Investigating Multi-Thread Utilization as a Software Defence Mechanism Against Side Channel Attacks

Ibraheem Frieslaar *
Council for Scientific and Industrial Research
Modelling and Digital Sciences
Pretoria, South Africa
ifrieslaar@csir.co.za

Barry Irwin
Rhodes University
Computer Science Department
Grahamstown, South Africa
b.irwin@ru.ac.za

ABSTRACT

A state-of-the-art software countermeasure to defend against side channel attacks is investigated in this work. The implementation of this novel approach consists of using multi-threads and a task scheduler on a microcontroller to purposefully leak out information at critical points in the cryptographic algorithm and confuse the attacker. This research demonstrates it is capable of outperforming the known countermeasure of hiding and shuffling in terms of preventing the secret information from being leaked out. Furthermore, the proposed countermeasure mitigates the side channel attacks, such as correlation power analysis and template attacks.

CCS Concepts

•Security and privacy → Side-channel analysis and countermeasures;

Keywords

Software Countermeasure, Side Channel Attacks, Multi-Threading, Electromagnetic

1. INTRODUCTION

It has been many years since Kocher *et al.* [11] introduced side channel analysis (SCA) as a means to obtain secret information from an embedded device. This method is still in widespread use today [18, 20]. Different cryptographic algorithms have been attacked over the years, such as DES [14], RSA [7], and AES [17].

The need to provide individuals with a sense of security assurance has become prominent. Therefore, many applications offer a form of cryptographic security to ensure their data is confidential and secured. Although the cryptographic algorithms are secured, the implementation of these algorithms on an embedded device could be vulnerable to SCA attacks. SCA attacks exploits the power consumption or the electromagnetic emissions [8] to retrieve secret information by finding a correlation between the intermediate values and the power consumption [11].

Since the National Institute of Standards and Technology (NIST) has declared AES the standard protocol to encrypt information, the research community has focused extensively on attacking AES implementations on various devices [3, 12, 17, 19].

The design of a software countermeasure is extremely challenging due to microcontroller hardware leaking out information as opposed to the software countermeasure leaking out sensitive information. Furthermore, the average software developer does not possess all the knowledge about an embedded device and this could lead to unintentional side channel leakage.

This work takes advantage of the fact that most commercial microcontrollers leak out information during normal operations, and turns it into a strength by using multiple threads and a task scheduler to purposely leak out obfuscated information at critical points in the AES-128 algorithm to defend against a side channel attacks. The task scheduler would control the number of threads to generate these leakages and enable the microcontroller to have a different power trace on each execution of the cryptographic algorithm. Furthermore, this research is compared to an existing software countermeasure as the success rates of both countermeasures are compared against two different types of side channel attacks.

The remainder of this paper is organized as follows: the research carried out in the area of developing software countermeasures against SCA will be discussed in Section 2; Section 3 will define the proposed countermeasure; the techniques and equipment used in this research will be elaborated in Section 4; followed by the results and analysis in Sections 5; finally, the paper is concluded in Section 6.

2. SOFTWARE COUNTERMEASURES

The most prominent existing software countermeasure techniques will be discussed in this section. A few of these techniques are known as random precharging, masking, hiding and shuffling. Random precharging can be carried out at a software level by flooding the datapath with a random operand instruction before and after an important value is used [21].

A well known approach to defend against SCA is masking [5]. This approach involves removing the correlation between the intermediate values and the power consumption of the device. To achieve masking, many masking tech-

*Studies at Rhodes University through a CSIR studentship.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

niques are available such as: Boolean [5]; additive [9]; and multiplicative [10] masking. Boolean masking uses an XOR operation between the sensitive variables (a) and random numbers (r) to generate a variable mask $a_m = a \oplus r$. The original value is returned by applying the same XOR on that mask. The additive and multiplicative masks use the equations ($a_m = a + r \mod n$) and ($a_m = a * r \mod n$), respectively, to mask the secret information.

The hiding countermeasure aims to reduce the correlation between the intermediate variables and its power consumption. This is achieved by adding dummy instructions at critical locations in the algorithm, making use of a variable clock frequency or having random delays in the algorithm. Furthermore, shuffling of the sensitive values can be used as a hiding countermeasure. These methods can be combined with each other to from a resistance to SCA attacks. The resistance is calculated by the number of possible occurrences the subkey can occur at, in a specific location in time on the power trace. An example of combining methods is the insertion of dummy instruction and shuffling of the AES subkeys. An example of combining methods are the insertion of dummy instruction and shuffling of the AES subkeys. This research will compare the implementation of the hiding and shuffling technique against the proposed countermeasure.

3. PROPOSED COUNTERMEASURE

The proposed countermeasure will be discussed in this section. It is intended to improve on the basic hiding and shuffling countermeasure by introducing multi-threads and a task scheduler as a new software countermeasure. Furthermore, this novel approach will aim to incorporate the basic countermeasure into its design.

This research implemented the AVR Thread Library [6] as the multi-thread framework. This library was chosen since it was designed to work with Atmel AVR family of microcontrollers. The library consists of having basic preemptive multitasking and implements a simple round-robin style task switcher. It was originally designed to work with the previous generation of Atmel microcontrollers. Therefore, it has been modified and recompiled to work for the Atmega328p MCU and current generation of Atmel AVR microcontrollers. It is noted that the implementation of multiple threads as a countermeasure has the capabilities of being used on other microcontrollers that has dedicated support for multi-thread operations.

The design of the system would allow the AES-128 algorithm to be executed on one thread. This will be known as the encryption thread. While the encryption thread is processing, many other threads would start to execute. These other threads are known as noise threads. These noise threads comprise of dummy mathematical instructions. Based on the related work these noise threads would be executed at the S-boxes. Furthermore, on each execution of the code, the noise threads would randomly change based on the dynamic power consumption produced by the task scheduler.

In order to determine the number of threads to be used per execution, the system would randomly select a number (x) between 0 – 15. This x value would be used to generate noise threads at random subkeys $y[x]$. Each $y[\,]$ array value had been randomised to ensure on each execution that the noise threads would occur at different locations. It is noted that a checking method is in place to prevent the $y[\,]$ array values from being the same. Therefore, a dynamic power trace would be generated per execution.

Various types of noise threads are used to increase the resistance to an attack. A noise thread has the ability of executing a different set of mathematical instructions. An example of such a noise thread is the implementation of greatest common divisor (GCD). It is possible to place more than one noise thread at a subkey. Therefore, if multiple noise threads were placed at a subkey the system would be capable of hampering attackers that make use of low cost equipment.

Before the countermeasure was placed onto the MCU a value (y) was stored into the EEPROM. Once the countermeasure was in place, it would call this value from memory and use it to change the random seed value on each execution of the algorithm. Upon completion of the S-box procedure, y would be incremented, XOR'D and the value would get stored back into memory to be used for the next execution. This generates more confusion since y is XOR'D and gets passed through the same S-box as the AES-128 algorithm. The next procedure is to combine the shuffling technique into the new system

4. EXPERIMENT SETUP

This section has been divided into three subsections as follows. The equipment used in this research will be discussed in Subsection 4.1, Subsection 4.2 will explain side channel attacks and Subsection 4.3 will detail the experimental setup and the experiments carried out.

4.1 Equipment

In order to capture data from a microcontroller the research makes use of the ChipWhisperer kit [16]. This kit is well suited to carry out SCA attacks. The kit consists of four main items which is a target device, measuring equipment, capturing software, and attack software. The ChipWhisperer uses a field-programmable gate array (FPGA). The FPGA is the ZTEX FPGA Module which uses a Spartan 6 LX25 FPGA [22]. Using the ChipWhisperer the attacker is able to use synchronous sampling. It has been shown that synchronising the clock at 96 MS/s would give the same results as using an asynchronous sampling rate of 2 GS/s [15]. Furthermore, using a synchronised clock provides a very repeatable power consumption as seen in Figure 1. The figure illustrates that all traces are very similar to each other, due to synchronously captured power traces.

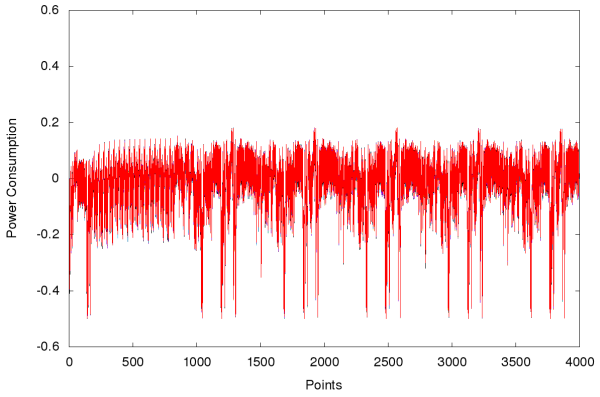

Figure 1: 10 traces captured using a synchronised clock.

The ChipWhisperer Lite had been used to capture data from the Atmel ATmega328p MCU. The MCU forms part the Atmel 8-bit microprocessors family. They offer various functionality such as on-chip Flash, SRAM and internal EEPROM memories [2]. It has been shown that Atmel's MCUs have been used in various scientific research and in industrial applications [13].

The capturing software allows for the preprocessing of the power traces. The software makes provision for sampling the device at four times its original clock speed, this improves the interpretation of the power consumption. The variables that remained consistent in the experiments are as follows: a low gain setting was used with the digital signal amplified by 34.5039 decibel (dB); the rising edge logic level is used to trigger the capture of power traces; and finally, the clock frequency was multiplied to rise from 7.375 MHz to a frequency of 29.5 MHz. These set variables assisted in increasing the power output. Furthermore, all measurements were captured synchronously.

Figure 2 depicts a comparison between preprocessing and trace processing of traces. From the figure it is seen that the bottom power trace is more defined than that of the power trace above with no processing. Additionally, Figure 3 illustrates a comparison between capturing at 7.38 MHz and 29.5 MHz. It is observed that at 29.5 MHz the power consumption is more stable than at 7.38 MHz. This is further evident when observing from points 3000 onwards.

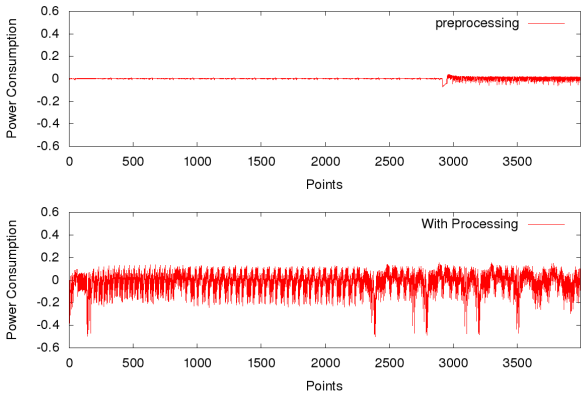

Figure 2: Comparison between capturing with no processing and with processing.

4.2 Attack Procedure

This research makes use of the coreelation power analysis (CPA) and template attack to retrieve the secret keys of the cryptographic implementation of AES-128 on a microcontroller. It has been well documented that the Differential Power Analysis (DPA) is outperformed by the CPA [3]. The DPA technique requires thousands of power traces to retrieve the secret key whereas the CPA approach only requires a few traces. In this research it was shown that only 1550 traces were needed to retrieve the correct secret key using the CPA method. Therefore, in terms of speed the CPA approach is much faster and more accurate since it looks at the correlation between all the key guesses.

Template attacks are probabilistic side channel attacks. It makes use of a Gaussian noise model and the maximum likelihood principle to reveal the secret information from the

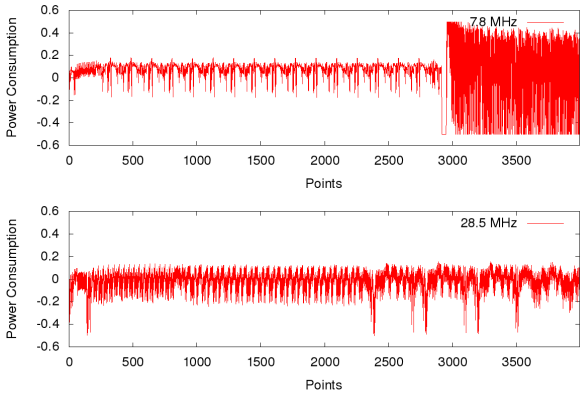

Figure 3: Comparison between capturing at 7.38 MHz and 29.5 MHz.

leakages [1]. This attack is used since it is able to obtain secret information from countermeasures whose security is dependent on the assumption that an attacker is unable to obtain more than one or a limited amount of samples [4].

4.3 Experiments

This subsection elaborates on the experimental setup used in this research. The setup procedure to recover the secret key of the cryptographic algorithm comprises of two stages. These two stages are the capture data and analysis data phase.

The capture data phase comprised of using the embedded MCU. The PCB board which contains the Atmega328p MCU, is referred to the device under test. The ChipWhisperer was connected to the device under test. In order to capture Electromagnetic (EM) emanations the ChipWhisperer was connected to a low noise amplifier which intern was connected to an EM probe. The probe was placed over the Atmega328p processor. Figure 4 illustrates the hardware setup.

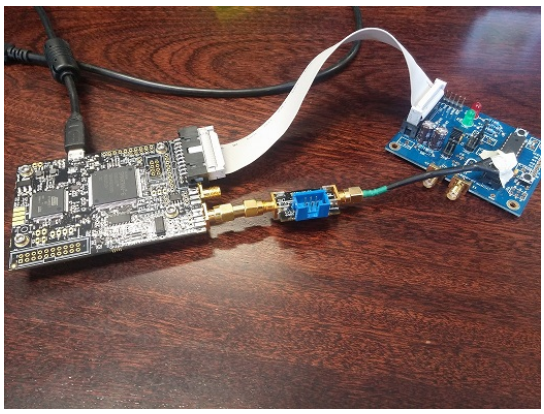

Figure 4: the hardware setup: On the left, the ChipWhisperer connected to the PCB containing the Atmega328p processor; A low noise amplifier is connected to an EM probe placed on the processor.

The first set of experiments was to determine the breaking point of an unprotected implementation of the AES-128 algorithm. The AES-128 program was flashed onto the At-

mega328p. While the Atmega328p was executing the crypto code, EM emanations was captured via the EM probe and ChipWhisperer. Upon acquiring the EM data, the CPA attack was carried out against it.

The second experiments involved making use of the template attack. The attack consisted of a two step data collection setup. Firstly, the secret key and text used as input are set to random on each occasion the Atmega328p executed the crypto code. This random set captured was used as the training set to generate a template. The second step was to capture the EM emanations again. However, on this occasion the secret key was set to a fixed value. This secondary data required a few input traces and was the testing set.

Experiments three and four consisted of implementing the countermeasures into the AES-128 algorithm. The hiding and shuffling, and the threaded countermeasure was used. Both implementations was tested against the CPA and template attacks.

5. RESULTS AND ANALYSIS

The results of the experiments explained in Subsection 4.3 are examined in this section. The first experiment is analysed in order to achieve a baseline for CPA attacks. Observing Table 1, it consists of three columns. The first column indicates the number of traces used, followed be the number of secret keys recovered, and finally, the third column represents the success rate. The success rate is calculated by the total number of subkeys (16) divided by the number of recovered keys. The table indicates that it is possible to recover the entire secret key as EM emanations is used as input data. Furthermore, only 1550 traces is required to carry out a successful CPA attack with EM data.

Table 1: CPA results against an unprotected implementation of AES-128.

Traces	Recovered Keys	Success Rate
410	9	0.5625
450	10	0.625
480	12	0.75
1050	14	0.875
1550	16	1

Table 2 depicts the results of the template attack against an unprotected AES-128 implementation while the EM emanations are collected as input data. The table setup is the same as Table 1 and the success rate is calculated in the same manner. From the table it is seen that the entire secret key is able to be recovered. Only 383 input samples is required when 50 000 samples is used as training samples.

Table 2: The results of a template attack using 50 000 traces as training input

Traces	Recovered Keys	Success Rate
88	9	0.5625
100	10	0.625
123	12	0.75
286	14	0.875
383	16	1

Experiment three results is examined in Figure 5. The figure displays a comparison between the results of the CPA attacks as the unprotected implementation, known countermeasure, and proposed countermeasure is used. It is seen that as the hiding and shuffling countermeasure is in place the entire secret is recoverable. However, 63 450 traces is needed. Furthermore, a success rate below 0.2 is achieved as the proposed countermeasure is in place. This relates to only three subkeys being recoverable. Additionally, almost 80 000 traces is required to only recover three subkeys.

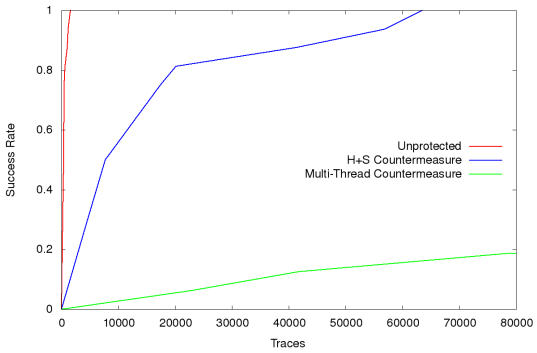

Figure 5: A comparison between the success rate of the CPA attack against the different implementations.

Experiment four results is depicted in Table 6. The table illustrates the success rate as the template attack is carried out against the AES-128 implementation with the various implementations in place. Both countermeasures prevents the recovery of the entire secret key. Although, both countermeasures is successful, the multi-thread countermeasure success rate for template attacks is less than that of the hiding and shuffling countermeasure. Therefore, the multi-thread countermeasure outperforms the hiding and shuffling countermeasure in terms of preventing more information from being leaked.

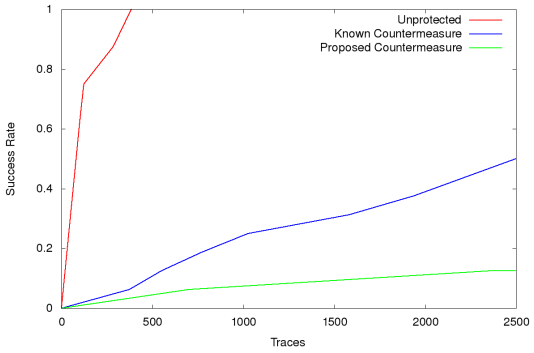

Figure 6: The success rate of the template attack with 50 000 training samples against the various implementations.

The research is able to successfully obtain the secret key on an unprotected device using less than 1600 power traces as input for a CPA attack and is further able to successfully recover the secret key using a template based attack.

The proposed countermeasure demonstrated the ability to prevent the CPA attack from predicting the correct key as oppose to the known countermeasure of hiding and shuffling which fails to prevent critical information being leaked out as the CPA attack is used. Furthermore, the proposed

countermeasure is able to defend well against the template attacks. It is shown that the template attack achieved a success rate below 0.2. The implementation of a task scheduler and multi-threads allows for the creation of dynamic power traces and these dynamic power traces cause both side channel attacks to predict the incorrect secret key.

6. CONCLUSION AND ONGOING RESEARCH

In this research a state-of-the-art software countermeasure was introduced to mitigate side channel attacks. The implementation consists of using multi-threads and a task scheduler as a novel approach to defend against attacks. These multi-threads are comprised of various mathematical operations to generate noise at various points in time in the AES-128 algorithm. This approach introduces dynamic power traces, where the power traces are changed based on the task schedulers conditions. It is demonstrated that this novel countermeasure outperforms an existing countermeasure of hiding and shuffling against two different types of attacks.

This work sets the basis for the forthcoming research where this approach would be implemented and tested on a true multi-threading platforms, such as smartphones and laptops. Moreover, the research aspires to implement a countermeasure capable of defending embedded hardware and high-powered devices against known and future side channel attacks.

Acknowledgement

This work was undertaken as part of the Distributed Multimedia CoE at Rhodes University, with financial support from the department of Modelling and Digital Science at CSIR, Telkom SA, Tellabs/CORIANT, Easttel, Bright Ideas 39, THRIP and NRF SA (UID 90243). The authors acknowledge that opinions, findings and conclusions or recommendations expressed here are those of the author(s) and that none of the above mentioned sponsors accept liability whatsoever in this regard.

7. REFERENCES

[1] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Template attacks in principal subspaces. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–14. Springer, 2006.

[2] Atmel. ATxmega128D4., Nov. 2015.

[3] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 16–29. Springer, 2004.

[4] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[5] J.-S. Coron and L. Goubin. On boolean and arithmetic masking against differential power analysis. In *Cryptographic Hardware and Embedded SystemsŮCHES 2000*, pages 231–237. Springer, 2000.

[6] D. Ferreyra. AVR development., Nov. 2008.

[7] T. Finke, M. Gebhardt, and W. Schindler. A new side-channel attack on RSA prime generation. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 141–155. Springer, 2009.

[8] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded SystemsŮCHES 2001*, pages 251–261. Springer, 2001.

[9] L. Genelle, E. Prouff, and M. Quisquater. Thwarting higher-order side channel analysis with additive and multiplicative maskings. In *Cryptographic Hardware and Embedded Systems–CHES 2011*, pages 240–255. Springer, 2011.

[10] J. D. Golić and C. Tymen. Multiplicative masking and power analysis of AES. In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pages 198–212. Springer, 2002.

[11] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in CryptologyŮCRYPTOŠ99*, pages 388–397. Springer, 1999.

[12] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, 2011.

[13] W. Kunikowski, E. Czerwiński, P. Olejnik, and J. Awrejcewicz. An overview of ATmega AVR microcontrollers used in scientific research and industrial applications. *Pomiary, Automatyka, Robotyka*, 19, 2015.

[14] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[15] C. O'Flynn and Z. Chen. A case study of side-channel analysis using decoupling capacitor power measurement with the OpenADC. In *Foundations and Practice of Security*, pages 341–356. Springer, 2012.

[16] C. O'Flynn and Z. D. Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In *Constructive Side-Channel Analysis and Secure Design*, pages 243–260. Springer, 2014.

[17] C. O'Flynn and Z. D. Chen. Side channel power analysis of an AES-256 bootloader. In *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on*, pages 750–755. IEEE, 2015.

[18] P. Pessl and S. Mangard. Enhancing side-channel analysis of binary-field multiplication with bit reliability. In *Topics in Cryptology-CT-RSA 2016*, pages 255–270. Springer, 2016.

[19] K. Schramm, G. Leander, P. Felke, and C. Paar. A collision-attack on AES. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 163–175. Springer, 2004.

[20] H. Seuschek, J. Heyszl, and F. De Santis. A cautionary note: Side-channel leakage implications of deterministic signature schemes. In *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems*, pages 7–12. ACM, 2016.

[21] S. Tillich and J. Großschädl. *Cryptographic Hardware and Embedded Systems - CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings*, chapter Power Analysis Resistant AES Implementation with Instruction Set Extensions, pages 303–319. 2007.

[22] ZTEX. Spartan 6 LX9 to LX25 FPGA Board., Nov. 2016.