Correlation Power Analysis on the PRESENT Block Cipher on an Embedded Device

Owen Lo
Edinburgh Napier University
United Kingdom
o.lo@napier.ac.uk

William J Buchanan
Edinburgh Napier University
United Kingdom
b.buchanan@napier.ac.uk

Douglas Carson
Keysight Technologies
United Kingdom
douglas_carson@keysight.com

ABSTRACT

Traditional cryptographic techniques have proven to work well on most modern computing devices but they are unsuitable for devices (e.g. IoT devices) where memory, power consumption or processing power is limited. Thus, there has been an increasing amount of work on the design and implementation of lightweight cryptographic algorithms to provide a solution for running cryptography on low resource devices. One particular cryptographic algorithm designed specifically to be used on low resource devices is the PRESENT algorithm. Although the design of PRESENT provides a small memory footprint alongside low power consumption our results show it is susceptible to information leakage when power analysis is performed against a device running this algorithm. In this paper, we present our methodology and results on performing correlation power analysis against this light weight block cipher. Our chosen device under test is an Arduino Uno which was programmed to run the Add Round Key and S-Box functions of PRESENT during the first round of encryptions. Results demonstrate that the Add Round Key function is susceptible to information leakage but a high number of false-positives were observed. Greater success was obtained when targeting the S-Box of the PRESENT algorithm and we were able to derive the first 8 bytes of the key.

CCS CONCEPTS

• **Security and privacy → Cryptanalysis and other attacks**; **Side-channel analysis and countermeasures**; • **Computer systems organization →** *Embedded hardware*; *Embedded software*;

KEYWORDS

Side channel attacks, power analysis, cryptography, PRESENT, Internet of Things

ACM Reference Format:
Owen Lo, William J Buchanan, and Douglas Carson. 2018. Correlation Power Analysis on the PRESENT Block Cipher on an Embedded Device. In *ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany*. Hamburg, Germany, 6 pages. https://doi.org/10.1145/3230833.3232801

ARES 2018, August 27–30, 2018, Hamburg, Germany

ACM ISBN 978-1-4503-6448-5/18/08…$15.00
https://doi.org/10.1145/3230833.3232801

1 INTRODUCTION

Traditional cryptographic methods such as AES (symmetric cryptography), RSA (asymmetric cryptography) and SHA-256 (hashing functions) have proven to work well on most modern computing devices but they are unsuitable for devices where memory, power consumption or processing power is limited. Thus, there has been an increasing amount of work on the design and implementation of lightweight cryptographic algorithms to provide a solution for running cryptography on low resource devices.

In the context of this paper, our focus is on the side channel security of the PRESENT block cipher [1]. PRESENT is one of two block ciphers which is deemed suitable for lightweight cryptography within ISO/IEC 29192-2:2012 [5] (the other being CLEFIA [12] which is outwith the scope of this paper). Although the design of the PRESENT algorithm provides a small memory footprint alongside low power consumption our results show it is susceptible to information leakage when power analysis is performed against a device running this algorithm. We demonstrate that certain functions of the algorithm may be targeted (Add Round Key and S-Box) and, via a correlation based power analysis attack, we were able to derive 8 out of the 10 byte key used in the PRESENT algorithm. We first begin by giving overview of the PRESENT block cipher.

2 PRESENT ALGORITHM OVERVIEW

PRESENT is a block cipher based algorithm. It was designed as a lightweight algorithm for use in devices where energy consumption and/or memory is limited (e.g. IoT devices). The block length (i.e. the data that is to be encrypted or decrypted) is 64 bits while the key size may be 80 bits or 128 bits. The designers of PRESENT have stated that they do not expect the 128 bit key to be used in practical implementations [1] thus our focus is on the 80 bit key variant of this algorithm.

In total, the PRESENT algorithm requires 31 rounds of operation (Figure 1) to encrypt or decrypt a single block of data. The core functions which are conducted in each round of operation include: Key Scheduling, Add Round Key, S-Box Substitution and Permutation. During encryption, the Add Round Key function is first applied to our input (i.e. block of data to be encrypted) where the plaintext values are XOR'd against the first 64 bits of the key value. Next, the S-Box Substitution function is applied whereby each value from the Add Round Key result is replaced with a 'substitute' value. Afterwards, the Permutation function is used to further reorder the data. The output of this is then fed into the next round, which follows the same process, but this time our input is from the previous round, and the key values used are computed by the key scheduling function. The reverse of this process is carried out if decryption of data is conducted.

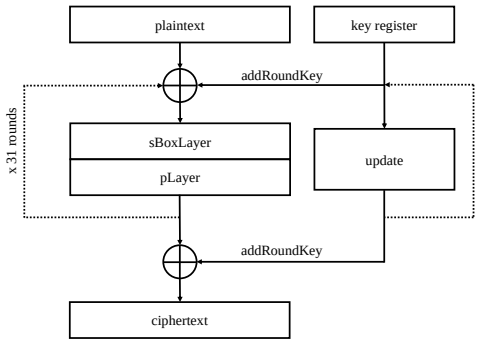

Figure 1: PRESENT Algorithm Description [1]

Our power analysis attack is focused on the first round of encryption during the Add Round Key and S-Box Substitution steps thus the next subsection provides further details on these two functions. Readers are advised to refer to the original paper [1] for full details on the design of this algorithm.

2.1 Add Round Key

Recall that the standard implementation of the PRESENT algorithm will consists of 64 bit block lengths and an 80 bit key. During the Add Round Key operation, an XOR operation is applied between the first 64 bits of the key and the 64 bit block. During the first round of operation, the key used will be the 80 bit value defined by the user while the rounds which follow will use the key values defined by the Key Scheduling function.

The Add Round Key function is represented in (1) where b_j is our 64 bit block defined in the range of $b_{63}...b_0$ while k is our current round key thus $k_i = k^i_{63}...k^i_0$ for $1 \leq i \leq 32$.

$$b_j \rightarrow b_j \oplus k^i_j \tag{1}$$

2.2 S-Box Substitution

Table 1 depicts the S-Box used by the PRESENT algorithm. It is a 4 bit to 4 bit S-Box. In other words, $b_{63}...b_0$ from the Add Round Key output is split into sixteen 4 bit values and used as the lookup against the S-Box. The block value is then replaced with the looked up S-Box value. To provide an applied example, if the input is hexadecimal value 0 the S-Box output would be C.

Table 1: PRESENT 4 Bit S-Box [1]

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

3 DESIGN & IMPLEMENTATION

Building upon previous work in side channel attacks, we conduct power analysis attacks on PRESENT to determine whether the keys used by this algorithm is susceptible to leakage. In summary, power analysis aims to reveal information leakage (e.g. the secret key) of a device under test by analysing the characteristics of the devices power consumption during cryptographic operations.

Two well known techniques in power analysis are differential power analysis (DPA) and correlation power analysis (CPA). DPA attacks involve finding significant differences between sets of power traces relative to a secret key guess while CPA involves modeling the hyopthesised power model of each key guess. The correct secret key guess will exhibit significant difference in the power traces for DPA while the greatest level of correlation between the model and real life power trace indicates the correct key in CPA. For further background and information on the concepts behind power analysis, readers are advised to refer to the works of [2, 7, 8].

In the scope of this work, we chose to use Correlation Power Analysis (CPA) and focused on two areas of the PRESENT algorithm to attack: Add Round Key and S-Box Output. In each attack, we model the power output of the chosen function against a key guess using a Hamming Weight Model. The modelled power output are then compared against our real-life power traces using linear correlation. The PRESENT algorithm's functions are implemented on an Arduino Uno.

3.1 Experiment Setup

The following equipment and tools were used in the experiments carried out:

- **Arduino Uno** - The device under test which runs the Add Round Key and S-Box step of PRESENT algorithm. The software implementation of this code is described in Section 3.3 and 3.4.
- **Keysight MSOX4104A** - The oscilloscope used to gather power traces as the Arduino Uno runs the PRESENT encryption functions.
- **Keysight N2894A & 1160A Probe** - Passive oscilloscope probes used for monitoring the power consumption and trigger on the Arduino Uno respectively. The capture workflow we applied is described in the next subsection in greater detail.
- **Apple Macbook Pro** - 3.1 GHz Intel i7 based laptop used for communication and retrieving data from the Arduino Uno and oscilloscope respectively.

3.2 Data Capture Workflow

Figure 2 depicts the data capture work flow in conducting power analysis on the PRESENT algorithm running on the Arduino Uno.

(1) We issue a command to begin the encryption functions (i.e. the Add Round Key and S-Box Output on Round 1 of PRESENT algorithm) on the Arduino Uno. Known plaintext values are used for encryption purposes in this attack.
(2) Each time the Arduino Uno runs the PRESENT encryption functions the oscilloscope captures the waveform.
(3) Each instance of data capture consists of 5 averaged waveforms against a known plaintext value. This waveform is sent to the laptop for processing later.
(4) Once 256 plaintext values (i.e. 00, 00, 00, 00, 00, 00, 00, 00 to FF, FF, FF, FF, FF, FF, FF, FF in hexadecimal) have been enumerated we can process the captured waveforms and perform a Hamming Weight based CPA attack on the Add Round Key and S-Box output functions.

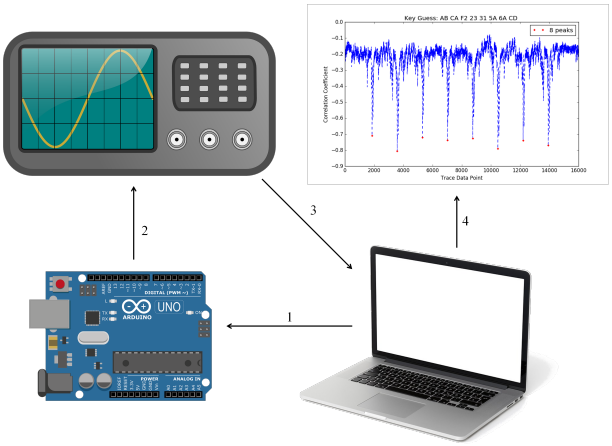

Figure 2: Data Capture Workflow

The next subsection proves details on how we implemented the Add Round Key and S-Box output for the Arduino Uno and CPA attack respectively.

3.3 Add Round Key Implementation

Our implementation of Add Round Key and S-Box substitution on the Arduino Uno was based on code originally derived from [6, 11]. To validate the correctness of code, test vectors provided by [1] and [13] were used. Version 1.8.1 of the Arduino Integrated Development Environment (IDE) was used for writing the code and the default settings were used during compilation and program upload.

The Add Round Key implementation, as programmed on the Arduino Uno, is shown in the code snippet below.

```
state[0] = plain[0] ^ key[0];
state[1] = plain[1] ^ key[1];
state[2] = plain[2] ^ key[2];
state[3] = plain[3] ^ key[3];
state[4] = plain[4] ^ key[4];
state[5] = plain[5] ^ key[5];
state[6] = plain[6] ^ key[6];
state[7] = plain[7] ^ key[7];
```

The variable $state_0 ... state_7$ is the state which will be captured after each Add Round Key operation. The variables $plain_0 ... plain_7$ and $key_0 ... key_7$ are the 64 bit plain text values to be encrypted and the first 64 bits of the key used for encryption respectively.

3.4 S-Box Output Implementation

Our S-Box output implementation, as programmed on the Arduino Uno, is shown in the code snippet below:

```
for(int i =0; i < 8; i++)
{
state[i] =  sbox[state[i] >> 4] << 4 |
            sbox[state[i] & 0xF];
}
```

The variables $state_0 ... state_7$ are the state which will be captured after each S-Box substitution. Note that the state values used in the S-Box lookup are derived from the previous Add Round Key step. The first part of the code:

```
sbox[state[i] >> 4] << 4
```

performs a S-Box lookup against the first 4 most significant bits of $state_i$ while:

```
sbox[state[i] & 0xF]
```

performs a lookup on the 4 least significant bits. The bitwise OR operator | is used to combine the two results into a single 8 bit value as output.

3.5 Implementation of CPA Attack using Hamming Weight

The Hamming Weight algorithm is used to model the expected power output during the Add Round Key and S-Box output of the PRESENT algorithm against a key guess. In essence, the Hamming weight of our implementation is simply the count of the number of bits set to true in a hypothesised power output. Thus, if our hypothesised output is hexadecimal value *AB* then the Hamming Weight would be 5 units since *AB* is represented as 1010 1011 in binary.

Performing a CPA attack on the PRESENT algorithm's Add Round Key and S-Box output functions follow the same concepts. First, we use Hamming Weight to model the hypothesised power consumption of the state output based on a key guess. Thus, for Add Round Key, we model our power consumption based on (2) by computing the Hamming Weight of the output of $state_i$ where i is in the range 0 to 7 and $plain_i$ is known to the attacker.

$$state_i = plain_i \oplus keyGuess_i \tag{2}$$

For the S-Box output, we model our power consumption of the state output as shown in (3) where $state_i^{0\cdots3}$ is the first 4 most significant bits produced during the Add Round Key step and $state_i^{4\cdots7}$ are the 4 least significant bits.

$$state_i = sbox[state_i^{0\cdots3} | state_i^{4\cdots7}] \tag{3}$$

Once a model of each key guess (256 in total for each function) has been produced, and the power traces have been captured on on the device whilst the relevant PRESENT encryption functions were running, we may compare our hypothesised power trace against the real-life power traces to determine which key guesses are correct. We chose to apply Pearson's Correlation Coefficient to find correlation between our model and real-life power traces. Pearson's Correlation Coefficient is represented in (4) where W represents our real-life power traces values while P is our predicted Hamming weight values against a key guess. A correct key guess may demonstrate strong linear correlation while an incorrect key guess should demonstrate no correlation.

$$p(W,P) = \frac{Cov(W,P)}{\sqrt{Var(W)}\sqrt{Var(P)}} \tag{4}$$

We present out results in the section which follows.

Table 2: $plain \oplus Key$ Lookup (4 by 4 bit)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	0	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	3	0	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	2	1	0	7	6	5	4	B	A	9	8	F	E	D	C
4	4	5	6	7	0	1	2	3	C	D	E	F	8	9	A	B
5	5	4	7	6	1	0	3	2	D	C	F	E	9	8	B	A
6	6	7	4	5	2	3	0	1	E	F	C	D	A	B	8	9
7	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8
8	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	9	8	B	A	D	C	F	E	1	0	3	2	5	4	7	6
A	A	B	8	9	E	F	C	D	2	3	0	1	6	7	4	5
B	B	A	9	8	F	E	D	C	3	2	1	0	7	6	5	4
C	C	D	E	F	8	9	A	B	4	5	6	7	0	1	2	3
D	D	C	F	E	9	8	B	A	5	4	7	6	1	0	3	2
E	E	F	C	D	A	B	8	9	6	7	4	5	2	3	0	1
F	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Table 3: $sbox[plain \oplus Key]$ Lookup (4 by 4 bit)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2
1	5	C	B	6	0	9	D	A	E	3	8	F	7	4	2	1
2	6	B	C	5	A	D	9	0	F	8	3	E	1	2	4	7
3	B	6	5	C	D	A	0	9	8	F	E	3	2	1	7	4
4	9	0	A	D	C	5	6	B	4	7	1	2	3	E	F	8
5	0	9	D	A	5	C	B	6	7	4	2	1	E	3	8	F
6	A	D	9	0	6	B	C	5	1	2	4	7	F	8	3	E
7	D	A	0	9	B	6	5	C	2	1	7	4	8	F	E	3
8	3	E	F	8	4	7	1	2	C	5	6	B	9	0	A	D
9	E	3	8	F	7	4	2	1	5	C	B	6	0	9	D	A
A	F	8	3	E	1	2	4	7	6	B	C	5	A	D	9	0
B	8	F	E	3	2	1	7	4	B	6	5	C	D	A	0	9
C	4	7	1	2	3	E	F	8	9	0	A	D	C	5	6	B
D	7	4	2	1	E	3	8	F	0	9	D	A	5	C	B	6
E	1	2	4	7	F	8	3	E	A	D	9	0	6	B	C	5
F	2	1	7	4	8	F	E	3	D	A	0	9	B	6	5	C

4 RESULTS

Figure 3 provides an example result of the key guess produced when attacking the Add Round Key function while Figure 4 gives an example result of attacking the S-Box output. In both instances, the key used in the algorithm was: *AB, CA, F2, 23, 31, 5A, 6A, CD, EF, 1F*. The 'peakutils' [10] library was used to automate the detection of significant negative linear correlation coefficients in our results (i.e.key guesses which exhibited strong correlation between model and real-life power traces).

Out of a total of 8 possible keys, CPA attack against the Add Round Key function was able to reveal six correct keys. CPA attack against the S-Box output was successful for the first 8 bytes of the key.

4.1 Discussion

With the Add Round Key function, we note that although CPA has the potential to derive correct key guesses we found that repeated experiments in power analysis focused on this function would not produce reliable results. The attack against the Add Round Key function is susceptible to false-positives due to the linear nature of XOR. This is most evident when comparing columns 0 and F in the lookup tables for Add Round Key (Table 2) and S-box (Table 3) when 4 bits are applied to each function. With Add Round Key the majority of results are incremental (relative to its neighbour) while S-Box has more variation due to it's non-linear design [1]. Thus, CPA attacks on Add Round Key has the potential for separate key guesses to produce a Hamming weight value which is only 1 bit different from its neighbouring value which are more difficult to distinguish when comparing the models against our real-life power traces.

On the other hand, we found success with the CPA attack against the S-Box output of the PRESENT algorithm. In the experiments we observed, all 8 (out of 10 in total) keys in the algorithm were correctly guessed by our Hamming Weight power model. We address a theoretical example of how an attacker may derive the final 2 bytes of the PRESENT algorithms private key in the subsection which follows.

4.2 The Final 2 Bytes of Key

Recall that PRESENT is a 80 bit key block cipher. Under the assumption that an attacker is capable of deducing the first 8 bytes of the key, potential exists to perform a brute force attack on the final 2 byte key. Naturally a response which indicates success to the attacker will be necessary in order for this brute force attack to work.

In an ideal scenario, one may be able to correctly brute force the remaining key in the scope of seconds since the our key space is 256^2. In other words, only 65,536 guesses in total would be necessary. In a simulated environment, we were able to successfully brute force the final 2 bytes of the key (hexadecimal values *EF* and 1*F*) in 166 μs based on a average of 1000 iterations.

5 RELATED WORKS

Related works on side channel attacks against the PRESENT algorithm first include Cube Based attacks as demonstrated by [15] and [16]. Cube based attacks aims to represent certain outputs of a cryptographic algorithm as low-degree polynomial. Theoretically, by providing certain inputs to the polynomial one has the capability of deriving the secret key of the algorithm. Our work differs from cube based attacks as we aim to monitor the power consumption of a device to reveal information leakage rather than apply a cryptanalysis technique.

More recently, [3] demonstrated the capability of performing a Differential Power Analysis (DPA) attack on the S-Box of the PRESENT algorithm. Our technique differs since we use CPA in place of DPA. Additionally, as power analysis against the AES algorithm has shown [9], although DPA is certainly a valid technique it appears quite susceptible to noise and interference thus increasing the probability of false-positives and incorrect results.

Most closely aligned to our work is [14] who demonstrate the capability of using CPA via a Hamming Distance model to target the hardware implementation of the PRESENT algorithm. Our work differs here since our attack is against a software implementation of PRESENT and we chose to use a Hamming weight model instead. Finally, the work of [4] has also demonstrated capabilities of CPA attack on PRESENT however, their work is based on theoretical

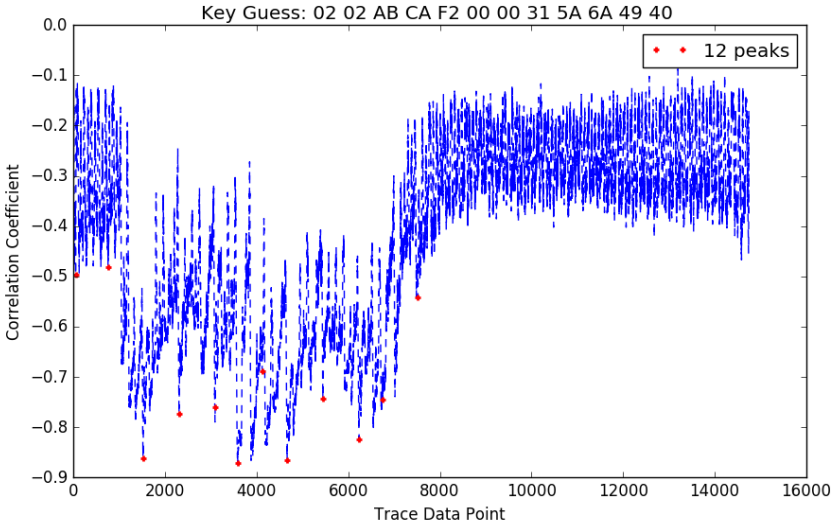

Figure 3: CPA Attack on Add Round Key

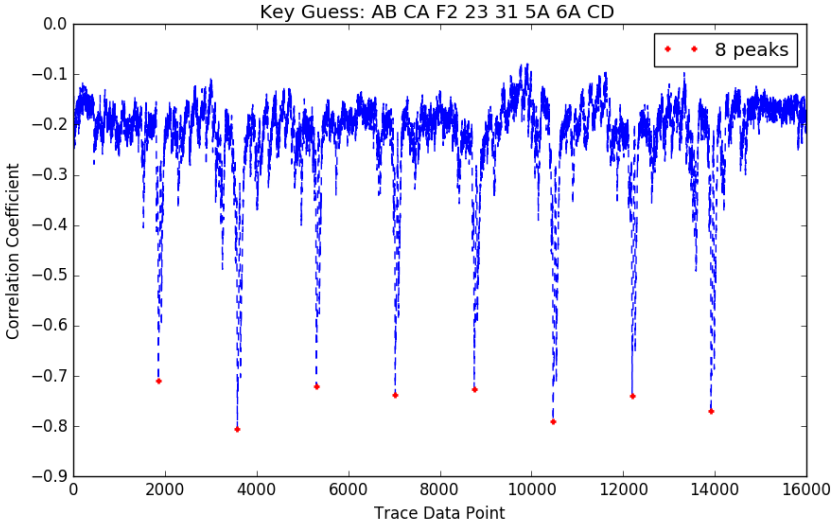

Figure 4: CPA Attack on S-Box Output

models rather than the algorithm running on physical hardware (an Arduino Uno in our case).

6 CONCLUSION

In this paper, we have demonstrated the capabilities of performing CPA on the PRESENT algorithm. Our targets include the Add Round Key and S-Box functions of the algorithm during the first round of encryption. Although information leakage is susceptible with Add Round Key the linear nature of this function produces a high number of false-positives. On the other hand, we found that the S-Box is a more ideal target and 8 (out of the 10 in total) keys of the PRESENT algorithm were successfully retrieved when attacking this function.

In our concluding remarks, we acknowledge a few limitations in this paper: 1) the attack has been conducted against a software implementation of PRESENT rather than a hardware implementation; 2) our attack was conducted under the assumption that the implementation of PRESENT stores its state values as 8 bit states (it is possible to store the states as 4 bit values instead); and 3) experiments were conducted in a controlled white-box testing environment using an Arduino Uno rather than a consumer ready

device. Each of these limitations should be considered for future work. In particular, we believe that further research should be conducted in IoT devices which make use of light weight cryptography to ensure they are well defended against any form of side channel attack.

ACKNOWLEDGMENTS

This work was supported by the Innovation Centre for Sensor and Imaging Systems (CENSIS).

REFERENCES

[1] A Bogdanov, L R Knudsen, G Leander, C Paar, A Poschmann, M.J.B Robshaw, Y Seurin, and C Vikkelsoe. 2007. PRESENT : An Ultra-Lightweight Block Cipher. *Springer Berlin Heidelberg* (2007), 450–466. DOI : http://dx.doi.org/10.1007/978-3-540-74735-2

[2] Jean-Sébasticn Coron, Paul Kocher, and David Naccache. 2001. Statistics and Secret Leakage. In *Financial Cryptography: 4th International Conference, FC 2000 Anguilla, British West Indies, February 20–24, 2000 Proceedings*, Yair Frankel (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 157–173. DOI : http://dx.doi.org/10.1007/3-540-45472-1_12

[3] Xiaoyi Duan, Qi Cui, Sixiang Wang, Huawei Fang, and Gaojian She. 2016. Differential power analysis attack and efficient countermeasures on PRESENT. In *Proceedings of 2016 8th IEEE International Conference on Communication Software and Networks, ICCSN 2016*. 8–12. DOI : http://dx.doi.org/10.1109/ICCSN.2016.7586627

[4] Annelie Heuser, Stjepan Picek, Sylvain Guilley, and Nele Mentens. 2014. Side-channel Analysis of Lightweight Ciphers : Does Lightweight Equal Easy ? (2014), 1–14.

[5] ISO. 2012. ISO/IEC 29192-2:2012 Information technology - Security techniques - Lightweight cryptography - Part 2: Block ciphers. (2012). https://www.iso.org/standard/56552.html

[6] Dirk Klose. 2008. C PRESENT Implementation (8 Bit). (2008). http://www.lightweightcrypto.org/downloads/implementations/PRESENT8-bit_implementation.rar

[7] Paul Kocher, Joshua Ja, and Benjamin Jun. 2011. Differential Power Analysis. *Journal of Cryptographic Engineering* (2011), 1–10. DOI : http://dx.doi.org/10.1007/3-540-48405-1_25

[8] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology — CRYPTO' 99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings*, Michael Wiener (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 388–397. DOI : http://dx.doi.org/10.1007/3-540-48405-1_25

[9] Owen Lo, William J Buchanan, and Douglas Carson. 2016. Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA). *Journal of Cyber Security Technology* 1, 1 (2016), 1–20. DOI : http://dx.doi.org/10.1080/23742917.2016.1231523

[10] Lucas Hermann Negri. 2018. PeakUtils. (2018). https://pypi.python.org/pypi/PeakUtils

[11] Christophe Oosterlynck and Philippe Teuwen. 2008. Python PRESENT implementation. (2008). http://www.lightweightcrypto.org/downloads/implementations/pypresent.py

[12] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. 2007. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *Fast Software Encryption, 14th International Workshop, {FSE} 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*. 181–195. DOI : http://dx.doi.org/10.1007/978-3-540-74619-5_12

[13] Thomas Siebert. 2008. Testvectors for PRESENT. (2008). http://www.lightweightcrypto.org/downloads/implementations/present21.tgz

[14] Chenxu Wang, Mingyan Yu, Jinxiang Wang, Peihe Jiang, and Xiaochen Tang. 2013. A more practical CPA attack against present hardware implementation. *Proceedings - 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, IEEE CCIS 2012* 3 (2013), 1248–1253. DOI : http://dx.doi.org/10.1109/CCIS.2012.6664584

[15] Lin Yang, Meiqin Wang, and Siyuan Qiao. 2009. Side channel cube attack on PRESENT. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5888 LNCS, 60525201 (2009), 379–391. DOI : http://dx.doi.org/10.1007/978-3-642-10433-6_25

[16] Xinjie Zhao, Tao Wang, and Shize Guo. 2011. Improved Side Channel Cube Attacks on PRESENT. 165 (2011), 1–11. http://eprint.iacr.org/2011/165