Power Analysis Based Side Channel Attack

Hasindu Gamaarachchi, Harsha Ganegoda, Darshana Jayasinghe, Roshan Ragel
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
hasindu2008@gmail.com

***Abstract*—Power Analysis Attacks which break the key of a cryptosystem by measuring power consumption of a device have become a huge security threat. Advanced Encryption Standard (AES) which takes billions of years to break via a brute force attack can be broken in few minutes using a power analysis attack. Therefore to minimize the threat imposed, research on countermeasures has become extremely important. First we build a testbed for power analysis, which is a set of complete hardware and software components that can be used to do a practical demonstration of a power analysis attack. Then using the testbed, we show that even a latest encryption algorithm like Speck can be still broken in a time less than 1 hour. Despite being an add-rotate-xor cipher that does not even use substitution box operations, we show that Speck can be broken not only on a 8 bit microcontroller but also on a 16 bit microcontroller. Next we practically test and evaluate the effectiveness of some selected countermeasures. We work on both circuit level hardware countermeasures as well as software countermeasures. While experimenting with new ideas for countermeasures finally we provide improvements to an existing countermeasure.**

Keywords*—*Power Analysis Attack, Correlation Power Analysis, Power Measurement Testbed, Speck, Countermeasures for Power Analysis attacks

I. Introduction

Side channel attacks are a type of cryptographic attacks where unintended channels such as heat, sound, power and electromagnetic radiation given out by a cryptosystem are used for breaking the secret key. Power Analysis Attacks which fall under side channel attacks uses the power consumption of the cryptographic device as the side channel. Currently electronic components such as microcontrollers and memory are made out of Complementary Metal Oxide Semiconductor (CMOS) circuits. In CMOS circuits, power consumption depend on switching of transistors. Therefore the power consumption of a computational device depends on the data that is being processed and the type of operation being performed [1]. When power consumption pattern (power traces) of a cryptographic device is captured as shown in Fig. 1, it is possible to derive the secret key of a system. Power traces are obtained for encryption on several plain text samples. Then those power traces are analysed on a computer using techniques such as Simple Power Analysis, Differential Power Analysis (DPA) [2] or Correlation Power Analysis (CPA) [3]. This type of attack has hugely affected the security of embedded devices such as smart card. Even a most used encryption such as Advanced Encryption Standard (AES) can be broken in few minutes. Because of the imposed security risks, research on power analysis attacks and countermeasures has become greatly important.

CPA is a statistical technique which uses Pearson correlation [3]. Power consumption of the device is calculated by using a power model such as hamming weight for each possible key. Then the calculated power data (hypothetical power consumption values) are compared with real power values from power traces using Pearson correlation. The maximum correlation corresponds to the correct key. Due to the advantages CPA has such as requirement of less number of power traces, we use CPA for the research.

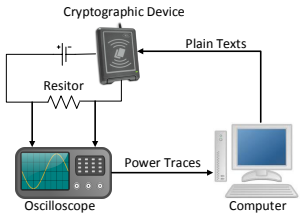


Fig. 1. Power analysis attack

The first requirement for power analysis is a testbed. Few works such as [4], [5] and [6] mention the specifications of their testbed but unfortunately no descriptive steps are provided, making it difficult to reproduce. Building a testbed is a complicated and time consuming task especially without a proper guide. We have contributed by providing a step by step guide for building a testbed from the scratch, so that it would save the time of future researchers. Further we have introduced novel power measurement methods that can break AES even with less than 200 power traces in less than 10 minutes time, while just using passive oscilloscope probes.

Most works such as [4], [5] and [6] focus on attacking AES. Speck is a recent light weight cipher introduced by National Security Agency (NSA) that may become famous among embedded devices [7]. This algorithm has lot of differences with AES and hence the attack approach for AES cannot be used. Up to now according to our best knowledge there is no work on attacking Speck. We show that Speck can be broken in less than 1 hour. We contribute by showing the vulnerability in Speck while emphasizing the need for countermeasures.

Various work such as [1], [2] and [8] propose countermeasures against power analysis attacks. But most of them are not practically demonstrated. Some have used simulators to test their countermeasures but in real world, ideal conditions assumed in simulators are no longer there. Therefore we practically implement some selected countermeasures on the testbed we have created and attack them to analyse effectiveness. Further we check the effectiveness of some ideas of our own while also proposing improvements for existing

countermeasures.

II. Power Analysis Testbed

A. Making of the Testbed

A power analysis testbed includes hardware components (Fig. 2) such as the cryptosystem and the power measurement setup. Software components include programs that automate the power capturing process and the ones that does analysis.

Fig. 3 shows the circuit diagram of a testbed we have created. The cryptosystem which carries out the encryption is an 8 bit microcontroller. It is programmed to run 128 bit AES. The microcontroller is interfaced with the computer via Universal Serial Bus (USB) using a USB to RS232 TTL (Transistor Transistor Logic) converter shown as *FTDI* in Fig. 3. Either the *Vdd resistor* or the *Ground resistor* marked in Fig. 3 can be used for power measurements. The resistor is usually of about 100 ohms. If ground resistor is used, the oscilloscope probe tip is connected to the place marked as *Connection Point 1*. If Vdd resistor is used, the oscilloscope probe tip is connected to the place marked as *Connection point 2*. In both cases, the ground of the oscilloscope is connected to the place marked as *Ground*. The pin marked as *Trigger* is used to provide the trigger for the oscilloscope.

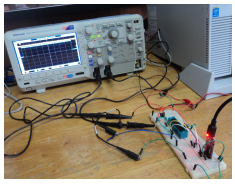


Fig. 2. The testbed implemented on a breadboard

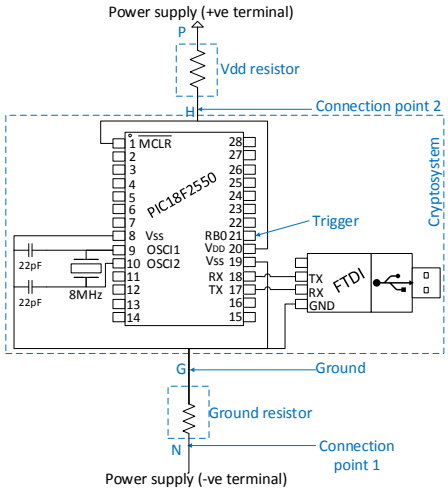


Fig. 3. Circuit diagram for the testbed

Works such as [6] when measuring power, measure the voltage across the resistor. Though Vdd resitor method provides better results (less power traces needed for a successful attack) it requires a differential probe which is costly [6]. We introduce a new technique where the voltage across the microcontroller is measured rather than across the resistor, that still enables the use of a normal oscilloscope probe when using the Vdd resistor method. Also our testbed even on a breadboard enables a successful attack with only about 200 power traces, despite the high level of noise that would be there when implemented on a breadboard. Further, we have made a working setup that uses the internal USB module of a microcontroller which lets the microcontroller be connected directly to the computer even without a USB to RS232 TTL converter.

A digital oscilloscope interfaced with a computer running Matlab is used to collect power traces. The CPA algorithm which is the analysis part. written in Compute Unified Device Architecture (CUDA) C, runs on the Graphics Processing Unit (GPU). Complete source code repository for the testbed can be found at [9].

B. Results

About 200 power traces were enough to derive the key successfully on the discussed testbed that runs AES. It took only 370 seconds to collect the power traces. The analysis was done on a NVIDIA Tesla C2075 GPU where the time taken was only 8 seconds. Therefore the total time taken was even lesser than 10 minutes.

Fig. 4 shows how the correlation coefficient changes with the number of power traces used. It contains graphs for all possible keys. Here it is clearly notable that one key lies significantly higher from the others. This one being the highest correlated key turns out to be the correct key. Note that even at 50 traces, the correct key has become significantly higher than the others. Therefore the minimum number of traces needed to attack is even lesser, which shows that the attack is feasible in very small time.

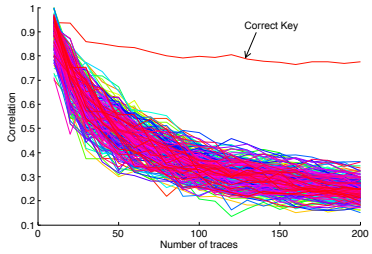


Fig. 4. Variation of correlation coefficient with the number of power traces

III. Attacking Speck Encryption Algorithm

A. Speck Implementation

Fig. 5 shows the overview of the Speck algorithm. There it uses only three operations namely Addition, Rotation and Xor (ARX). It does not include any complex operations such as substitution box (sbox) lookups and matrix multiplication as in AES. Therefore, the algorithm performs brilliantly on low cost microcontrollers.

The reference implementation given by NSA [7] is implemented using 64 bit unsigned integers in C. But due to the lack of 64 bit registers in the microcontroller we used, it was implemented by us for both 8 and 16 bit microcontrollers using small registers.

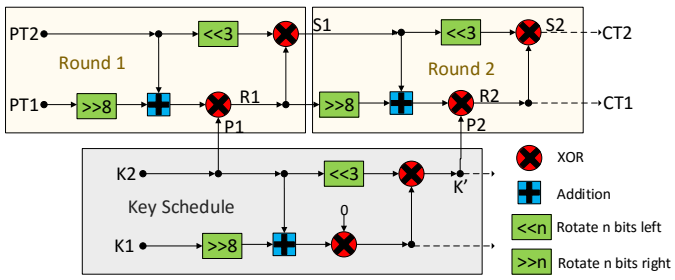


Fig. 5. Speck encryption algorithm

B. Attack Methodology

The key mixing in Speck happens in a completely different fashion than AES. Also the well known attack on AES uses the sbox lookup where no such operations are present in Speck. We have introduced methodologies that enable the successful derivation of the key in a Speck cryptosystem.

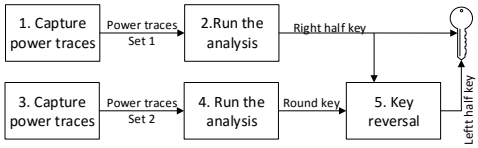


Fig. 6. Speck encryption algorithm

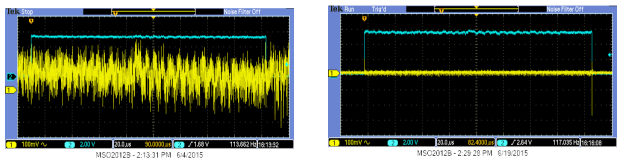
As shown in Fig. 5 mixing of the key happens in two phases. Right half of the key (K2) is directly mixed at P1 as shown there, but the left half key (K1) is further modified to create a round key (K') that is mixed at P2. Therefore unlike in AES, the attack on Speck has to go in two phases as shown in Fig. 6. In the first phase, power is collected to include power consumption during memory access of K2 and analysed to break K2. In the second phase, power is collected for K' and analysed to derive K'. If the oscilloscope has good resolution, power can be collected to include both K2 and K'. Then with the help of K2, K' is reversed back to derive K1. Finally K1 and K2 are concatenated to get the whole key.

Sbox lookup in AES consumes high amount of power. It also facilitate the concept called confusion [10] where the change in one bit of the key completely changes the output. These facts would make CPA easier on AES. As sbox lookups are not present in Speck, the effort needed for the attack increases. The attack methodology introduced by us for Speck, targets the xor operations marked as R1 and R2 in Fig. 5.

Using xor operation which is a very simple operation causes several issues. First due to the lack of enough confusion, the number of power traces required would increase. Also as xor is not a byte wise operation as sbox, attacking using xor on 16 bit or higher microcontrollers become a challenge. First we addressed the issue by changing the CPA algorithm to attack several bytes at a time. But it is not feasible for 32 bit or higher microcontroller due to time complexity. Therefore again going back to the byte wise attack, with attacks points changed to S1 and S2 in Fig. 5 instead of R1 and R2, a successful attack was realized. Further due to a property of the xor operation (P XOR 0 = P), the key can be always falsely returned as all

TABLE I. TIME TAKEN FOR AN ATTACK ON AN 8 BIT SPECK CRYPTOSYSTEM

Step	Time taken / s
Phase 1: Collecting power traces	913.52
Phase 1: Running CPA	28.97
Phase 2: Collecting power traces	907.63
Phase 2: Running CPA	28.63
Sum	**1878.75**



(a) (b)

Fig. 7. Power traces (a)without filter (b)with filter

zeros. This issue was solved by trimming the beginning part of the power traces that caused the unwanted correlation. In depth details of those approaches are not elaborated here as the space is restricted.

C. Results

The 8 bit testbed was reprogrammed to carry out the Speck algorithm. The attack which was carried out using 500 power traces became successful. Table I shows the time taken for the attack. As two separate phases are carried out and because more traces are required, time required is higher than AES, but yet it is less than one hour. We also tested on a 16 bit PIC microcontroller, where now the required traces grew up to about 5000. The time required was about 2.5 hours. Therefore though Speck is a new algorithm yet it is vulnerable to power analysis.

IV. COUNTERMEASURES AGAINST POWER ANALYSIS

A. Hardware Countermeasures

We selected the power line filter based approach proposed by Mangard et al. [1], as no one has practically tested it to our best of our knowledge. A low pass filter is a device that attenuates high frequency components of a signal. The spikes in the power lines when data is read or written, from or to the memory, are the culprits that leak secret information. These sharp peaks are usually formed of higher frequency components. Once a low pass filter is applied to the power line, those peaks are smoothed out, reducing the leakage. Fig. 7 shows how power traces are affected when a filter is used. There, the yellow waveform is the power trace while the blue rectangular pulse is the trigger used. When no filter is used, power traces have high peaks and large variations as shown in Fig. 7a. But when a filter is introduced traces become squeezed and flattened as shown in Fig. 7b.

We implemented several filters and connected them to our testbed. All filters were implemented using passive electronic components such as resistors, capacitors and inductors. The results of the attacks are summarized in Table II. When no filter is connected, for breaking AES even 50 traces are enough. But Table II shows that when filters are implemented, the required

TABLE II. RESULTS FOR DIFFERENT FILTERS

Method	Approximate number of traces	Approximate time
Without filters	50	5 minutes
Capacitor (1mF) connected in parallel	1500	1.5 hours
Inductor (1mH) connected serially	500	30 minutes
LC (Inductor-capacitor) second order filter	5000	4.5 hours

TABLE III. RESULTS FOR DIFFERENT COMPONENTS

Method	Approximate number of traces	Approximate time
Without countermeasures	50	5 minutes
Voltage regulator	50	5 minutes
Current source	50	5 minutes
Zener diode	300	20 minutes
Operational amplifier (UA741)	4000	3.5 hours

number of power traces are increased. But yet the best one out of the ones we tested can be broken in 4.5 hours. This is not a considerable time for an attacker. Therefore we infer that though filters increase the effort required for power analysis, it is not good enough to make an attack infeasible.

We tried some of our own ideas as well, to see whether they have any effectiveness as countermeasures. Results for those ideas are shown in Table III. Voltage regulator had no effect. Then a constant current source was used to provide power but yet it was not useful at all. A zener diode connected in parallel to the device increased the number of traces by a little. Then power was supplied through an operational amplifier and it had some what better results. But it could be broken in 3.5 hours. Therefore, circuit methods we tested so far are not good enough.

B. Software Countermeasures

Random instruction injection is a software based countermeasure introduced by Ambrose et al. [8]. The effectiveness has been tested by them only on a simulator and therefore we decided to test it practically on a real system. The CPA algorithm requires all the power traces to be aligned. That is, power consumption during a certain operation should always be at the same time position in all power traces. When some false instructions are randomly inserted in to the middle of the encryption algorithm, this alignment is broken. The more misaligned the power traces are, the more number of power traces are required for a successful attack.

We modified the AES implementation in section II, so that instructions are randomly inserted. Table IV shows the results of the attack for different number of random instructions injected. The first column is the maximum number of random instructions injected at runtime. The number of power traces required are quadratically increasing and we can predict that injecting about 100 random instructions would increase the attack time to more than 20 days. Therefore we infer that random instruction injection is effective when compared to the filter based countermeasures described in section IV-A.

The randomness determines the misalignment of the power traces and hence the security basically depends on that. Pseudo random algorithms used for generating random numbers in software always generate the same sequence for a same seed.

TABLE IV. RESULTS FOR RANDOM INSTRUCTION INJECTION

Maximum number of random instructions injected	Approximate number of traces	Approximate time
0	50	5 minutes
1	200	15 minutes
3	500	30 minutes
7	2000	2 hours
15	40000	45 hours

In a computer, time would be a good value for the seed. But unfortunately in microcontrollers, real time clocks are generally not there. Currently we are working on a solution where the seed is to be generated using amplified noise signals which are considered to be true random.

V. CONCLUSION

Building a testbed for power analysis attack is the first step in power analysis based research. It is a time consuming task that requires patient debugging especially when a proper step by step guide is lacking. With the testbed we built, AES could be broken in even lesser than 10 minutes time. Speck algorithm has several differences that makes it impossible to use the power analysis approach for AES. But using novel approaches speck could be broken in less than an hour. Though the effort taken to break Speck is larger than AES, still it is very vulnerable. Therefore even for a new algorithm, countermeasures are required. Circuit based countermeasures such as adding power filters and other electronic components were not much effective. But on the other hand software based countermeasures such as random instruction injection proved to be effective.

REFERENCES

[1] S. Mangard et al., *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in CryptologyCRYPTO99*. Springer, 1999, pp. 388–397.

[3] E. Brier *et al.*, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems-CHES 2004*. Springer, 2004, pp. 16–29.

[4] F. Tepanek *et al.*, "Differential power analysis under constrained budget: Low cost education of hackers," in *2013 Euromicro Conference on Digital System Design (DSD)*. IEEE, 2013, pp. 645–648.

[5] Z. Martinasek *et al.*, "General scheme of differential power analysis," in *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2013, pp. 358–362.

[6] M. Petrvalsky *et al.*, "Differential power analysis of advanced encryption standard on accelerated 8051 processor," in *Radioelektronika (RADIOELEKTRONIKA), 2013 23rd International Conference*. IEEE, 2013, pp. 334–339.

[7] R. Beaulieu *et al.*, "The simon and speck families of lightweight block ciphers," 2013.

[8] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "Rijid: random code injection to mask power analysis based side channel attacks," in *Proceedings of the 44th annual Design Automation Conference*. ACM, 2007, pp. 489–492.

[9] H. Gamaarachchi. (2015, Aug.) Poweranalysis. [Online]. Available: https://github.com/hasindu2008/PowerAnalysis

[10] B. A. Forouzan, *Cryptography and Network Security*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2008.