Correlation Power Analysis of SipHash

Matúš Olekšák, Vojtěch Miškovský
Department of Digital Design
Faculty of Information Technology
CTU in Prague, Czech Republic
{oleksmat,miskovoj}@fit.cvut.cz

***Abstract*—SipHash is ARX-based pseudorandom function optimized for short inputs. It was developed as a hash table lookup function, but it is also used for MAC generation. At the time of writing, there was no side-channel attack on SipHash known to us. This work is about application of CPA attack on SipHash. Attack was performed on ChipWhisperer CW308 UFO Board with STM32F0 target. Approximately 800 power traces were needed for succesful attack. There are two parts of attack as there is secret key divided into two halves. Leakage information from XOR was used to attack cipher key. The main contribution of this work is power model of binary addition including carry propagation.**

***Index Terms*—CPA, SipHash, ARX, side-channel, Chipwhisperer**

I. Introduction

SipHash [2] is promising modern ARX-based pseudorandom function, that is also used for MAC generation. And because of that, it is important to know whether it is resilent against side-channel attacks, as is commonly believed [5] about ARX-based algorithms. It is few times faster than AES-CMAC, and that implies usage in time critical systems or lightweight systems, e.g., Internet of Things.

It is widely known, that AES is prone to side-channel attacks because of low differential uniformity of S-Boxes [6]. On the other hand, there is not that much research about side-channel attacks on ARX-based algorithms. This is probably caused by lower popularity of such algorithms. In this work, we explore vulnerability of SipHash to Correlation Power Analysis [4].

Correlation Power Analysis (CPA) was introduced in 2004 by Brier et al. [4] and it is non-profiled side-channel attack. In the first phase of the attack, power consumption traces are measured. After this physical prerequisite, all the remaining steps are purely computational. Leakage function needs to be chosen according to the attacked algorithm and its implementation. The secret information (usualy the cipher key) needs to be split into smaller parts (e.g., bytes) to make its enumeration computationaly feasible (e.g., 256 posible key candidates for a byte). Then the power consumption is estimated for each key candidate using the chosen leakage function. In the final step, correlation between measured and estimated power consumption is calculated. The best correlated key candidate is supposed to be the correct cipher key.

II. Our contribution

Main goal of this work was succesful side-channel attack on SipHash. It was achieved by CPA, which was adjusted for each half of the cipher key. Crucial part of this method is carry flag-based power model.

III. Analysis and related work

A. SipHash description

SipHash is ARX-based pseudorandom function, that can be used for 64-bit MAC generation. There are 4 state variables called $v0$-$v3$, their initial default values corresponds to "somepseudorandomlygeneratedbytes" in ASCII. Nevertheless, any other initial values can be used as long as $v0$ and $v1$ differ from $v2$ and $v3$. For the sake of clarity, subscript in the name of the variable indicates round number and it represents value of variable in the beginning of chosen round, otherwise it represents the initial value of variable.

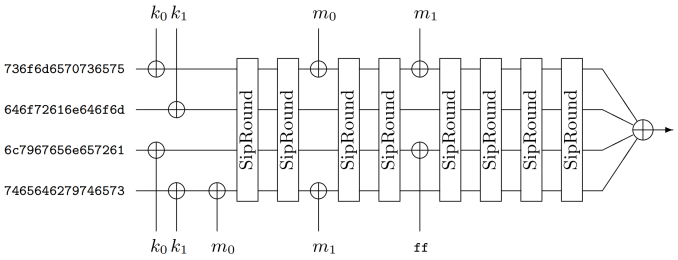

Fig. 1. SipHash Diagram [2].

128-bit key is halved into two 64-bit values $k0$ and $k1$. Before the first round, initial transformation is done: $v0_1 = v0 \oplus k0$, $v1_1 = v1 \oplus k1$, $v2_1 = v2 \oplus k0$, $v3_1 = v3 \oplus k1 \oplus m0$, where $m0$ represents the first 64 bits of plaintext. Default number of rounds is 2 per 64 bits of plaintext, and 4 finalization rounds.

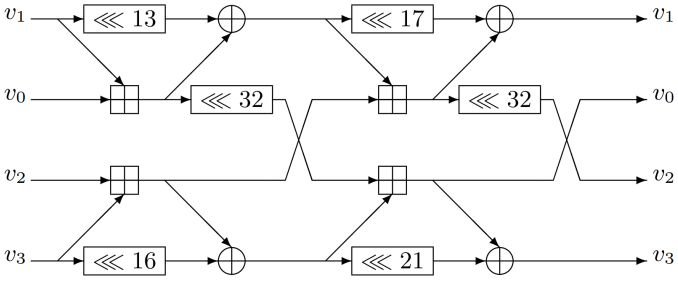

Fig. 2. SipHash Round Diagram [2].

SipHash is an ARX algorithm, and because addition and rotation does not leak information as good as XOR, it should

be convenient to focus on XOR operations in computation. As plaintext is XORed with $v3$ in the beginning, it is good idea to focus on $v3_1$ and get the second half of cipher key from this part of computation. Knowing $v3_1$ value, it should be possible to discover the rest of the cipher key later.

B. ARX-based algorithms CPA Attacks

Because of SipHash being ARX-based function, it means, that there is not much research about its side-channel power analysis because of its commonly incorrect conclusion [5], that they are almost impossible to attack. It is true, that XOR does not leak that much information as, e.g., S-Box in AES, but that is solvable using higher number of traces. Also, SipHash is not that much widespread as, e.g., AES, so that implies its lower popularity in research dealing with power analysis.

C. SPARX CPA

An attack on SPARX algorithm was presented in [3], and its authors demonstrated, that single bit DPA is ineffective on modular addition and so is CPA [4], because there were many incorrect key candidates yielding at higher correlations than the correct ones. Afterward, authors tried to attack on rotation and XOR. And it turned out to be better target of information leakage. Using CPA on XOR, authors were able to discover encryption key with tens of traces.

IV. Attack

Because of SipHash's way of dealing with $k0$ and $k1$ is different, it is necessary to attack differently on $k0$ and $k1$. We need to discover $k1$ first as its value is needed for attack on $k0$. Regarding informations from experiment presented in [3], we decided to focus on XOR operations in this attack. The first step is obvious to attack on output of $v3 \oplus k1 \oplus m0$.

Then, with fully known $v3_1$ value, the most straightforward way of discovering $k0$ is through intermediate value $(v2_1 + v3_1) \oplus (v3_1 <<< 16)$. As long as full $v3_1$ value is known, it is only necessary to enumerate all possible $v2_1$ values to evaluate $k0$ candidates.

A. Measurement

All testing is performed on STM32F0 target of Chipwhisperer CW308 UFO Board and power traces are captured with ChipWhisperer-Lite. Measurement is setup to capture 2000 power traces with 20000 samples, which cover first three rounds. Plaintext is randomly generated. Measurement is started by trigger, which results in obtaining correctly aligned power traces. Reference implementation of SipHash [1] is used with default initial values of $v0$-$v3$.

B. Attack on k1

We decided to attack using intermediate value $f_{k1}(m0) = v3 \oplus k1 \oplus m0$. As a leakage function, $-HW(f_{k1}(m0))$ is used, because it was experimentally discovered that it is generating better results than commonly used $HW(f_{k1}(m0))$, where $HW(x)$ is Hamming Weight of x. Pearson correlation between the power consumption estimated by the leakage function and the measured power traces is calculated.

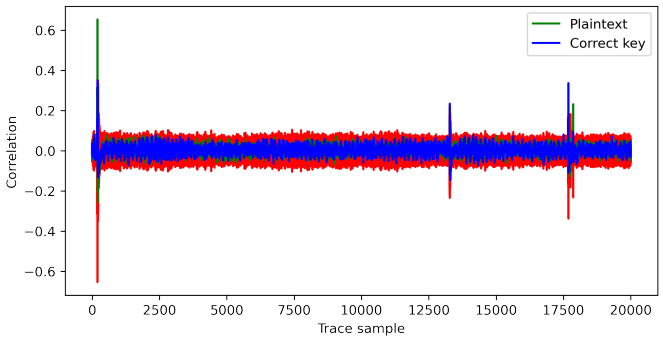

Fig. 3. Correlation graph of $k1$ key attack.

Graph in Figure 3 shows significant peaks of correlation values. All the significant parts are showed in detail on graphs in Figure 4, 5 and 6.

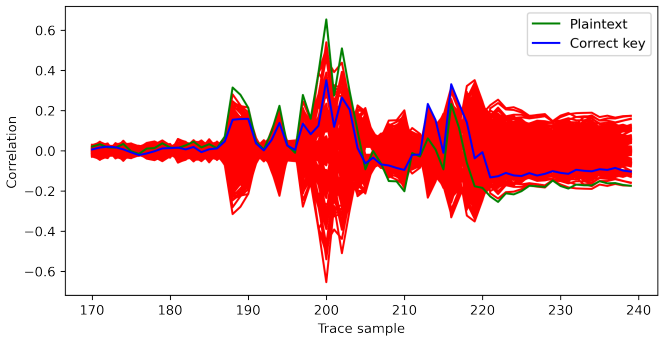

Fig. 4. Detail of first correlation graph peak of $k1$ key attack.

The point with highest correlation represents key candidate 0x00, therefore, it is correlated with plaintext (resp. $v3 \oplus m0$). It can be used for power traces alignment and cropping, when trigger is not available, since the correlation with the first byte of plaintext peaks before the first round, and it also peaks before the third round, as can be seen in Figure 6.

Also, there is a point, where the correct key has the highest correlation, but its peak is hardly distinguishable from peaks of other keys.

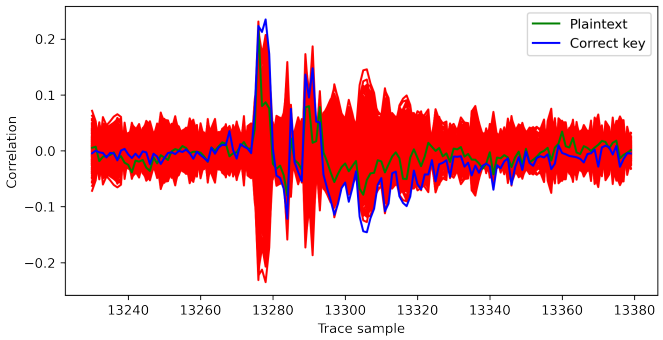

Fig. 5. Detail of second correlation graph peak of $k1$ key attack.

The origin of correlation peaks in Figure 5 is not clear as processing of the targeted intermediate values is not expected in this part of computation.

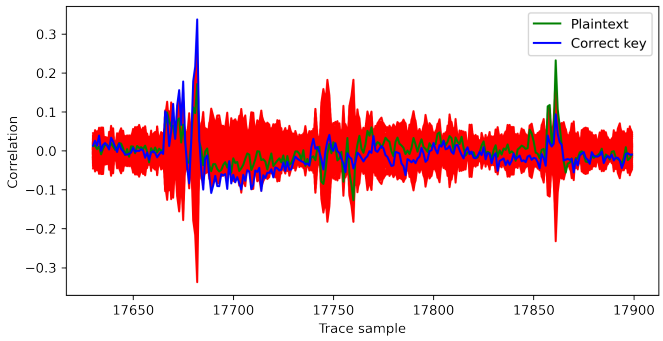

Fig. 6. Detail of third correlation graph peak of $k1$ key attack.

On the correlation graph in Figure 6, there is a correlation peak of correct key, and few samples later, there is a correlation peak of plaintext from third round. The origin of correlation peak of the correct key at this part of computation is also unclear.

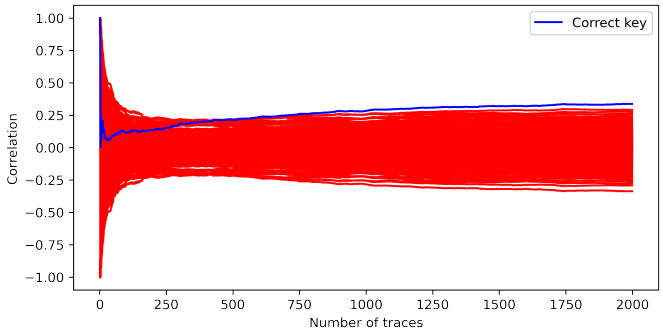

Fig. 7. Correlation graph according to number of traces of $k1$ key attack.

Graph in Figure 7 shows how correlation of correct key evolves with rising number of traces. The correct key byte has the highest correlation with more than 800 power traces.

C. Attack on k0

This part of attack is more interesting and more complex. We choose $f_{k0}(m0) = (v2_1 + v3_1) \oplus (v3_1 <<< 16)$ as the leakage function. $k1$ is known from the first part, correct values of $v3_1$ can be computed.

The binary addition complicates things a little bit. To overcome this problem, we decided to simulate binary adder and incorporate the carry flag value. In the beginning, the carry flag is set to 0. We compute $(v2 \oplus k0) + v3_1 + carry$.

Each hypothetical byte carry flag is saved for next byte attack (using $result >> 8$) and removed from sum result (using $result\& = 0xFF$). It is simulated, what data is available on bus, since it constitutes the main power consumption of microprocessor. Then it is XORed with $v3_1 <<< 16$ and correlation between measured and estimated power consumption is calculated.

The best correlated key candidate is the found key. Saved carry flag from each byte is used in attack on the next byte. And so it continues until all the remaining bytes of $k0$ are discovered.

The proposed method of saving the carry flag and subtracting it from the sum result is essential for finding the correct key.

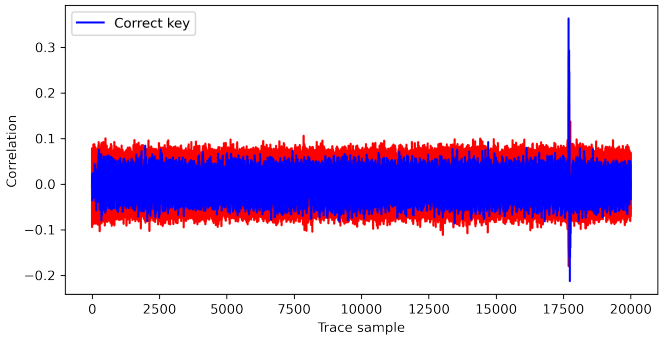

Fig. 8. Correlation graph of k0 key attack on XOR.

As is visible on graph in Figure 8, there is only single one correlation peak. It is displayed in detail on graph in Figure 9, that it is showing three significant correlation peaks for the correct key and no ghost peaks.

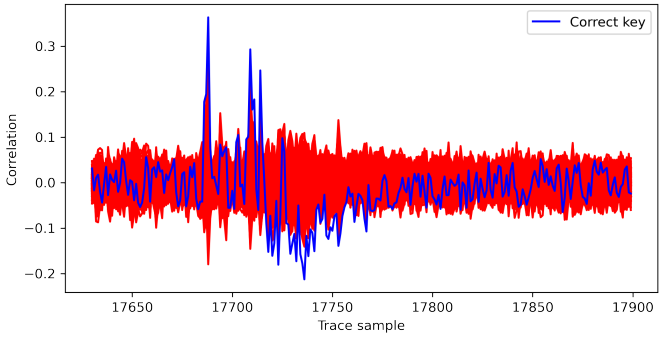

Fig. 9. Detail of correlation graph peak of $k0$ key attack on XOR.

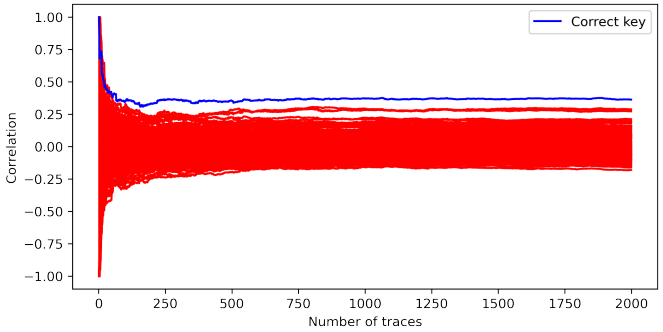

Fig. 10. Correlation graph according to number of traces of $k0$ key attack on XOR.

Figure 10 shows, that lower number of power traces is needed for discovery of the correct key in comparison with $k1$. Approximately 300 power traces are needed to discover all bytes of k0.

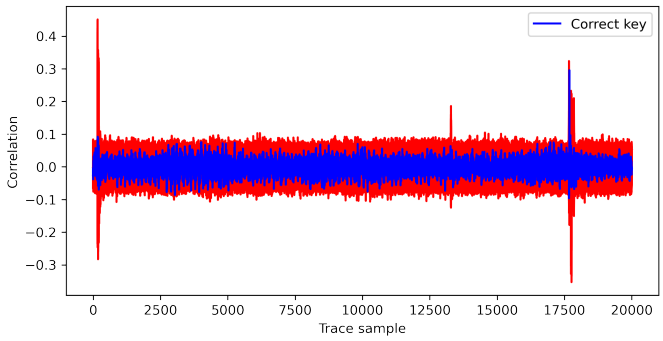

Fig. 11. Correlation graph of $k0$ key attack on addition.

Graph in Figure 11 shows correlation of attack on $f_{k0}(m0) = (v2_1 + v3_1)$. Therefore, the attack is focused on the result of addition instead of XOR. It shows multiple ghost peaks in contrast with attack on XOR operation.

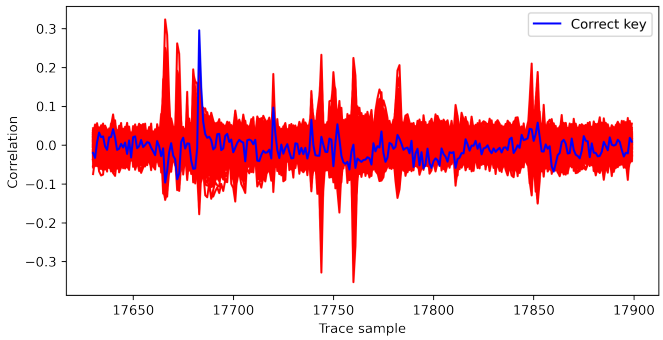

Fig. 12. Detail of correlation graph of $k0$ key attack on addition.

In graph in Figure 12, it is clearly visible, that even though the correct key has correlation peak, it is not the highest one.

D. Summary

It is shown, that SipHash is prone to CPA, however it needs to be customized properly. With different method on $k0$ and $k1$ key value, it is possible to discover the correct key with around 800 power traces. Attack on $k1$ value is not that exceptional, on the other hand $k0$ attack is more advanced. It is because of binary addition between XOR operations. Carry flag propagation is needed in order to correctly estimate the result of addition. Also, it is shown, that it is considerably better to attack on XOR operation instead of binary addition.

V. Conclusion

This work shows that even though ARX-based functions are often considered resistant or irelevant for side-channel attacks, it is not true, as some other research has also already shown. The noticeable difference as opposite to, e.g., AES, is the higher number of power traces needed for full key discovery, because of attacking on XOR function. But there is inconsistency with an attack on SPARX algorithm, because we needed high hundreds of traces instead of mentioned tens of traces.

It is important to do more research on ARX-based functions as they are used more and more in various embedded systems, yet research around them is not that widespread. And that may be one of the reasons, why they are getting so popular - because there is not that much research about their weaknesses.

In this work, we extend existing research of power analysis of ARX-based algorithms by adjusting it for use on SipHash. Succesful CPA attack is presented enabling to discover full secret key.

Main contribution of this work is method of carry propagation in binary addition, that occurs in SipHash round. It was possible to easily recover $k0$ key using this method.

Next steps in our research will include attack on different implementation of SipHash or using presented CPA method on different ARX algorithms. It would be useful to compare microprocessors of different architectures or even different platforms (e.g., FPGAs) as well. We would also like to analyze the different behavior for XOR and addition leakage functions and differences in results presented in [3] and ours.

VI. Aknowledgement

This work has been partially supported by the Student Summer Research program of FIT CTU in Prague and CTU project SGS20/211/OHK3/3T/18.

References

[1] Jean-Philippe Aumasson, “GitHub - veorq/SipHash: High-speed secure pseudorandom function for short messages” [Online], Available: https://github.com/veorq/SipHash

[2] Jean-Philippe Aumasson and Daniel J. Bernstein, “SipHash: a fast short-input PRF”, Cryptology ePrint Archive, Report 2012/351. Available: https://ia.cr/2012/351

[3] Yan, Yan and Oswald, Elisabeth, “Examining the Practical Side Channel Resilience of ARX-Boxes”, Cryptology ePrint Archive, Report 2019/335. Available: https://ia.cr/2019/335

[4] Eric Brier and Christophe Clavier and Francis Olivier, “Correlation Power Analysis with a Leakage Model”, Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings. Available: https://www.iacr.org/archive/ches2004/31560016/31560016.pdf

[5] Alex Biryukov, Daniel Dinu, and Johann Großschädl. “Correlation Power Analysis of Lightweight Block Ciphers: From Theory to Practice”. In Applied Cryptography and Network Security - 14th International Conference, ACNS 2016. Available: https://doi.org/10.1007/978-3-319-39555-5_29

[6] Heuser, Annelie and Rioul, Olivier and Guilley, Sylvain, “A Theoretical Study of Kolmogorov-Smirnov Distinguishers”. Constructive Side-Channel Analysis and Secure Design (2014). Available: https://doi.org/10.1007/978-3-319-10175-0_2