Side-Channel Evaluation on Protected Implementations of Several NIST LWC Finalists

Dawu Gu, Pei Cao, Yuhang Ji, Xiangjun Lu, Shipei Qu, Tengfei Wang, Chi Zhang, Hongyi Zhang, Xiaolin Zhang (sorted alphabetically by last name)
Cryptology and Computer Security Laboratory (LoCCS)

School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China

August 12, 2022

On the Side Channel Leakage Assessment of First-Order Masked GIFT-COFB

Xiangjun Lu[1], Shipei Qu[1], Tengfei Wang[1], Pei Cao[1]

School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China

1 Introduction

1.1 Background

GIFT-COFB is an Authenticated Encryption with Associated Data (AEAD) scheme, based on the GIFT lightweight block cipher and the COFB lightweight AEAD operating mode[BCI+20]. It has been selected as one of the finalists in the NIST lightweight cryptography standardization process. The side channel analysis of the native GIFT-COFB is carried out in [HBB20], while its mask-protected implementation has not yet been explored.

In power side-channel analysis, the attacker tries to recover secret information from the hardware running the cryptographic algorithm by recording the power consumption traces. In order to protect cryptographic algorithms from such attacks, it is often implemented with boolean masks to hide the real secret information.

In this report, we will perform a side-channel leakage assessment against GIFT-COFB with first-order boolean masking in both software and hardware implementations. The collected power traces are going through leakage detection and attack attempts to investigate the performance of the power side-channel resilience of GIFT-COFB.

1.2 Our Work and Results Overview

Our work in this report and the results of the side-channel leakage assessment on firstorder masked GIFT-COFB can be summarized as follows.

- We collected two trace sets from the given software and hardware implementations of GIFT-COFB on an MCU and a side-channel attack evaluation board.
- We performed Welch's t-test [BCD+13] and χ^2-test [MRSS18] to evaluate the power leakage of GIFT-COFB. We tried to recover the private keys of GIFT-COFB by correlational power attack (CPA).
- χ^2-test applied on the power traces from the given hardware implementations shows a slight potential power leakage from the input nonce. However, such leakage is missing in Welch's t-tests or χ^2-test on software implementations.
- CPA attack cannot recover the private key bytes under the given implementations.

2 Assessment Strategy

Our assessment strategy on the given GIFT-COFB implementations can be summarized as the following three phases:

1. Analysis of the target

The initial phase of GIFT-COFB is shown in Figure 1. We consider the known-plaintext attack scenario, which is a common assumption in side-channel attacks. Then we take the first block encryption component as the target of the attack, which is a GIFT-128 cipher.

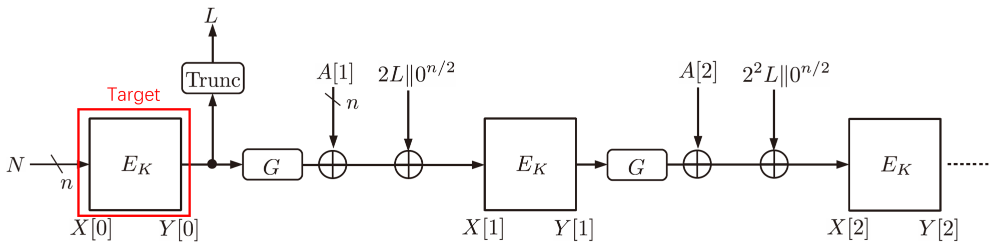


Figure 1: Analysis target

Next, we choose the output of *SubCells* operation in the second round as the intermediate value. The reason for choosing this intermediate value is that it has both a direct correspondence with the round key and a high degree of non-linearity.

Let's take the notation in the GIFT-COFB specification[BCI+20]. Specifically, the 128-bit secret key is loaded into the key state KS partitioned into 8 16-bit words:

$$KS = \begin{bmatrix} W_0 & \| & W_1 \\ W_2 & \| & W_3 \\ W_4 & \| & W_5 \\ W_6 & \| & W_7 \end{bmatrix} \leftarrow \begin{bmatrix} b_{127} & \cdots & b_{112} & \| & b_{111} & \cdots & b_{98} & b_{97} & b_{96} \\ b_{95} & \cdots & b_{80} & \| & b_{79} & \cdots & b_{66} & b_{65} & b_{64} \\ b_{63} & \cdots & b_{48} & \| & b_{47} & \cdots & b_{34} & b_{33} & b_{32} \\ b_{31} & \cdots & b_{16} & \| & b_{15} & \cdots & b_{2} & b_{1} & b_{0} \end{bmatrix} \tag{1}$$

And the cipher state S is expressed as 4 32-bit segments:

$$S = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} \leftarrow \begin{bmatrix} b_{124} & \cdots & b_8 & b_4 & b_0 \\ b_{125} & \cdots & b_9 & b_5 & b_1 \\ b_{126} & \cdots & b_{10} & b_6 & b_2 \\ b_{127} & \cdots & b_{11} & b_7 & b_3 \end{bmatrix} \tag{2}$$

Suppose the cipher state before the *AddRoundKey* operation in the first round is $\{S_0^{'}, S_1^{'}, S_2^{'}, S_3^{'}\}$, which can be derived from the nonce N. Then *AddRoundKey* will update the cipher state with the round key and constant in the first round:

$$\begin{aligned} S_0^{''} &\leftarrow S_0^{'} \\ S_1^{''} &\leftarrow S_1^{'} \oplus W_6 \| W_7, \\ S_2^{''} &\leftarrow S_2^{'} \oplus W_2 \| W_3, \\ S_3^{''} &\leftarrow S_3^{'} \oplus \text{0x80000001} \end{aligned} \tag{3}$$

In order to perform side channel attacks such as CPA, we must be able to compute the corresponding intermediate value from parts of the key we guessed. In typical side-channel attacks (e.g. AES), one byte is often guessed and another byte is obtained as an intermediate value. However, this strategy does not work for the GIFT-128, because $S_1^{''}$ and $S_2^{''}$ affects multiple bytes in the output of the second round's *SubCells*:

$$\begin{aligned}
S_1 &\leftarrow S_1^{''} \oplus \left(S_0^{''} \& S_2^{''}\right) \\
S_0 &\leftarrow S_0^{''} \oplus \left(S_1 \& S_3^{''}\right) \\
S_2 &\leftarrow S_2^{''} \oplus (S_0 \mid S_1) \\
S_3 &\leftarrow S_3^{''} \oplus S_2 \\
S_1 &\leftarrow S_1 \oplus S_3 \\
S_3 &\leftarrow \sim S_3 \\
S_2 &\leftarrow S_2 \oplus (S_0 \& S_1) \\
\{S_0, S_1, S_2, S_3\} &\leftarrow \{S_3, S_1, S_2, S_0\}
\end{aligned} \tag{4}$$

Noting that the bit position of each byte does not change, so we can solve this problem by guessing 1 byte in $S_1^{''}$ and 1 byte in $S_2^{''}$, and calculate the corresponding 1 byte in the result. For example, if we choose the last byte of the output S_3 as the intermediate value, the calculation can be expressed as:

$$S_3[0] \leftarrow S_0^{''}[0] \oplus \left(\left(S_1^{''}[0] \oplus \left(S_0^{''}[0] \& S_2^{''}[0]\right)\right) \& S_3^{''}[0]\right)$$

where the index 0 indicates the position of the byte, $S_i^{''}$ can be obtained from Eq. 1 (2 bytes from $W_{2,6}/W_{3,7}$ is guessed). Based on the same principle, we can also use bit-level intermediate values, which can help to verify the leakage of side-channels more quickly.

2. Side-channel leakage detection

Next, we applied TVLA (Test Vector Leakage Assessment) to determine whether the collected power traces had noticeable plaintext or intermediate value leaks. Specifically, the main techniques used here are Welch's t-test and χ^2 test. They can roughly locate where in the traces the power leakage occurred.

3. Key recovery attack evaluation

Note that if there is power leakage detected in Phase 2, we can apply CPA here to reveal half of the key ($W_{2,3,6,7}$). The other half of the master key needs to attack the third round of *SubCells* with the same strategy based on the success of the first half of the key.

3 Experimental Setup

In this section, we will describe the details of power traces acquisition process.

3.1 Overall Procedure

The procedure of out power trace collection experiments is presented in Figure 2.

As shown in the figure, we first need to download the firmware which including the implementation of GIFT-COFB and our custom communication protocol into the device under evaluation. Then we connect the device to the host computer through a USB serial port so that we can invoke the cipher and record its input and output. Meanwhile, we use a high-precision electromagnetic probe to capture the electromagnetic power emitted from the device chip. The captured power is then transmitted to the oscilloscope to generate and display the waveform of electronic signals. With the help of the oscilloscope, we can acquire enough raw power traces of protected GIFT-COFB in the host computer for later assessment.

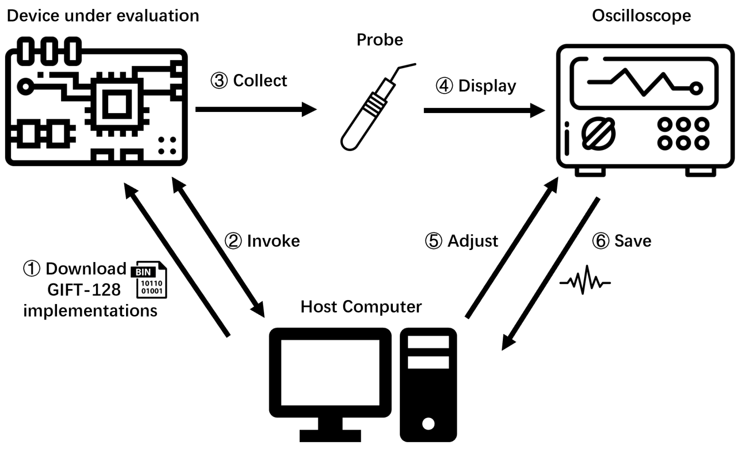


Figure 2: Overall procedure of power trace collection

3.2 Experimental Setting

3.2.1 Experimental environments

The details of devices and analyzing suites used for GIFT-COFB are presented in Table 1.

Table 1: Details of experimental environments

Type	Items	Details
Hardware platform	Target MCU	STM32F303RCT6
	Target evaluation board	Saseabo-giii(Kintex-7)
Measuring tools	High Precision EM probe	Langer RF-U 5-2
	Oscilloscope	Pico 3203D, LeCroy 610Zi
Sampling parameters	Sampling rate for MCU	125 MHz
	Sampling rate for FPGA	500 MHz
Random source	standard C library	`rand()`, `srand()` in stdlib.h

We assign GPIO_12 of STM32F303RCT6 (CN9 of Saseabo-gii) as the pin sending the trigger signals. The given software and hardware implementations of GIFT-COFB will be tested on STM32F303RCT6 and Saseabo-giii, respectively.

3.2.2 Input and output of GIFT-COFB

For the experiments of power trace collection on software implementation, the input of GIFT-COFB encryption consists of three parts: a 16-byte nonce, 16-byte associated data and 16-byte plaintext. The output consists of 16-byte ciphertext and a 16-byte authenticated tag. For the hardware implementation, it requires the input to be already masked data and thus twice as long as the original ones. The 16-byte encryption key is fixed throughout the collection. The specific information about the fixed input is shown in 2. All the fixed value are directly copied from the official test vectors provided in the implementer's code repository.

According to the analysis in 2, changing either the input nonce N or plaintext will change the intermediate values. Here we choose to alter the nonce in each encryption. Then the intermediate values will change under the same key, thereby generating different but related power consumption patterns. This allows us to perform CPA and other tests.

Table 2: Input details of GIFT-COFB

Implementation	Fixed Input	Value
Software	Master key	000102030405060708090A0B0C0D0E0F
	Plaintext	000102030405060708090A0B0C0D0E0F
	Associated data	000102030405060708090A0B0C0D0E0F
Hardware(masked)	Master key	B54F97F73F0716B75845D3D652C015A7 FEA43B246C15EA6E619601E3FACC42A7
	Plaintext	C5F8D832CBF8D832
	Associated data	A25D267C615D267C

3.2.3 Trigger setting

Apart from the equipment mentioned in 2, another probe attached to the oscilloscope can receive trigger signals to help us locate the timing when GIFT-COFB is executed. Thus, we need to modify the original GIFT-COFB implementations so that they can control the corresponding pins of the device to send trigger signals to the oscilloscope.

For the software implementation, the codes to control the pin and send the trigger signals are inserted into prior and after the call to the first call to `giftb128_encrypt_block`, as shown in Figure 3.

```
int giftcofb_crypt(
...
    gift128_keyschedule(key, m_rkey.rkey, key_m);
    // Call trigger
    HAL_GPIO_TogglePin(Trigger_GPIO_PORT, Trigger_Pin);
    giftb128_encrypt_block(y, m_rkey.rkey, nonce);
    // Call trigger
    HAL_GPIO_TogglePin(Trigger_GPIO_PORT, Trigger_Pin);
```

Figure 3: Code snippet to set triggers in the software implementation

For the hardware implementation, we use a passive way to set the trigger signal, i.e. the algorithm will block until we supply a high level to a certain pin. The trigger is set as an external signal that enables the hardware to start executing the algorithm by pulling up for 1 clock cycle. This signal is also connected to the oscilloscope as a trigger control for the trace acquisition.

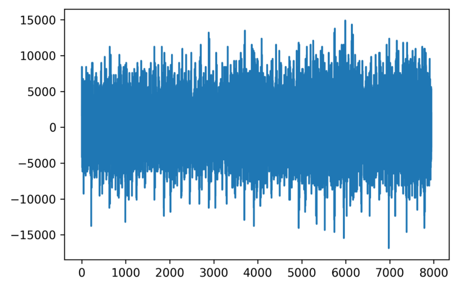
4 Description of Collected Raw Traces

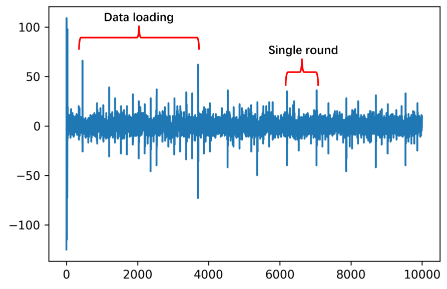
We collected two sets of power traces, (S) and (H). (S) is acquired from the given software GIFT-COFB implementations under settings described in Section 3, and (H) is from the hardware implementation. Their basic information is presented in Table 3.

Table 3

Item	Software Implementation	Hardware Implementation
Trace set ID	S	H
Rounds contained	40	7
No. of traces	20,000	1,000,000
No. of points per trace	8,000	10,000
Precision	$-2^{15} \sim 2^{15}$	$-2^{7} \sim 2^{7}$
Sampling time	5h	12h

The sample plots of trace set (S) and (H) are presented in Figure 4. As seen from Figure 4a, we can easily distinguish the rounds in GIFT-128 encryption from (H).


(a) (S)



(b) (H)

Figure 4: Sample graph of trace set (S) and (H)

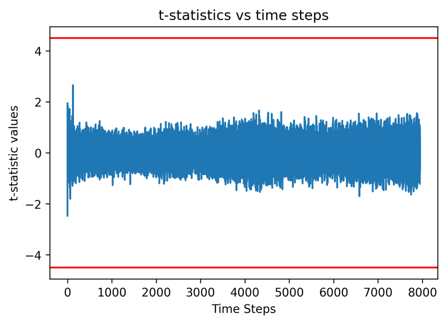
Then we can perform different tests mentioned in Section 2 on them to evaluate the power leakage of the given implementations.

5 Main Result

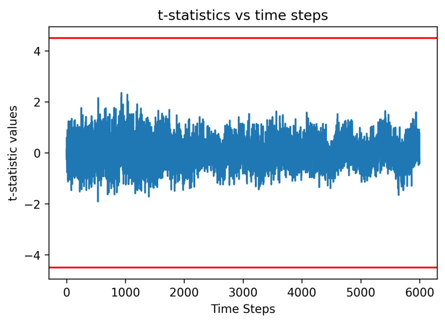
5.1 Welch's *t*-test

Welch's *t*-test is a statistical hypothesis test used to compare the means of two groups, especially when the two groups have unequal sample sizes and variances. In terms of side-channel analysis, we can divide the power traces into two groups according to the difference in intermediate values. More precisely, when the master key is fixed, we can divide the power traces of GIFT-COFB by the following two cases.

- Case(I): The last bit of the first byte of the input nonce is 0 or 1.
- Case(II): The last bit of the first byte of the intermediate value is 0 or 1.



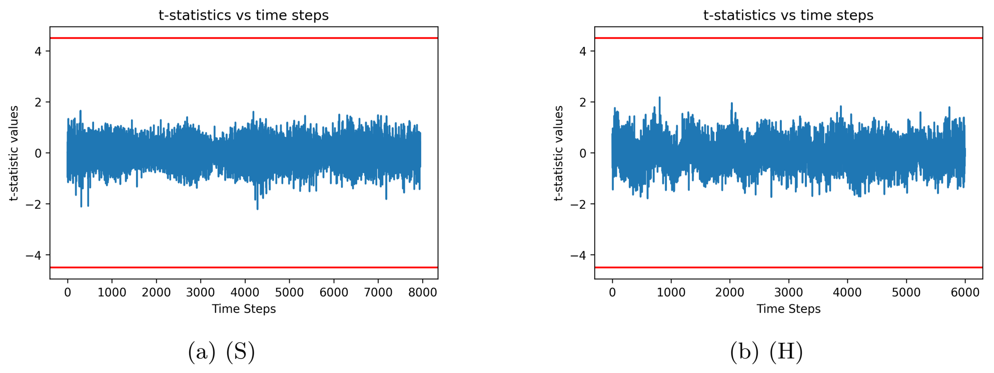
(a) (S)



(b) (H)

Figure 5: Welch's t-test results of (S) and (H) (divided by Case (I))

The test results are shown in Figure 5 and Figure 6. We can see from the figure that the results failed to reach the threshold of the Welch's *t*-test for either the software or the hardware implementation, suggesting that no significant leakage information can be detected using this test approach.



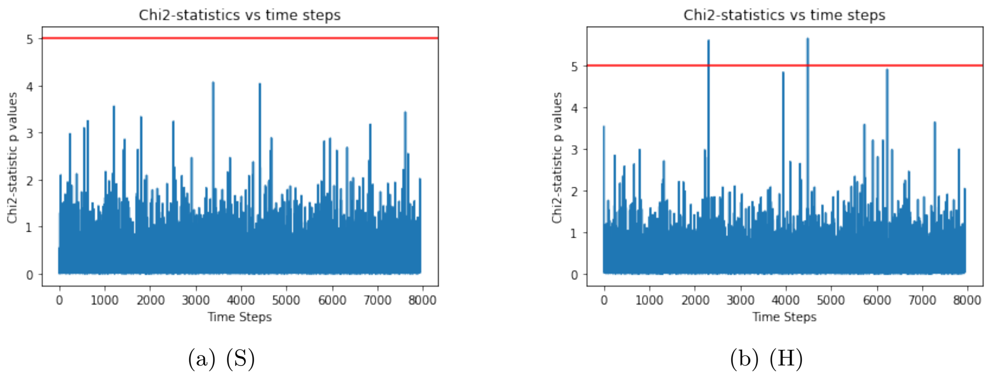
(a) (S) (b) (H)

Figure 6: Welch's t-test results of (S) and (H) (divided by Case (II))

5.2 χ^2-test

χ^2-test is another statistical hypothesis test to determine whether there is a significant difference between the expected and observed frequencies, which is a natural complement to Welch's t-test for black box leakage detection, especially in the case of higher-order masked implementations. It can also test the null hypothesis of independence of a pair of random variables. Therefore, like t-test, we divide the power traces by the following two cases and observe their statistical differences.

- Case(I): The last bit of the first byte of the input nonce is 0 or 1.
- Case(II): The last bit of the first byte of the intermediate value is 0 or 1.



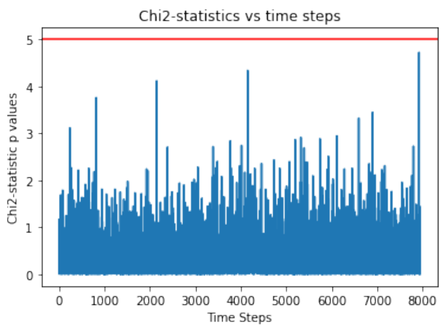
(a) (S) (b) (H)

Figure 7: χ^2-test results of (S) and (H) (divided by Case (I))

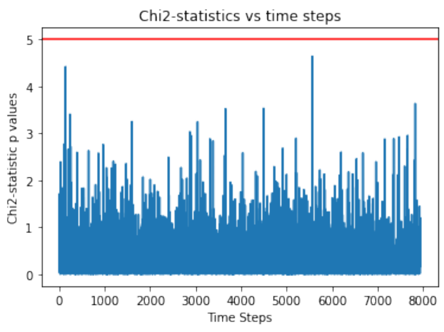
The test results are shown in Figure 7 and Figure 8. We can see from the figure that the results failed to reach the confidence level of theχ^2-test in 7a, but there is slight power leakage detected from (H) in Figure 7b when the traces are divided by nonce. However, when the traces are divided according to the intermediate values, χ^2-test cannot find statistically significant differences of two trace groups.

5.3 Correlational power attack (CPA)

CPA is an efficient side-channel analysis method to reveal the secret from power leakage of a cryptographic device. According to the analysis in Section 2, we will guess 1 bit (or 1



(a) (S)



(b) (H)

Figure 8: χ^2-test results of (S) and (H) (divided by Case (II))

byte, using more computational resources) from two different subkeys $W_{2,6}/W_{3,7}$ at a time and compute the corresponding intermediate value. Taking the bit model for example, we can get a sequence of bits from the calculation of intermediate value. The correct guess will exhibit the greatest level of correlation between the bit sequence and real power trace, which indicates the correct subkey bit.

Table 4: CPA guess result of key bits in (S)

Target bit index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Target bit value (W2||W3)	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
Target bit value (W6||W7)	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	1
Real key rank(/4)	3	2	3	3	4	3	1	3	1	1	2	4	2	4	1	1
Best guess	01	10	01	10	00	00	00	11	00	00	01	10	00	00	00	11
Real key	00	00	00	00	01	11	00	00	00	00	00	00	01	11	00	11
Target bit index	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Target bit value (W2||W3)	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1
Target bit value (W6||W7)	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1
Real key rank(/4)	2	3	1	3	4	4	3	1	4	1	1	2	1	2	3	1
Best guess	10	11	00	01	10	10	10	00	10	00	00	01	01	01	00	11
Real key	00	00	00	00	01	11	11	00	00	00	00	00	01	11	11	11

Table 5: CPA guess result of key bits in (H)

Target bit index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Target bit value (W2||W3)	1	1	0	0	1	1	0	0	1	0	1	0	0	1	0	1
Target bit value (W6||W7)	0	1	0	1	1	1	1	1	1	1	1	0	1	1	0	0
Real key rank	4	2	3	2	4	3	3	1	2	2	4	4	1	3	1	1
Best guess	11	01	11	11	10	00	10	01	00	10	01	10	01	10	00	01
Real key	10	11	00	01	11	11	01	01	11	11	11	00	01	11	00	01
Target bit index	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Target bit value (W2||W3)	0	0	0	1	1	0	0	1	1	0	0	0	1	1	1	1
Target bit value (W6||W7)	0	1	0	1	1	0	0	1	1	1	0	1	1	1	0	1
Real key rank	1	2	1	2	1	2	3	1	1	2	2	1	3	3	3	2
Best guess	00	11	00	10	11	01	11	11	11	10	01	01	00	10	01	01
Real key	00	01	00	11	11	00	00	11	11	01	00	01	11	11	10	11

For trace set (S), we perform CPA on $W_{2,6}$ and $W_{3,7}$, including half of all key bits of GIFT-128. The CPA guess results for each bit is presented in Table 4.

Note that we are guessing 2 bits of 2 different subkeys each time, hence the guessed bits and real key bits are expressed as $(W2||W3)[i]||(W_6||W_7)[i]$, where i indicates the bit location from the lowest. In the result above, the average ranking of the correct key is 2.3125, which is similar to the theoretical result of 2.5 for any random guess. Furthermore, the success rate of key guessing over all bits is 34.375%, which is a slight increase compared to the random guess of 25%, but still far from restoring the real master key.

For trace set (H), we have the following CPA results in Table 5.

The hardware implementation results are similar to the previous ones overall. The average rank order of correct keys in the correlation results is 2.1875, and the percentage of correct guessed keys is 28.125%. In general, CPA fails to perform effective attacks on the given software and hardware implementations.

References

[BCD+13] Georg T. Becker, Jim Cooper, Elizabeth K. DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, T. Kouzminov, Andrew J. Leiserson, Mark E. Marson, Pankaj Rohatgi, and Sami Saab. Test vector leakage assessment (tvla) methodology in practice. 2013.

[BCI+20] Subhadeep Banik, Avik Chakraborti, Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. Gift-cofb. Cryptology ePrint Archive, Paper 2020/738, 2020. https://eprint.iacr.org/2020/738.

[HBB20] Xiaolu Hou, Jakub Breier, and Shivam Bhasin. Dnfa: Differential no-fault analysis of bit permutation based ciphers assisted by side-channel. Cryptology ePrint Archive, Paper 2020/1554, 2020. https://eprint.iacr.org/2020/1554.

[MRSS18] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage detection with the x2-test. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):209–237, Feb. 2018.