# Evaluating The Multi-Threading Countermeasure

Ibraheem Frieslaar [#*1], Barry Irwin [*2]

[#] *Modelling and Digital Science Council for Scientific and Industrial Research*
*Pretoria, South Africa*
[1] ifrieslaar@csir.co.za

[*] *Department of Computer Science, Rhodes University,*
*Grahamstown, South Africa*
[2] b.irwin@ru.ac.za

***Abstract*—This research investigates the resistance of the multi-threaded countermeasure to side channel analysis (SCA) attacks. The multi-threaded countermeasure is attacked using the Correlation Power Analysis (CPA) and template attacks. Additionally, it is compared to the existing hiding countermeasure. Furthermore, additional signal processing techniques are used to increase the attack success ratio. It is demonstrated that the multi-threaded countermeasure is able to outperform the existing countermeasures by withstanding the CPA and template attacks. Furthermore, the multi-threaded countermeasure is unaffected by the elastic alignment and filtering techniques as opposed to the existing countermeasures. The research concludes that the multi-threaded countermeasure is indeed a secure implementation to mitigate SCA attacks.**

***Index Terms*—Multi-Threading, CPA, Template Attack, Elastic Alignment.**

## I. Introduction

We are currently in a society where the need to secure and protect our information has filtered down into our daily lives such as making transactions with our mobile devices. Hence, individuals wants to be reassured that their credentials would not be stolen as transactions are made. Therefore, numerous applications enforce the use of cryptographic algorithms to obfuscate the individuals information from people attempting to intercept data. One of these cryptographic algorithms is the AES algorithm [1]. This algorithm has been declared to be the standard protocol to encrypt information by the The National Institute of Standards and Technology [2]. Although cryptographic algorithms are mathematically secured, various research has demonstrated to use side channel analysis (SCA) attacks to recover the secret information [3], [4].

The attacker uses SCA attacks to locate a correlation between the intermediate values and the power consumption by using the captured data from a device. This captured data can be the power consumption or the electromagnetic emissions as the device is executing the cryptographic algorithm [5], [6].

The objective of this research is to perform a rigorous assessment of the multi-threaded countermeasure [7]. They proposed a software solution by using multi-threads to mitigate SCA attacks. The authors performed a Correlation Power Analysis (CPA) attack on their implementation with minimal samples. Therefore, this research performs both the CPA and template attacks. Furthermore, the research applies additional processing techniques such as the elastic alignment and low pass filtering algorithms to enhance the attacks. Additionally, this research uses over 100 000 traces to perform the SCA attacks.

The remainder of this work is organised as follows: the side channel analysis attacks of the CPA and template techniques and procedure is detailed in Sections II and III; the hiding countermeasure is explained in Section IV; the methodology of this research is elaborated in Section V; followed by the experiments, results and analysis in Section VI; finally the work is concluded in Section VII.

## II. Correlation Power Analysis

This section explains the Correlation Power Analysis (CPA) attack. The CPA attack is used 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 [8]. 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 50 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.

For a detailed explanation of the mathematical approach of the correlation equation the reader is referred to [8]. The correlation equation is needed to determine the correlation between all the key guesses. The correlation equation is as follows.

$$r_{i,j} = \frac{\sum_{d=1}^{D}[(h_{d,i} - \bar{h_i})(t_{d,j} - \bar{t_j})]}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \bar{h_i})^2 \sum_{d=1}^{D}(t_{d,j} - \bar{t_j})^2}} \quad (1)$$

Where $t_{d,j}$ is the captured power trace, with the total number of traces $D$ and $h_{d,i}$ is the hypothetical values produced by the power leakage model. The power leakage model that was used is the Hamming weight power model [9].

In order to obtain the secret key from the AES-128 algorithm, four steps were followed: While the AES-128 algorithm was executing the encryption process, the power traces along with its corresponding input text were captured; a power leakage model was implemented where the guess of a key byte is used with a known input text; the correlation equation,

Equation 1 was implemented to run through all the captured power traces; and finally, a ranking procedure was created that predicted the most likely key based on the correlation accuracy.

The attack on the AES-128 cryptographic algorithm takes place at the point the secret key is sent to the lookup table (S-Box). Upon acquiring the power traces (data), a power leakage model is implemented with a guessing procedure. The system loops through one subkey at a time and guesses every possible outcome for that subkey. These guess values range from 0 – 255. The next phase is to calculate the corresponding intermediate value of that guess. The value of each guess is converted to its binary representation of the value, with the total number of 1's summed up to determine the *weight*. This is referred to the hamming weight power model [9]. It is known that the encryption algorithm has 16 subkeys [10] and since each subkey is attacked one at a time there is only $2^{12}$ possibilities, instead of $2^{128}$ possibilities to predict the correct secret key.

The values are substituted in Equation 1 to obtain the correlation. A negative correlation is possible. However, the absolute value is taken. Furthermore, to achieve a ranking system the correlation of each guess is stored and the guess with the maximum correlation is predicted to be the subkey.

## III. Template Attacks

The template attack is a powerful probabilistic side channel analysis attack. To reveal secret information the template attack incorporates a Gaussian noise model and the maximum likelihood principal [11]. This subsection will be divided into a further three subsections consisting of multivariant statistics, template creation, data analysis and applying the attack.

### A. Multivariant Statistics

It is well known that electrical signals are noisy. Upon measuring voltage the user does not expect to capture the perfect reading. However, it is possible to model the voltage source. If we have the equation $X = X_{true} + N$, where $X_{true}$ is the perfect signal and $N$ is the noise. The probability density function (PDF) of a Gaussian distribution is used to calculate how likely a certain measurement is [12]. The PDF equation is as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{2}$$

where, $\mu$ is the mean and the standard deviation is $\sigma$. Therefore if $f(x)$ is an extremely low value, it can be comfortably assumed that it would not be the correct subkey.

The PDF works well for one variable. However, the attacker requires to use multiple variables in order to recover the 16 subkeys. Multivariate distributions allows for the modelling of multiple variables that may or may not be correlated. Instead of determining one $\sigma$, the system would determine a matrix of covariances. In order to model three variables $X_1, X_2, X_3$, the matrix would be as follows:

$$\Sigma = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) & Cov(X_1, X_3) \\ Cov(X_2, X_1) & Var(X_2) & Cov(X_2, X_3) \\ Cov(X_3, X_1) & Cov(X_3, X_2) & Var(X_3) \end{bmatrix} \tag{3}$$

Furthermore, each value mean has to be calculated, therefore

$$\mu = \begin{bmatrix} \mu X_1 \\ \mu X_2 \\ \mu X_3 \end{bmatrix} \tag{4}$$

To calculate the PDF of a multi-variance a vector $(v)$ is required, where $v = [X_1, X_2, X_3 \dots X_n]^T$ Thus the equation for $N$ variables is:

$$f(x) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} e^{-\frac{(v-\mu)^T \sum^{-1} (v-\mu)}{2}} \tag{5}$$

Therefore if $N$ points are taken from the power trace and substituted into $v$ from Equation 5. with $f(x)$ returning as a high value. The system would interpret this as a good guess for the potential subkey value.

### B. Creating The Template

A template is effectively a set of probability distributions that portrays what the power traces looks like for various random keys. Therefore, if key $(k)$ is used, the power trace of that key would be the distribution $f_k(x)$. In order to create a good distribution to model the power trace for every key, thousands or even millions of traces are required as the attacker would require to create 256 models. However, there is an alternative approach. The attacker can model a sensitive section in the AES-128 algorithm. By focusing on the substitution box in the algorithm the attacker can create a model of every possible hamming weight, thus only 9 models are required. This reduces the order of magnitude drastically.

The attacker goes on the assumption that the selection of key does not affect the entire power trace. It is considered that the subkeys only influence the power consumption at a few critical times. If the attacker is able to determine these points of interest he would be able to create to only need three points to create a 3D distribution model.

The attacker would be able to determine these points of interest by using a simple method know as the sum of differences method. The algorithm is as follows: For every sample $i$ with every operation $k$, determine the average power consumption $M_{k,i}$ . Hence, if there are $T_k$ traces where the attacker performed $K$ operations, the average would be calculated as follows:

$$M_{k,i} = \frac{1}{T_k \sum_{j=1}^{T_k} t_{j,i}} \tag{6}$$

Upon acquiring the set of averages, the absolute pairwise differences is calculated, followed by the summation of these values. This will generate a power trace which consists of various peaks. The next procedure would be to select the highest peak, remove minimal peaks close to the highest peak and traverse through the power trace until enough points of interest have been selected.

The attacker's aim would be to determine the mean and covariance matrix for every operation. In order to achieve this he would set out to do the following:

1) Locate every power trace that falls under the category of certain operation. Example: locate every power trace where the subkey was 0xFF.
2) Determine the mean at every point of interest.
3) Determine the variance at each point of interest.
4) Determine the covariance of every pair of points of interest.
5) Repeat for all operations.

Once the procedure listed above is completed, the attacker will have $K$ mean and covariance matrices, modelling each of the $K$ different operations that the target can do and he would be ready to attack carry out the attack.

### C. Applying the Template

For the attack a small number of traces from the victim's device is required, a minimum of 1 trace and a maximum of 20 traces are required. Upon acquiring these traces the PDF is calculated for every key guess based on the location of the points of interest from the template. In order to carry out a successful template attack, it is assumed that the attacker has access to a copy of the device he intends to attack. Once the device is obtained, four steps are required to the perform the attack.

1) Capture a large data set from the copy device using random plaintext and crypto keys as input. This could be between a hundred thousand and a million traces.
2) Generate a template based on the points of interest which consist of multivariate distribution of the traces.
3) Capture the traces from the victim's device.
4) Apply the template on the captured trace from the victim's device.

## IV. Hiding

The hiding countermeasure aims to reduce the correlation between the intermediate variables and its power consumption from a cryptographic device. This can be achieved by two methods. The first method obscures the correlation by varying the power consumption in the amplitude domain and the second method is to randomise instructions in the time domain [13].

Variations in the amplitude domain can be purposely generated by changing the clock rate of the device on each execution of a program. Shifting the amplitude can lead to a lower signal-to-noise ratio (SNR) which increases the difficulty of a side channel attack. Furthermore, embedded devices often integrate noise generators to add to the difficulty of performing a successful attack.

Hiding in the time domain can occur when traces are misaligned intentionally or unintentionally. Unintentional alignment arise due to lack of a satisfactory trigger signal. Intentional hiding in the time domain is attributed to hiding through randomization. This can be achieved by inserting dummy operations at critical sections in the algorithm or having random delays.

A second approach is to shuffle the sequence of operations. In terms of the AES crypto algorithm operations such as AddRoundKey, SubBytes, ShiftRows, or MixColumns can be shuffled. This research will compare the implementation of the combination of the hiding countermeasures against the multi-threaded countermeasure [14].

## V. Methodology

This research implements the proposed software countermeasure using multiple threads as a countermeasure to mitigate side channel analysis attacks [7]. It uses multi-threads and a task scheduler combined with the hiding countermeasure into its threaded design.

The system is designed to execute the AES-128 cryptographic algorithm on one thread. During this execution, multiple other threads, henceforth referred to as noise threads, are executed in parallel with the encryption thread. These noise threads contains dummy mathematical instructions such as calculating the Fibonacci sequence or determining the lowest common denominator. Additionally, the noise and crypto threads would randomly occur at different locations on each execution of the algorithm.

The countermeasure was designed to insert the noise threads at the subkey round. On each execution the noise thread would be inserted at a different calculation of a subkey. Furthermore, the system reviews previous output and based on data it would create a noise thread at a different location on the next execution cycle. The system is further randomised by altering the number of noise threads and the type of noise to be produced on each occasion. Since the countermeasure is able to insert multiple noise threads at any given subkey the the system would be capable of inhibit attackers that make use of low cost equipment.

The countermeasure is further reinforced by storing and updating the random seed in the EEPROM. This is to prevent the randomiser from generating the same sequence of values. Furthermore, the sequences in the AES-128 algorithm are shuffled as describe in the hiding technique.

This section is further categorised into two subsections. Subsection V-A covers the equipment used in the research and Subsection V-B discusses the attack strategy to obtain the secret key.

### A. Equipment

The research adopts the same ChipWhisperer equipment described in [7]. Furthermore, the same target device will be used. The target device is the Atmel ATmega328p microcontroller. The ChipWhisperer makes use of a field-programmable gate array (FPGA). The FPGA is the ZTEX FPGA Module which uses a Spartan 6 LX25 FPGA [15]. The FPGA allows for capturing of samples synchronously. It has been demonstrated that capturing synchronously with a low clock would yield the same results while capturing with a high asynchronous clock [16]. The use of synchronisation allows the attacker to capture the power signature in a repeatable manner. Fig 1 exhibits that the power consumption of multiple traces are similar to each other since a synchronous clock is used.

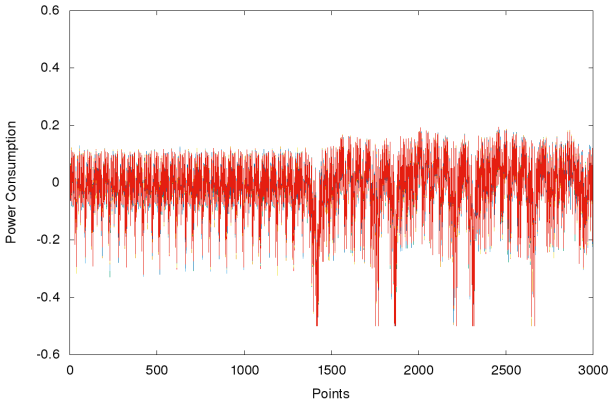


Fig. 1. 10 traces captured synchronously.

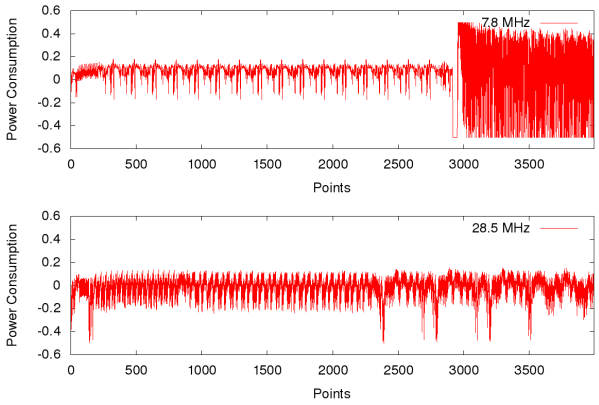


Fig. 3. A comparison in signal when the clock is multiplied by four.

The FPGA contains a clock module which routes the desired clock to the analog-to-digital converter (ADC) which in turns is capable of amplifying the clock frequency. The capture configuration is as follows: The digital signal is amplified by 34.5039 decibel (dB) with a low gain setting; the rising edge trigger logic is used to prompt the capturing of the power consumption; followed by the clock frequency which is multiplied by four, this multiples the base frequency 7.375 MHz to 29.5 MHz. Additionally, all samples is captured synchronously.

Fig 2 depicts a comparison between capturing with and without digital amplification. In Fig 3(b) it is seen that the signal is more defined than that of Fig 3(a). Additionally, Fig 3 depicts the comparison of capturing at a clock rate of 7.375 MHz and 29.5 MHz. While observing the points after 3000 it is evident that capturing with a high clock is more desirable.

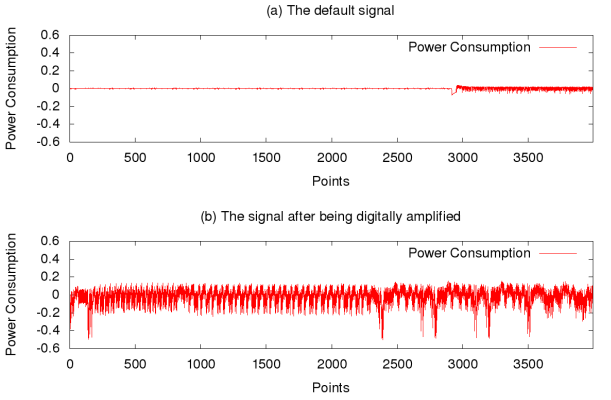


Fig. 2. A comparison between the signal being digitally amplified.

## B. Attack Procedure

To retrieve the secret key from the power consumption, this research will use the CPA and template attacks. The procedure for acquiring the secret information has been discussed in Sections II and III. Furthermore, the research will carry out additional signal processing techniques that was not applied in [7]. The elastic alignment technique as well the Savitzky?Golay (SG) digital filter will be used to increase the success ratio for retrieving the secret key. A data smoothing technique based on local least-squares polynomial approximation was proposed by Savitzky and Golay . Their research demonstrated that the smoothing reduces noise while preserving the peaks in the waveform [17].

The elastic alignment algorithm uses FastDTW to align power traces. Dynamic time warping (DTW) is a technique used in speech recognition to align record speech patterns. The dilemma speech recognition researchers were faced with was when two similar utterances with differences in timing needed to be aligned. However, Sakoe *et al.* introduced a solution by using dynamic programming to match these utterances using a nonlinear time path [18]. The elastic alignment algorithm first computed the DTW of the samples. Upon acquiring this information the algorithm aligns the two signals. Research has shown that using this algorithm significantly increases the success rate of a side channel attack. For an in-depth discussion on DTW and the elastic alignment algorithm the reader is referred to [19]. Figure 4 depicts the flow diagram of the experimental setup. It depicts that there are three major areas. The capture data, post processing and the analyse data phases. The capture data phase is the first phase where the data is captured from the device under test. The device under test is referred to the Atmega328p microcontroller which is placed onto a printed circuit board (PCB) and the device used to capture the data is the ChipWhisperer. Figure 5 depicts the hardware configuration as the ChipWhisperer is connected to the PCB via a serial cable, a low noise amplifier is connected to the ChipWhisperer which intern is connected to an electromagnetic probe which is used to capture the electromagnetic data from the Atmega328p microcontroller.

The second phase known as the post processing stage will take the captured data and apply the elastic alignment algorithm followed by the Savitzky?Golay digital filter. The process is concluded in stage three where the data will be attacked by the CPA and template attacks.

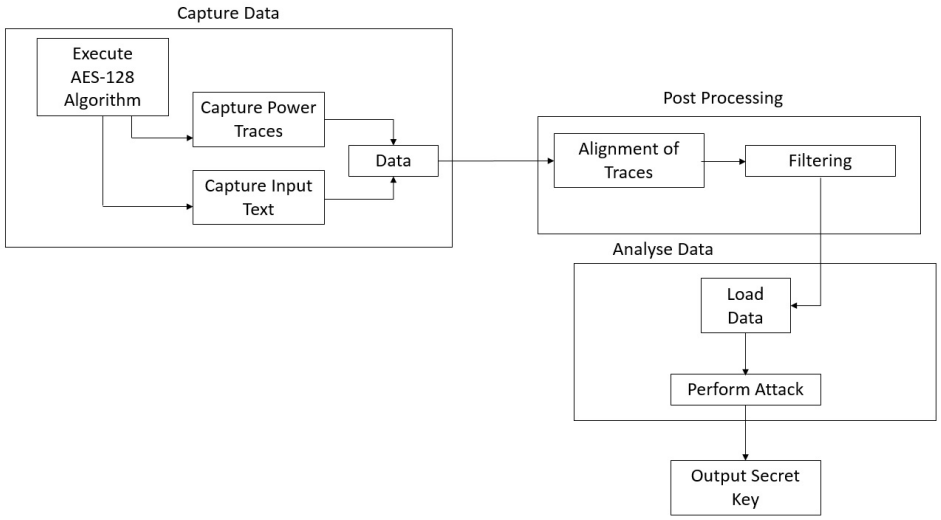


Fig. 4. The flow diagram of the experimental setup.

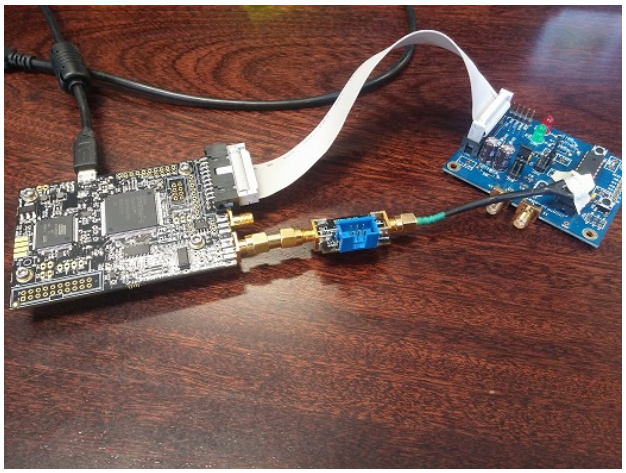

Fig. 5. The hardware Configuration: The ChipWhisperer capturing EM data from the Atmega328p processor.

## VI. Experiments, Results and Analysis

This sections covers the experiments carried out in this research, followed by the results obtained and an analysis on the results.

The first group of experiments was to recover the secret key from an unprotected implementation of the AES-128 algorithm. The AES-128 algorithm was executed on the Atmega328p microcontroller. While the execution was taking place, the EM emanations was captured, along with the input text and fixed secret key as the data. For this experiment The data was used with no post processing and attacked using the CPA attack approach. Table I illustrates the results acquired from the attack. The table contains the number of traces used, the number of subkeys recovered and finally, the success rate. The success rate was calculated by the number of recovered subkeys divided by the total number of the subkeys(16). It is observed from the table that all subkeys (16) or the entire key was recovered with only 50 traces using the CPA attack method.

TABLE I
CPA results against an unprotected implementation of AES-128.

| Traces | Recovered Subkeys | Success Rate |
|---|---|---|
| 10 | 3 | 0.1875 |
| 20 | 7 | 0.4375 |
| 30 | 10 | 0.625 |
| 40 | 14 | 0.875 |
| 50 | 16 | 1 |

The next experiment was to apply the template attack against the unprotected implementation. Since large amount of data is needed to create a template. The system captured two sets of data, firstly the training set and then the test set. The training set which was used to create the template consisted of 100 000 traces and the test set consisted of a 100 traces. Once the template was created the attack commenced and the results is depicted in Table II.

TABLE II
CPA results against an unprotected implementation of AES-128.

| Traces | Recovered Subkeys | Success Rate |
|---|---|---|
| 3 | 4 | 0.25 |
| 4 | 6 | 0.375 |
| 8 | 8 | 0.5 |
| 13 | 13 | 0.8125 |
| 16 | 15 | 0.9375 |
| 18 | 16 | 1 |

Table II displays that the template attack was successful. The attack only required 18 test samples to predict the correct

secret key. It is possible to create a template that would only require one test sample. However, this requires millions of training samples [11]. Based on the results obtained in the two experiments mentioned above, it is clear that the system is capable of recovering the secret key from an unprotected implementation of the AES-128 algorithm using both the CPA and template attack techniques.

The second experimental set consisted of attacking the AES-128 algorithm with the known and multi-thread countermeasure in effect. The known countermeasure was a combination of the hiding, shuffling, and random time delays countermeasures into one algorithm, followed by the multi-threaded countermeasure. Both these countermeasures was attacked by the CPA and template attack procedure.

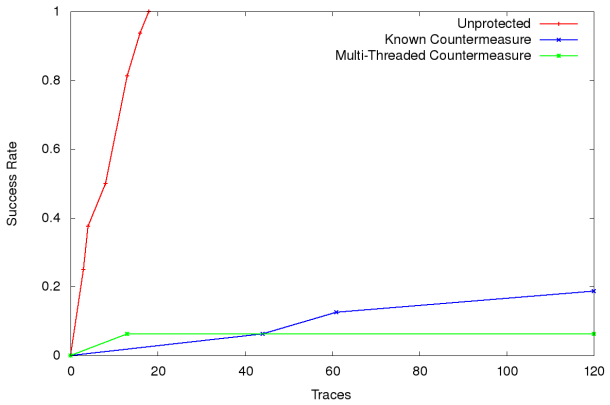


Fig. 6. Success rate graph for the template attack with 100 000 training and 120 testing samples.

Figure 6 depicts the success rate when 100 000 training and 120 testing samples were used for the template attack. The same calculation was used for the success rate as mentioned earlier. The figure illustrates for both the known and multi-threaded countermeasure the template attack was unsuccessful. However, the success rate of the known countermeasure was higher than that of the multi-threaded countermeasure. The known countermeasure leaked out three subkeys where as the multi-threaded countermeasure leaked out only one.

The designers of the multi-threaded approach stated that the write to EEPROM added to the resistance of their countermeasure. This research has noted that after the completion of the substitution round the write to memory has a high peak in power spike. The post processing phase was modified to add the functionality of removing these memory spikes from the captured power traces of the multi-threaded countermeasure. Figure 7 illustrates the comparison between 10 captured power traces from the multi-threaded countermeasure with and without the memory power signature. It is seen in Figure 7a after 8000 points there was a huge power spike when compared to the rest of the power signature. This power spike was when the algorithm was storing the random seed in the EEPROM. Figure 7b depicts the power traces as the post processing stage removed the memory spikes. Upon acquiring the modified power trace, the template attack was used to

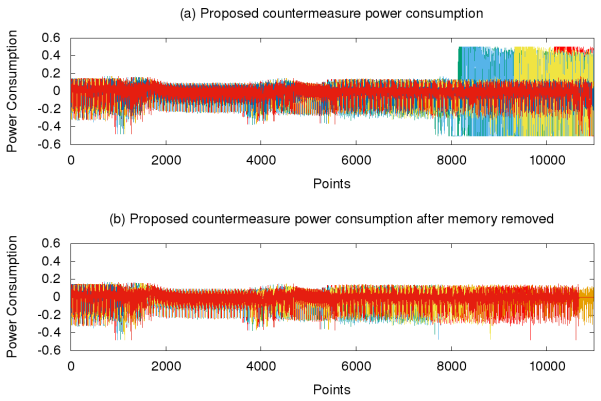


Fig. 7. A comparison between 10 captured power traces from the multi-threaded countermeasure with and without the memory power signature.

attack this data. Table III displays the results obtained. It is

TABLE III
TEMOLATE ATTACK RESULTS AFTER THE MEMORY WAS REMOVED FROM THE MULTI-THREADED COUNTERMEASURE.

| Traces | Recovered Subkeys | Success Rate |
|---|---|---|
| 6 | 1 | 0.0625 |
| 56 | 2 | 0.125 |

depicted that removing the memory yielded in retrieving an additional subkey. However, all the subkeys were not fully recovered. Furthermore, only two subkeys were recovered, that is still less than the subkeys when the known countermeasure was used.

The third set of experiments involved applying the post processing techniques of elastic alignment and filtering. The first stage was to apply the elastic alignment to the captured traces, followed by applying the SCA attacks. Figure 8 depicts an example of two traces misaligned, followed by the alignment after applying the elastic alignment technique. In Figure 8a it is seen that the two power traces are similar in pattern with each other. However, they are not aligned correctly after being captured. Furthermore, Figure 8b demonstrates the capabilities of the elastic alignment by aligning the traces correctly.

The elastic alignment algorithm was firstly applied to the captured traces of the known countermeasure. Table IV illustrates the results of the CPA attack against the captured traces of the known countermeasure after elastic alignment was applied. The table indicates that there was a great improvement

TABLE IV
CPA RESULTS AGAINST THE KNOWN COUNTERMEASURE WITH ELASTIC ALIGNMENT.

| Traces | Recovered Subkeys | Success Rate |
|---|---|---|
| 20 | 1 | 0.0625 |
| 40 | 4 | 0.25 |
| 60 | 6 | 0.375 |
| 100 | 10 | 0.625 |

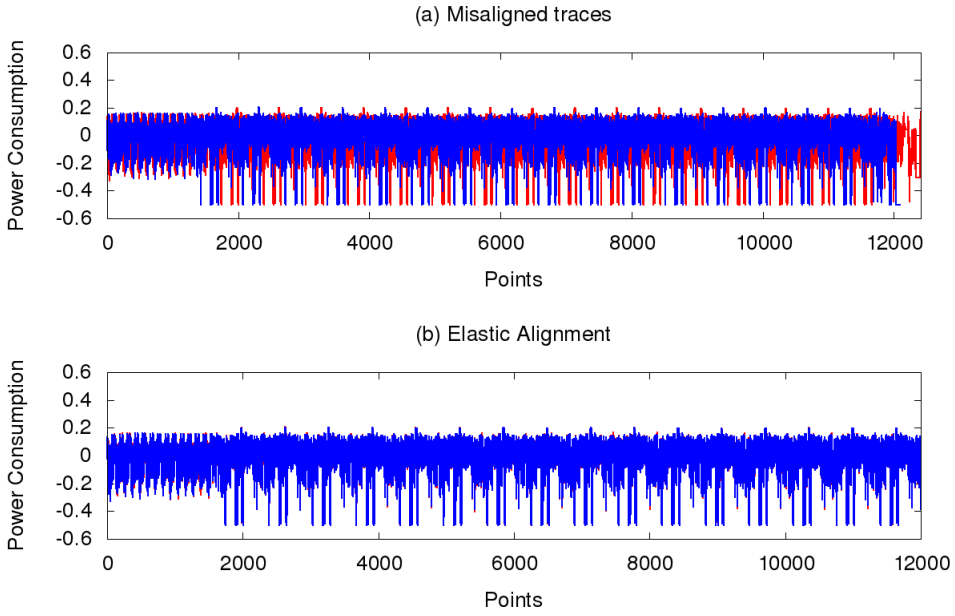


Fig. 8. Alignment of traces using the elastic alignment technique

towards recovering the subkeys using the elastic alignment technique. Comparing to earlier results were only three keys were recovered, this approach allowed the recovery of 10 subkeys with only requiring a 100 of traces as opposed to 100 000 traces which barely gave significant results. These results demonstrated the strength of using the elastic alignment processes.

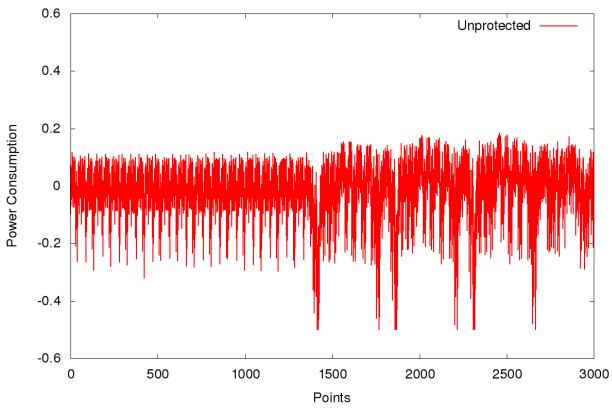


Fig. 9. The power consumption of an unprotected implementation of the AES-128 algorithm

The Savitzky?Golay (SG) filter is a powerful filter, capable of removing noise and keeping the important information. To demonstrate this, the filter is applied to the unprotected power traces that was captured. Figure 9 depicts the power consumption of an unprotected implementation of the AES-128 algorithm. As mentioned before, the system was able to recover the entire secret key.

The aim is to apply the SG filter on these traces and achieve the successful recovery of the secret key. Figure 10 demonstrates the power consumption as the SG filter is applied. From the figure it is seen that there is a dramatic reduction in the noise levels compared to that of Figure 9. As mentioned in Subsection V-B the SG filter consists of using the least-squares polynomial approximation, thus this research empirically tested the various values which yielded the full recovery of the secret key. Table V illustrates the results of these findings.

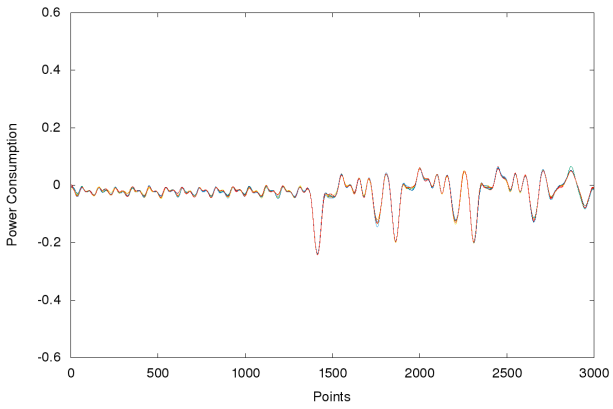


Fig. 10. Alignment of traces using the elastic alignment technique

Table V displays that using a polynomial of nine still leads to the full recovery of the secret key. However, it is noted that after the seventh polynomial the power traces had minimal to no reduction in noise as opposed to using a polynomial of five. Furthermore, the window size after 71 the traces had no change as well.

The next procedural step involved applying the SG filter

TABLE V
.

| Window Size | Polynomail | key Recovered |
|---|---|---|
| 51 | 3 | Yes |
| 71 | 3 | Yes |
| 91 | 3 | Yes |
| 51 | 5 | Yes |
| 71 | 5 | Yes |
| 91 | 5 | Yes |
| 51 | 7 | Yes |
| 71 | 7 | Yes |
| 91 | 7 | Yes |
| 51 | 9 | Yes |
| 71 | 9 | Yes |
| 91 | 9 | Yes |

onto the power consumption that was aligned by the elastic algorithm. Figure 11 illustrates a comparison of the success

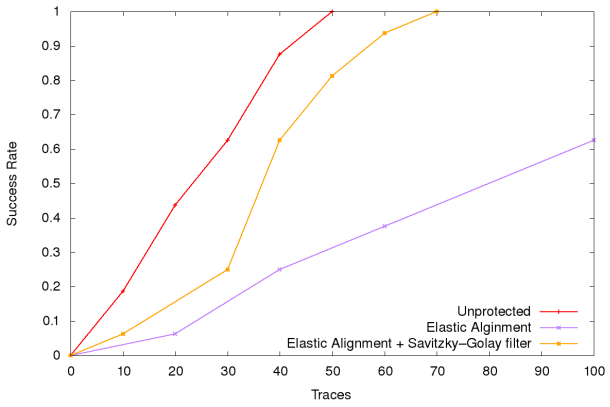


Fig. 11. The results obtained from the CPA attack after the additional signal processing was applied

ratio as the elastic alignment and SG filter was used on the known countermeasure power consumption when the CPA attack was performed. It is observed when using the elastic alignment alone, the success rate was 0.625, which relates to 10 subkeys being recovered. Furthermore, as both the elastic alignment and SG filter was used together the entire secret key was revealed. Compared to the unprotected implementation only 10 additional traces were required. This demonstrates that using both techniques in conjunction, the system was able to recover the secret key as three different known countermeasures was used. i.e: hiding; shuffling and random delays. Additionally, the process was applied to the multi-threaded power consumption. This research was unable to recover one subkey with a maximum of 1000 traces used for the CPA attack with elastic alignment and the SG filter where as it can be seen that only 60 samples were required to break the existing countermeasure.

Since the research has demonstrated that the preprocessing process is capable of retrieving the secret key with minimal required traces with a CPA attack. The techniques mentioned above is used on the power consumption for the known and multi-thread countermeasures with the template attack applied. A 100 000 traces were used as training samples and 100 training samples were used. Figure 12 illustrates these results. It is observed that applying the elastic alignment and the

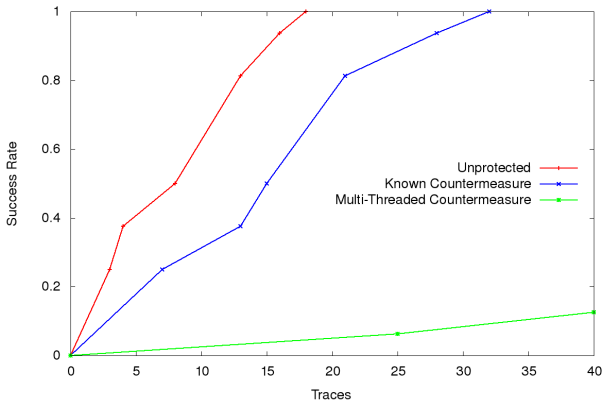


Fig. 12. The results obtained from the template attack after the additional signal processing was applied

SG filter to the known countermeasure the entire secret key is recovered with only 32 testing samples. Furthermore, the multi-threaded countermeasure only revealed two subkeys and mitigates these additional signal processing techniques. This further reinforces that the multi-threading technique is capable of withstanding various types of attacks.

The research has demonstrated the ability to retrieve the secret key on an unprotected cryptographic algorithm as well a protected implementation. It is shown that the entire secret key was recovered as the combination of known countermeasures of hding, shuffling and random delays was implemented. However, the research was unable to recover the full secret key as the multi-threaded countermeasure was used. The elastic alignment depends on the power consumption to be similar to reconstruct a signal that is close to the reference signal. Since the multi-threaded countermeasure has dynamic power consumption it is difficult to establish a good reference trace to align the rest of the power traces on as there are many permutations for a power trace. The filtering becomes ineffective as power consumption of the multi-threaded varies and becomes difficult to align.

Comparing the two countermeasures, the multi-threading approach is successful in mitigating information being leaked from the device. The multi-threaded countermeasure unique designed allowed for the countermeasure to mitigate the CPA and template attack. Furthermore, additional signal enhancement techniques had little effect on the mutli-threaded countermeasure.

## VII. Conclusion

This research extends the SCA attacks on the multiple thread countermeasure by performing both the CPA and template attacks. Furthermore, the research applies additional

processing techniques such as the elastic alignment and low pass filtering algorithms to enhance the attacks that was not previously applied against the countermeasure. Additionally, this research uses a far greater number of traces to perform the SCA attacks.

This work has demonstrated that the multi-threaded countermeasure was able to outperform the known combination of the hiding, shuffling, and random delay countermeasures by preventing the recovery of the secret key from an AES-128 cryptographic algorithm. Furthermore, the multi-threaded countermeasure is resistant to the CPA and template attacks. The multi-threaded countermeasure further demonstrates that it is able to mitigate additional signal processing techniques such as the elastic alignment and various digital filtering.

## Future Work

Since this work has demonstrated that the multi-threaded countermeasure is capable of mitigating side channel analysis attacks when using a microcontroller, it is thus our aim to implement this countermeasure on a system that has true multi-threading and multi-core functionality in the hope of preventing side channel analysis attacks against high powered devices.

## Acknowledgment

This work was undertaken as part of the Distributed Multimedia CoE at Rhodes University, with financial support from the Information Security Competency Area within Modelling and Digital Science at the 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.

## References

[1] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong authentication for rfid systems using the aes algorithm," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 357–370.

[2] U.S. Depatment of Commerce, "Advanced encryption standard (AES)," National Institute of Standards and Technology (NIST), Tech. Rep., 2001.

[3] 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*. IEEE, 2015, pp. 750–755.

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

[5] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded SystemsCHES 2001*. Springer, 2001, pp. 251–261.

[6] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology – CRYPTO99*. Springer, 1999, pp. 388–397.

[7] I. Frieslaar and B. Irwin, "Towards a software approach to mitigate correlation power analysis," in *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016.*, 2016, pp. 403–410.

[8] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 16–29.

[9] H. Mestiri, N. Benhadjyoussef, M. Machhout, and R. Tourki, "A comparative study of power consumption models for cpa attack," *International Journal of Computer Network and Information Security*, vol. 5, no. 3, p. 25, 2013.

[10] J. Daemen and V. Rijmen, "Rijndael/aes," in *Encyclopedia of Cryptography and Security*. Springer, 2005, pp. 520–524.

[11] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 13–28.

[12] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[13] T. Plos, M. Hutter, and M. Feldhofer, "Evaluation of side-channel preprocessing techniques on cryptographic-enabled hf and uhf rfid-tag prototypes," in *Workshop on RFID Security*, 2008, pp. 114–127.

[14] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert, "Shuffling against side-channel attacks: A comprehensive study with cautionary note," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2012, pp. 740–757.

[15] ZTEX, "Spartan 6 LX9 to LX25 FPGA Board." Nov. 2016. [Online]. Available: http://www.ztex.de/usb-fpga-1/usb-fpga-1.11.e.html

[16] 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*. Springer, 2012, pp. 341–356.

[17] R. W. Schafer, "What is a savitzky-golay filter?[lecture notes]," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[18] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[19] J. G. van Woudenberg, M. F. Witteman, and B. Bakker, "Improving differential power analysis by elastic alignment," in *Cryptographers Track at the RSA Conference*. Springer, 2011, pp. 104–119.