# S-NET

## A Confusion Based Countermeasure Against Power Attacks for SBOX

Aljuffri, Abdullah; Venkatachalam, Pradeep; Reinbrecht, Cezar; Hamdioui, Said; Taouil, Mottaqiallah

**DOI**
10.1007/978-3-030-60939-9_20

**Publication date**
2020

**Document Version**
Final published version

**Published in**
Embedded Computer Systems

**Citation (APA)**
Aljuffri, A., Venkatachalam, P., Reinbrecht, C., Hamdioui, S., & Taouil, M. (2020). S-NET: A Confusion Based Countermeasure Against Power Attacks for SBOX. In A. Orailoglu, M. Jung, & M. Reichenbach (Eds.), *Embedded Computer Systems: Architectures, Modeling, and Simulation - 20th International Conference, SAMOS 2020, Proceedings* (pp. 295-307). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 12471 LNCS). Springer. https://doi.org/10.1007/978-3-030-60939-9_20

# S-NET: A Confusion Based Countermeasure Against Power Attacks for SBOX

Abdullah Aljuffri[1,2(✉)], Pradeep Venkatachalam[1], Cezar Reinbrecht[1], Said Hamdioui[1], and Mottaqiallah Taouil[1]

[1] Delft University of Technology, Delft, The Netherlands
{A.A.M.Aljuffri,P.Venkatachalam,C.R.WedigReinbrecht, S.Hamdioui,M.Taouil}@tudelft.nl
[2] King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia
aaljuffri@kacst.edu.sa

**Abstract.** Side channel attacks are recognized as one of the most powerful attacks due to their ability to extract secret key information by analyzing the unintended leakage generated during operation. This makes them highly attractive for attackers. The current countermeasures focus on either randomizing the leakage by obfuscating the power consumption of all operations or blinding the leakage by maintaining a similar power consumption for all operations. Although these techniques help hiding the power-leakage correlation, they do not remove the correlation completely. This paper proposes a new countermeasure type, referred to as *confusion*, that aims to break the linear correlation between the leakage model and the power consumption and hence confuses attackers. It realizes this by replacing the traditional SBOX implementation with a neural network referred to as S-NET. As a case study, the security of Advanced Encryption Standard (AES) software implementations with both conventional SBOX and S-NET are evaluated. Based on our experimental results, S-NET leaks no information and is resilient against popular attacks such as differential and correlation power analysis.

**Keywords:** S-NET · Side channel analysis · Neural network · SBOX · Advanced Encryption Standard

## 1 Introduction

Since it was selected by National Institute of Standards and Technology (NIST) in 2001 as the official standard for block cipher cryptography [1], Advanced Encryption Standard (AES) gain a massive adoption in network communication protocols [2]. Nowadays AES is the most widely used symmetric encryption algorithm [2]. Hence, it is attractive for attackers to comprise keys which may lead to massive data breaches. A recent study made by IBM and Ponemon Institute showed that the average cost of a single data breach is estimated to be


A. Orailoglu et al. (Eds.): SAMOS 2020, LNCS 12471, pp. 295–307, 2020.
https://doi.org/10.1007/978-3-030-60939-9_20

$3.86 million [3], not to mention the additional reputation damage. Theoretically, AES is considered to be resistant against all attacks with the exception of brute force attacks. However, attackers found means to compromise the security of such algorithms based on their implementation [4]. One of the most popular techniques to retrieve the key is by analyzing the physical characteristics such as power consumption known as side channel attack (SCA). SCA poses a serious threat to the security of current encryption algorithms. As a consequence, the implementations of these algorithms have to be reevaluated and secure implementations have to be developed.

The current developed countermeasures can be classified into two groups referred to as *randomization* and *blinding*. Randomization aims at obfuscating the power consumption irrespective of the executed operation, while blinding aims at leveling the power consumption (i.e., keep as constant as possible) during the encryption/decryption. *Randomization* was first proposed in 1999 by Chari et al. [5] shortly after the first power attack was introduced in 1998 [6]. The authors split the operation using a random factor known as masking. However, this masking technique does not protect against second-order power attacks [7]. Higher order masking [8] was thereafter introduced to defend against the attacks; unfortunately, it decreased the performance significantly without guaranteeing protection. Other randomization countermeasure techniques such as dummy delay insertion [9] and shuffling [10] were also proposed. These techniques made attacks harder but could not necessarily prevent them [11,12]. In the second group of countermeasures, i.e. *blinding*, also several countermeasures have been proposed. In [13] the authors proposed dual-rail logic, a technique where all input and output signals of a gate also have complementary values. Hence, the technique balances the number of transitions. Another example of *blinding* consists of duplicating the design where one part operates on the original message while the other on the complementary message [14]. However, both these techniques still leak information due two reasons. First, a difference in load capacitance between the two complementary logic gates may cause an unbalanced power leakage for different input values. Second, different arrival times of signals leaks information as well [15]. Hence, both *randomizing* and *blinding* countermeasures are susceptible to power attacks as both techniques are focusing on covering the problem instead of solving it. To solve the leakage issue, the linear correlation between the power consumption and leakage model must be broken.

This paper proposes a radical new countermeasure type that aims to break the linear correlation between the power consumption and the leakage model. We realize this by substituting the SBOX operation of AES with a neural network which we call S-NET (short for substitution neural network). Due to the chaotic nature of S-NET and removal of the linear power-leakage relation, we classify this countermeasure as *confusion*. The main contributions of this paper can be summarized as follow:

– Proposal of S-NET: a new countermeasure based on *confusion*. It nullifies power attacks by invalidating the existing power-leakage models.

- Implementation of S-NET. This includes designing, training, and testing of an appropriate neural network.
- Validating S-NET security using conformance testing by applying signal-to-noise ratio analysis as a leakage assessment and evaluation style testing by applying key ranking analysis based on the most popular power attacks.

The rest of the paper is organized as follows. Section 2 provides a background on SBOX, side channel attacks, and neural networks. Section 3 explains the confusion countermeasure and the methodology applied to design S-NET. Section 4 validates its security. Section 5 discusses the benefits and limitations of S-NET. Finally, Sect. 6 concludes this paper.

## 2 Background

This section provides a background on SBOX, the most targeted component in SCA and the different SCA types. Thereafter, it introduces neural networks.

### 2.1 Substitution Box (SBOX)

An SBOX (or so called SubByte in AES) is an essential nonlinear substitution operation that is used in every block cipher. The purpose of an SBOX is to create confusion, i.e., to obscure the relationship between the private key and the ciphertext [16]. Among all the operations in block ciphers, the SBOX leaks the most information; hence it is typically the target of side channel attackers. The SBOX in AES has an 8-bit input and works as follows [1]:

1. The multiplicative inverse of the 8-bit input is calculated based on the finite Galois Field $GF(2^8)$ and the irreducible polynomial $p(x) = x^8+x^4+x^3+x+1$.
2. The intermediate result of the previous step is transformed using a predefined affine transformation.

To speed the calculations up, typically a 256-input Look-Up Table (LUT) is used containing pre-calculated values. Note that AES also contains other operations such as AddRoundKey MixColumns, and ShitfRows [1]. However, these are less relevant for side channel attacks and hence not addressed.

### 2.2 Side Channel Analysis

Side channels are observable characteristics such as time, power consumption, electromagnetic radiation, light, noise, heat, etc. that may leak secret information unintentionally. By analyzing these characteristics, i.e., perform a side channel attack, cryptographic keys can be retrieved [17]. Most of these type of attacks are non-invasive, as the observable characteristics can be observed from outside the chip and hence SCA attacks are relatively cheap. From the SCA attacks, power attacks are the most popular ones and hence the topic of this

paper. Power attacks are statistical analysis of the power consumption measurements at an intermediate target (e.g., SBOX operation) that are correlated to a leakage model. Leakage models make assumptions on how the secret information is leaked based on the operations and switching activity. Examples are [18]:

- hamming weight: HW = ones(Intermediate AES function)
- hamming distance: HD = ones(plain-/ciphertext $\bigoplus$ HW)

The function ones() represent the number of ones in a byte, while Intermediate AES function is the targeted function of the AES ecnryption/decryption process. Examples are:

- AddRoundKey = $plaintext \bigoplus key$
- SubByte = $SBOX[plaintext \bigoplus key]$
- LastRound = $SBOX^{-1}[ciphertext \bigoplus key]$

Power attacks can be classified in non-profiled and profiled attacks. Each class is briefly explained next.

- **Non-profiled power attacks:** in these attacks, an attacker gets access to a target electronic device that runs a cryptographic algorithm. Thereafter, he or she tries to perform a key recovery by correlating a leakage model with obtained power traces during the execution of the cryptographic algorithm. Famous examples of this type of attacks are Differential Power Attack (DPA) [6] and Correlation Power Attack (CPA) [18].
- **Profiled power attacks:** unlike the non-profiled attacks, in these attacks, an attacker holds a device under his control similar to the target device in order to build a leakage template. Thereafter, he or she takes advantage of this template to exploit the leakage of the target device and performs a key recovery. Famous examples of these types of attacks are template based power attacks (TBA) [19] and Deep Learning based Side Channel Attacks (DL-SCA) [20].In this paper we will focus only on non-profiled attacks.

### 2.3 Neural Network

A neural network (NN) is a simplified mathematical representation of a biological network of neurons, where each biological neuron is represented by an artificial neuron [21]. In its simplest form, an NN consists of a single node; it is also know as a Logistic Regression (LR) model that consists of two steps. First, it calculates $\sum_{i=1}^{n} x_i w_i + bias$; here $x_i$ presents the $i^{th}$ input value and $w_i$ its associated weight. Subsequently, a nonlinear activation function is applied. The output of this model is a probabilistic decision. More complex NNs can be constructed by forming a network of multiple layers of neurons to form the so-called Multilayer Perceptron (MLP). MLP mainly consists of three types of layers. The first layer is called the input layer and serves as the data entry point to the neural network. The second type of layer is called the hidden layer. The hidden layers of the NN are responsible for the extraction of features of the input data. The size of

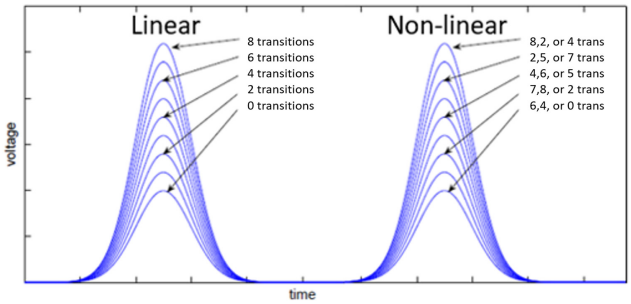



**Fig. 1.** Linear power-leakage correlation (modified from [23]).

the hidden layers is specified by the width and depth; the width represents the number of neurons in each layer while the depth refers to the number of layers. The final layer is called the output layer and is responsible for making decisions.

NNs can be used to map any finite or continues mathematical function independent of its complexity. According to the Universality Approximation Theorem (UAT) [22], a feed-forward neural network with a single hidden layer with a finite number of neurons and an arbitrary activation function can be used to approximate any continuous function. As the SBOX described in Sect. 2.1 is based on mathematical functions, it can theoretically be implemented using a neural network. In the next section we describe how we achieve this.

## 3 S-NET: A Countermeasure Based on Confusion

This section explains the *confusion* countermeasure, the idea behind S-NET and finally, the methodology to design it.

### 3.1 Confusion: Invalidating the Leakage Model

In side channel analysis (SCA), an attacker correlates the power consumption with a leakage model assuming a linear relation between them. In other words, a higher power consumption results in a larger hamming weight/distance as illustrated in the left part of Fig. 1. Hence, the different hamming weights/distances are traceable in the power traces. Note that the countermeasures based on randomization and blinding try to make this harder, but are typically not able to completely hide this linear relation when statistical analysis are performed. The reason for this is that these countermeasures only try to modify the power consumption, as shown in the left part of Fig. 2. On the other hand, it would be much more difficult for attackers to analyze power traces when the relation between the hamming weight/distance is nonlinear with the actual power consumption as the right part of Fig. 1 shows. In such a scenario, based on the message-key combination, different hamming weights/distances might have the same power consumption and message-key combinations with the same hamming weights/distances

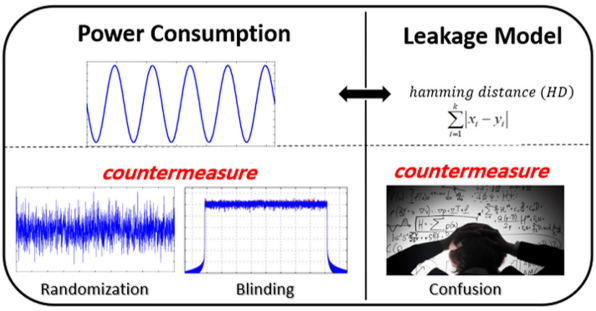



**Fig. 2.** Visual explanation of the confusion concept.

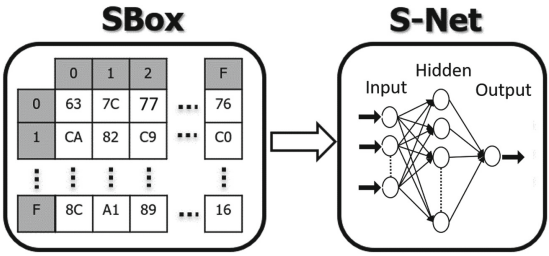



**Fig. 3.** SBOX representation in S-NET

might have a different power consumption; hence, attacks based on hamming weight/distance are confusing and not effective. The reason for this is that such a countermeasure confuses the leakage in relation to the power consumption. Therefore, this countermeasure targets the leakage model as illustrated in the right part of Fig. 2.

Note that the implementation of S-NET inherits the non-linearity from the stochastic properties of neural networks. Generally any mathematical function that tries to break the linear power-leakage behaviour can be categorized as a countermeasure based on confusion.

### 3.2 Motivation Behind S-NET

Besides their stochastic properties, neural networks also have other benefits. Neural networks can be considered to a certain degree as black boxes as it is unclear how their internals precisely work. This property makes neural network based implementations difficult to be characterized. Hence, finding a good leakage model against it is extremely hard.

### 3.3 Design Methodology

Figure 3 shows the concept of S-NET. S-NET implements the SBOX operation using a neural network without affecting the remaining AES operations. The size

and weights of the neural network can be achieved by iterating over three steps, namely design, training, and optimization until a satisfying solution is reached. Thereafter, in the final and fourth step, the neural network is integrated with the other parts of AES. Each step is described in detail next.

**1. Designing S-NET:** This step describes the methodology used to define the sizes of the input, output and hidden layers of S-NET.

The SBOX is typically represented by the look-up-table (LUT) shown in left side of Fig. 3. The LUT contains 256 elements arranged in a table with 16 rows and 16 columns. The row index is specified by the first 4 input bits and the column index by the latter 4 input bits. Since a neural network is not a table, S-NET is designed differently. The input layer of S-NET is fixed to 8 neurons, each representing a single bit of the input, respectively. To improve the resilience against attacks, only a single neuron in the output layer has been used that generates the output byte of the SBOX. The size of the hidden layer, i.e., its width and depth, depends on how easy it is for the neural network to learn the content of the LUT. We have tried different widths and depths to find the optimal solution in term of computation and memory efficiency. We observed that the cheapest solution from a computational and memory point of view consists of using a single hidden layer for two reasons: 1) as the inputs are binary, no multiplications are required in the hidden layer, and 2) by reducing the depth to a single layer, data can be represented using less number of bits. Note that the range of intermediate values increases for a larger depth.

**2. Training S-NET:** This step describes the training process and how the weights and biases of S-NET are determined. Usually the data set consists of three subsets during the training of a neural network. One subset is used for the training of the network, one for the validation of the network, and one for evaluating the performance after the training is completed. However, in case of S-NET, only a single data set is used for training. The validation and evaluation are not needed as S-NET must be 100% functional, i.e., it must generate correct outputs for all 256 SBOX inputs.

**3. Optimizing S-NET:** This step describes the optimization techniques used to increase the performance and reduce the overhead of S-NET.

The computational complexity and memory overhead of neural networks make them undesirable solutions for both hardware and software applications. Therefore, to reduce the cost of the proposed solution, multiple optimization techniques are applied before, during, and after the training process. These techniques are highlighted next.

**Integer Weights:** It is well understood that integer operations have a significant performance benefit in comparison with floating point operations. Therefore, the weights of the neural network are rounded to the nearest integers after the training phase. After this step, all the inputs of the SBOX are reevaluated to guarantee correct operation.

**Constrain Weights:** The neural network typically produces a wide range of values for the weight set and hence floating point numbers are used by default during training. An implementation of a neural network in hardware and software would be more optimal if the weight set is restricted to a limited number of bits. In S-NET, we fixed the sizes of the weights to 16 bit integers, thereby speeding up the operations and lowering the memory overhead, especially when customized hardware operations are used.
**Reduce Multiplications:** Multiplications are one of the most expensive operations in the neural network. For this reason, S-NET is designed to have a single hidden layer where no multiplications are needed as the input neurons are represented by a single bit. In the output layer, the number of multiplication is reduced by setting a threshold for the weight. Any weight value below this threshold is skipped. Hence, it results in a lower computational overhead.
**Use Simple Activation Functions:** Each neuron contains an activation function. The input to this activation is equal to the sum of the product of the inputs and weights of the neuron plus the bias. Many functions have been used as activation function such as tanh, sigmoid, Rectified Linear Unit (ReLU), etc. The computational complexity of these functions varies. In our design we intentionally chose Relu for the hidden layer and no activation function in the final layer to achieve simplicity in both software and hardware implementations.

**4. Integrating S-NET** in this final step, the designed S-NET component is integrated into the AES implementation by replacing the conventional SBOX.

## 4 Validation

This section describes the platform and validates the security of S-NET by analyzing the power traces using evaluation and conformance style testing.

### 4.1 Experiment Setup

To validate the proposed concept of the countermeasure, we compare the security of an unprotected and protected software implementation of AES128, where the protected implementation uses S-NET. The software implementations run on the Chipwhisperer board from NewAE Technology Inc [24]. It is a development board that comes with the Atmel XMEGA microcontroller as target device. We used the unprotected open source AES128 implementation that comes with the board as our reference for the unprotected AES128 implementation. The power consumption is measured with an ADC that is integrated in the development board. Finally, the development board is connected to a computer to control the execution and storage of the power traces.

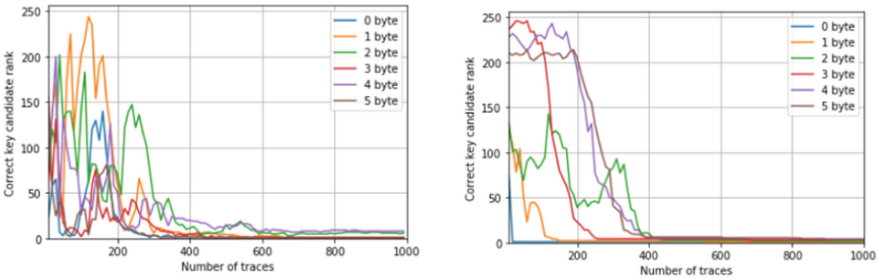



**Fig. 4.** Ranking analysis results of unprotected SBOX implementation

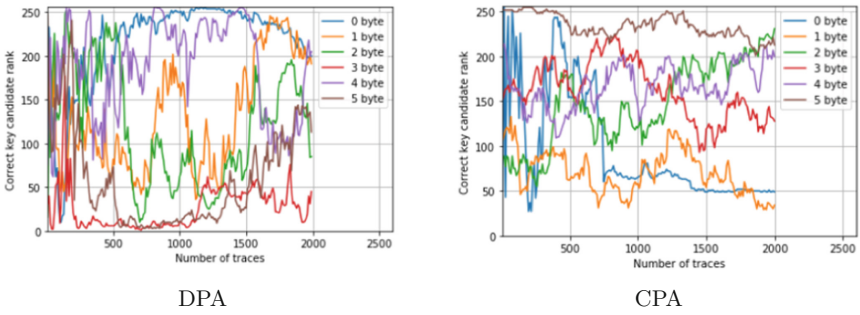



**Fig. 5.** Ranking analysis results of S-NET implementation

### 4.2 Results Analysis

To analyze the security of both the unprotected and protected AES implementations, two analysis methods are applied. They are referred to in literature as evaluation-style and conformance-style testing.

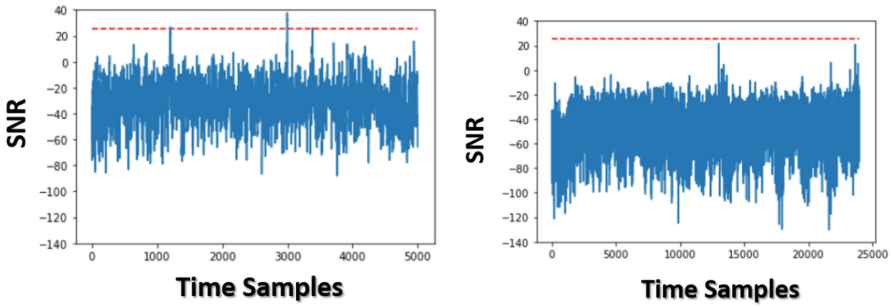
First, in evaluation-style testing traces are examined based on real attacks scenarios, preferably by advanced state-of-the-art attacks. They reveal whether the implementations are resilient against these attacks or not. Here, we limit ourselves to the most famous power attacks; they are: differential power analysis (DPA), and correlation power analysis (CPA). Second, in conformance-style testing the traces are checked to meet certain leakage requirements, without considering attacks. Examples of such analysis are TVLA [25] and signal-to-noise ratio (SNR) analysis [26]. Due to space limitations, we only limit ourselves to SNR analysis. The results of both analysis methods are provided next.

**Evaluation-style Testing:** Two popular attacks (i.e. DPA and CPA) are performed on the recorded traces of the unprotected and protected implementations.The traces are generated based on fixed keys. For each attack, we evaluate the rank of the correct sub-key values (i.e., 8 bits of the 128-bit key). A rank of zero means that the attacker is able to retrieve the correct sub-key, while a rank

**Table 1.** Number of correctly predicted sub-keys

| | Unprotected | | Protected (S-NET) | |
|---|---|---|---|---|
| Leakage Model & Attack | DPA | CPA | DPA | CPA |
| HW(AddRoundKey) | 0 | 0 | 0 | 0 |
| HW/HD(SubByte) | 16 | 16 | 0 | 0 |
| HW/HD(LastRound) | 16 | 16 | 0 | 0 |

of 255 represents the lowest confidence of guessing the right sub-key. Figure 4 shows the rank analysis of the first 6 bytes for both attacks for the unprotected implementation. The figure clearly shows, as expected, that the sub-key can be retrieved successfully when approximately 400 traces are used; this applies for both attacks. In contrast, the two attacks were unsuccessful for the protected S-NET implementation as shown in Fig. 5. The rank of the correct key behaves chaotically and never reaches zero and hence the correct sub-key could not be retrieved. The analysis have been done using only a single weight set for S-NET.




Unprotected SBOX implementation S-NET implementation

**Fig. 6.** SNR analysis results

**Conformance-style Testing:** Figures 6a and 6b show the SNR analysis of the unprotected and protected implementation, respectively. The traces for the analysis are generated based on random keys. The maximum SNR value of both figures differs. For the unprotected case, a high SNR value of 37.6 is observed around sample 3000 which is higher than the considered threshold value (which equals 25 [27]); hence, information leakes. However, for the protected case, the highest observed SNR value is 21.5 around sample 14000, which is below the minimum threshold value. Hence, it is hard to extract the secret key.

The results based on both evaluation-style and conformance-style testing clearly show that S-NET is secure against CPA and DPA power attacks. This can also be seen in Table 1. In the protected case, we were not able to recover

any of the sub-key values. However, for the unprotected case, all the 16 sub-keys were successfully retrieved for attacks based on SubByte and LastRound (see Sect. 2.2), for both DPA and CPA using both hamming distance (HD) and hamming weight (HW).

## 5 Discussion

This work proposes a new countermeasure type against SCA. Based on our experiments, we conclude the following:

**Security:** S-NET provides a unique solution to the leakage problem which is different from the randomization and blinding techniques, as it tries to break the linear correlation between power consumption and leakage model. It has additional benefits as it makes characterization difficult due to the inherent nature of neural networks.

**Performance:** The software implementation of S-NET has a large timing overhead. The protected AES runs 75 times slower than the unprotected AES implementation. However, this delay overhead is comparable to other countermeasures such as masking where the timing overhead is larger than 100x [28].

**Hardware Implementation:** One way to speed-up S-NET is to implement the neural network in hardware. In hardware, S-NET can be implemented in a single or a couple of cycles which increases the performance. However, it might impact the area overhead negatively.

**Optimization:** To tackle the area and performance issues two possible solutions can be investigated. The first method is based on the usage of emerging memory technologies. Several articles already showed that neural networks implemented with resistive memories have a huge area reduction compared to CMOS implementations [29]. Another way of trying to improve the performance is by looking at other (mathematical) functions that create a nonlinear power-leakage correlation. In case such a function exists, its implementation is most likely cheaper than using a neural network.

**Applicability:** Due to its widely usage, the SBOX of AES has been used to implement S-NET. However, the S-NET countermeasure technique can be easily applied to other block ciphers such as Data Encryption Standard (DES), Blowfish, Towfish, etc. In addition to that, it can be used to secure lightweight encryption systems such as PRESENT, which lately are gaining increasing attention due to increase in Internet-of-Things (IoT) applications.

**Analysis:** In this paper we proved that S-NET was able to secure AES against power attacks. However, other side channel attacks such as timing and electromagnetic field have not been investigated. Nevertheless, we believe that they are less powerful than power attacks.

## 6 Conclusion

This paper introduced a new countermeasure type against side channel power attacks referred to as *confusion*. In contrast to blinding and masking, that try

to hide the leakage, confusion countermeasure solves the leakage problem by removing the linear correlation between the power consumption and leakage model. We realized this by deploying a neural network, referred to as S-NET. The experimental results showed that S-NET is immune against DPA and CPA. However, the performance results showed a 75 times higher execution time than the conventional implementation. Overall, S-NET has the potential to replace existing countermeasures due to its high security.

**Acknowledgments.** This work was labelled by the EUREKA cluster PENTA and funded by Dutch authorities under grant agreement PENTA-2018e-17004-SunRISE.

## References

1. NIST: Announcing the advanced encryption standard (AES). Fed. Inf. Process. Stan. Publ. **197** 3 (2001)
2. Leech, D.P., et al.: The economic impacts of the advanced encryption standard, 1996–2017. NIST (2018)
3. IBM: 2019 Cost of a Data Breach Report: IBM Security (2019).https://databreachcalculator.mybluemix.net/. Accessed 23 Sept 2019
4. Ors, S.B., et al.: Power analysis attack on an ASIC AES implementation. In: ITCC (2004)
5. Chari, S., et al.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
6. Kocher, P., et al.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
7. Messerges, T.S.: Securing the AES finalists against power analysis attacks. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44706-7_11
8. Coron, J.S., et al.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 410–424. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_21
9. Durvaux, F., et al.: Efficient removal of random delays from embedded software implementations using hidden Markov models. In: Mangard, S. (ed.) CARDIS 2012. LNCS, vol. 7771, pp. 123–140. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37288-9_9
10. Luo, P., et al.: Towards secure cryptographic software implementation against side-channel power analysis attacks. In: ASAP (2015)
11. Durvaux, F., et al.: Cryptanalysis of the CHES 2009/2010 random delay countermeasure. IACR Cryptol. ePrint Arch. **2012**, 38 (2012)
12. Veyrat-Charvillon, N., et al.: Shuffling against Side-channel attacks: a comprehensive study with cautionary note. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 740–757. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_44

13. Tiri, K., et al.: A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: 28th European Solid-State Circuits Conference (2002)
14. Ambrose, J.A., et al.: MUTE-AES: a multiprocessor architecture to prevent power analysis based side channel attack of the AES algorithm. In: ICCD (2008)
15. Fang, X., et al.: Leakage evaluation on power balance countermeasure against side-channel attack on FPGAs. In: IEEE HPEC (2015)
16. Shannon, C.E.: Communication theory of secrecy systems. The Bell Syst. Tech. J. **25**(4), 656–715 (1949)
17. Zhou, Y., Feng, D.: Side-channel attacks: ten years after its publication and the impacts on cryptographic module security testing. http://eprint.iacr.org/2005/388. Accessed 23 Sept 2019
18. Brier, E., et al.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2
19. Chari, S., et al.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
20. Maghrebi, H., et al.: Breaking cryptographic implementations using deep learning techniques. In: IACR Cryptology ePrint Archive (2016)
21. Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, Cambridge (1995)
22. Csáji, B.C.: Approximation with artificial neural networks. Master's thesis, Eötvös Loránd University, Hungary (2001)
23. Standaert, F.-X.: Introduction to Side-Channel Attacks. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-71829-3_2
24. N. T. Inc: Chipwhisperer-Lite two part board. http://store.newae.com/chipwhisperer-lite-cw1173-two-part-version/. Accessed 31 Jan 2020
25. Becker, G., et al.: Test vector leakage assessment ( TVLA ) methodology in practice (2011). https://pdfs.semanticscholar.org/. Accessed 23 Sept 2019
26. Mangard, S.: Hardware countermeasures against DPA – a statistical analysis of their effectiveness. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 222–235. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24660-2_18
27. N. technology inc: Measuring SNR of Target. https://chipwhisperer.readthedocs.io/en/latest/tutorials/pa_intro_3-openadc-cwlitearm.html/. Accessed 13 May 2020
28. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_28
29. Sun, J.: CMOS and memristor technologies for neuromorphic computing applications. Technical report, University of California at Berkeley (2015)