signal-to-noise ratio, but the experimental verification shows that most Keys can improve the signal-to-noise ratio after preprocessing;
- Table 5 shows the final results, each preprocessing could reduce the guessing entropy, where regularization denotes the best efficiency.

| Table 1. Signal-to-Noise Enhancement (From Key0 to Key15)-Butterworth Filter |
|------------------|------------------|------------------|------------------|------------------|
| Key0             | 37.57%           | Key4             | 33.22%           | Key8             | 78.83%           |
| Key1             | -41.90%          | Key5             | 50.88%           | Key9             | 50.91%           |
| Key2             | 67.19%           | Key6             | 64.04%           | Key10            | 26.73%           |
| Key3             | 57.12%           | Key7             | 64.83%           | Key11            | 50.49%           |
|                  |                  |                  |                  | Key12            | 39.72%           |
|                  |                  |                  |                  | Key13            | 15.11%           |
|                  |                  |                  |                  | Key14            | 22.57%           |
|                  |                  |                  |                  | Key15            | 62.26%           |

| Table 2. Signal-to-Noise Enhancement (From Key16 to Key31)-Butterworth Filter |
|------------------|------------------|------------------|------------------|------------------|
| Key16            | -64.81%          | Key20            | 37.13%           | Key24            | 45.99%           |
| Key17            | -36.12%          | Key21            | 61.84%           | Key25            | 54.12%           |
| Key18            | 44.47%           | Key22            | 39.73%           | Key26            | 76.34%           |
| Key19            | -21.18%          | Key23            | 37.25%           | Key27            | 35.54%           |

| Table 3. Signal-to-Noise Enhancement (From Key0 to Key15)-Regularization |
|------------------|------------------|------------------|------------------|------------------|
| Key0             | 38.76%           | Key4             | 36.03%           | Key8             | 77.92%           |
| Key1             | -45.40%          | Key5             | 50.28%           | Key9             | 28.92%           |
| Key2             | 40.68%           | Key6             | 54.59%           | Key10            | 40.06%           |
| Key3             | 33.50%           | Key7             | 36.53%           | Key11            | 51.57%           |

| Table 4. Signal-to-Noise Enhancement (From Key16 to Key31)-Regularization |
|------------------|------------------|------------------|------------------|------------------|
| Key16            | -58.87%          | Key20            | 22.74%           | Key24            | 29.14%           |
| Key17            | -38.95%          | Key21            | 64.60%           | Key25            | 57.17%           |
| Key18            | 35.29%           | Key22            | 29.51%           | Key26            | 70.57%           |
| Key19            | -38.54%          | Key23            | 34.27%           | Key27            | 33.84%           |

7. Conclusion
Different preprocessing methods have different effects on the data set, because of different noise distributions. The three preprocessing methods are used to reduce the guessing entropy of the attack, yet, the attack parameters cannot be optimized fully. In the following time, the parameter optimization in the side channel analysis of the FPGA hardware device will continue.
Table 5. Attack Guessing Entropy (From 1st round to 10th round)

| Preprocessing  | 1rd | 2rd | 3rd | 4rd | 5rd | 6rd | 7rd | 8rd | 9rd | 10rd |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|
| None           | 119.8 | 99.5 | 84.6 | 72.8 | 62.8 | 55.3 | 55.3 | 52.5 | 52.1 | 44.9 |
| Regularization | 79 | 53.5 | 31.5 | 29.7 | 28.3 | 13.9 | 12.5 | 14.9 | 14.5 | 8 |
| Butterworth    | 93.2 | 85.4 | 59.2 | 38.6 | 43.8 | 38.4 | 27.5 | 37.6 | 28.3 | 13.7 |
| Wave Transform | 116.2 | 91 | 77 | 75.3 | 61.3 | 53.9 | 45.7 | 53.3 | 51 | 43.8 |

8. Acknowledgments
This work is supported in part by the National Key R&D Program of China (2018YFB0904900, 2018YFB0904901), The Key R&D Program of Sichuan Province (No.2019YFG0096), The “13th Five-Years” National Cryptogram Development Fund (No.MMJ20180244).

9. References
[1] François Koeune, François-Xavier Standaert. A Tutorial on Physical Security and Side-Channel Attacks[J]. 2005.
[2] Lv J, Han Y. Enhanced DES Implementation Secure Against High-Order Differential Power Analysis in Smartcards[C]// Australasian Conference on Information Security & Privacy. 2005.
[3] CHARI, S., RAO, J. R., and ROHATGI, P., Template attacks[M]. Cryptographic Hardware and Embedded Systems-CHES 2002. ed: Springer, 2003.
[4] Potlapally N R, Raghunathan A, Ravi S, et al. Aiding Side-Channel Attacks on Cryptographic Software With Satisfiability-Based Analysis[J]. IEEE Transactions on Very Large Scale Integration Systems, 2007, 15(4):465-470.
[5] Clavier C, Danger J L, Duc G, et al. Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest[J]. Journal of Cryptographic Engineering, 2014, 4(4):259-274.
[6] Hannan M A, Arebey M, Begum R A, et al. An automated solid waste bin level detection system using Gabor wavelet filters and multi-layer perception[J]. Resources Conservation & Recycling, 2013, Volume 72(2):Pages 33–42.
[7] Monmasson E, Cirstea M N. FPGA Design Methodology for Industrial Control Systems—A Review[J]. IEEE Transactions on Industrial Electronics, 2007, 54(4):1824-1842.
[8] Yang L J, Zhang B H, Xu-Zhen Y E. Fast Fourier transform and its applications[J]. Opto-electronic Engineering, 2004, 31:303-350.
[9] Harris F J. On the use of windows for harmonic analysis with the discrete Fourier transform[J]. IEEE Proceedings, 1978, 66(1):51-83.
[10] Griffin D, Lim J S. Signal estimation from modified short-time Fourier transform[J]. IEEE Transactions on Acoustics Speech & Signal Processing, 1984, 32(2):226-243.
[11] Yang L J, Zhang B H, Xu-Zhen Y E. Fast Fourier transform and its applications[J]. Opto-electronic Engineering, 2004, 31:303-350.
[12] Maghrebi H, PortigliattiT, ProuffE. Breaking Cryptographic Implementations Using Deep Learning Techniques[J]. 2016.