Power Side Channel Attacks: Where Are We Standing?

Taouil, M.; Aljuffri, A.A.M.; Hamdioui, S.

DOI
10.1109/DTIS53253.2021.9505075

Publication date
2021

Document Version
Final published version

Published in
2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)

2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS) | 978-1-6654-3654-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DTIS53253.2021.9505075

Power Side Channel Attacks: Where Are We Standing?

Mottaqiallah Taouil, Abdullah Aljuffri, Said Hamdioui
Computer Engineering
Delft University of Technology
Delft, The Netherlands
{m.taouil,a.a.m.aljuffri,s.hamdioui}@tudelft.nl

***Abstract*—Side channel attacks are a serious threat to integrated circuits. They are hardly detectable and use inherent information leaked by the hardware to infer sensitive information like secret keys. Over the last ten years, numerous side channel attacks have been examined, exploring various forms of leakage channels such as time, power, electromagnetic field, photon emission, and acoustic. Among them, power side channel attacks are the most popular ones. Developing an appropriate countermeasure against such attacks requires a deep understanding of these attacks. This paper presents a study of the most popular power attacks such as differential power attack and correlation power attack and discusses the latest countermeasures in this domain and their shortcomings.**

***Index Terms*—Side channel attacks, profiled attacks, non-profiled attacks, countermeasures, leakage assessment**

I. Introduction

Without a proper implementation, cryptographic algorithms such as Advanced Encryption Standard (AES), RSA, Elliptic Curve Cryptography (ECC) etc. are vulnerable to data attacks [1]. Several studies showed that adversaries can easily break the secrecy of the cryptographic algorithms using hardware attacks. Examples of such attacks are fault injection [2], side channel analysis (SCA) [3, 4], and hardware Trojans [5]. Among these hardware attacks, side channel attacks are one of the most difficult ones to protect as they are completely passive. These attacks extract sensitive information using covert channels such as power consumption, electromagnetic radiation, timing information, etc. Side channel attacks are continuously improved (e.g., usage of deep learning [6]) while the required equipment also becomes cheaper [7]. This makes it is important to understand where we are currently standing with respect to side channel attacks.

The first side channel attack was introduced by Paul C. Kocher in 1996 [3]. The author showed that carefully measuring the execution time of certain operations can reveal the secret key. Three years later, Kocher et al. [4] showed that power consumption can also leak secret information. Many different power based side channel attacks have been proposed after that. They have been grouped and classified in several papers. In [8], the authors presented a tutorial on physical attacks in which they extensively discussed simple power analysis, differential power analysis, and correlation power analysis. In [9], the authors published a chapter on side channel attacks and leakage assessment equipment. In [1, 10] the authors presented an overview on power attacks and hardware-based countermeasures. In [11], the authors published a survey on the security of differential power analysis. In [12], the authors published a study on machine learning-based power threats. Last but not least, in [13] the authors presented an in-depth study on the most popular power attacks as well as test vector leakage assessment. Note that most of these surveys focus on a specific topic of side channel attacks. For example, in [10] and [1], the authors focus only on hardware-based countermeasures where other implementation such as software ones are ignored. In [11], the survey targeted a single attack namely differential power analysis. In [12] the focus was only on machine learning attacks, while [13] ignored machine learning attacks in their survey. A survey that covers most aspects in power side channel analysis aspects (i.e., attacks, countermeasures, leakage assessment styles) is still missing.

This paper briefly discusses the history of power based side channel attacks and their countermeasures. It provides a classification of the attacks and countermeasures and gives examples of the most common attacks and countermeasures within each class, respectively. Finally, it discusses the various methods to perform a leakage assessment of the implemented cryptographic algorithms. In summary, the contributions of this paper are:

- An overview of the current threats of the side channel attacks
- A classification of the existing countermeasures and their limitations
- An overview of leakage assessment techniques and their shortcomings.

The remainder of the paper is organized as follows. Section II provides a classification of power attacks and describes the most famous ones in more detail. Section III classifies the state-of-the-art countermeasures and provides examples of each class. Section IV gives an overview of proposed leakage assessment techniques. Finally, Section V discusses and concludes this paper.

II. Side Channel Attacks

Power based side channel attacks are attacks where a malicious adversary takes advantage of the power consumption to deduce secret information. These attacks can be classified in non-profiled and profiled attacks as shown in Figure 1. Each class is briefly explained next.

A. Non-profiled attacks techniques

In these attacks, an attacker gets access to a target electronic device that runs a cryptographic algorithm. Thereafter, the at-

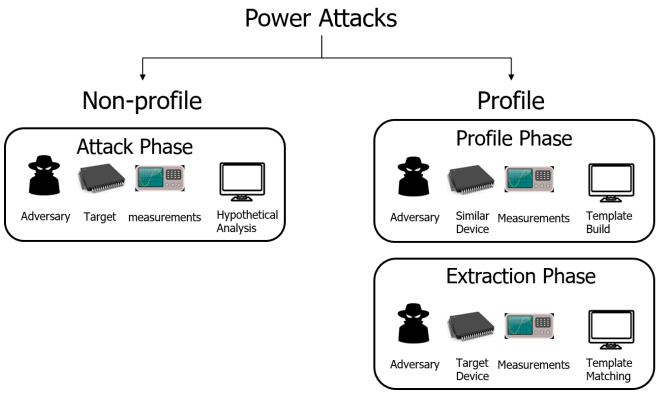

Fig. 1. Classification of Power Attacks

tacker 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 these types of attacks are Simple Power Attack(SPA) [4], Differential Power Attack (DPA) [4], Correlation Power Attack (CPA) [14], Collision Power Attack [15], Zero Value Attack [16], and Machine learning Attack [17]. Each one of these attacks will be explained briefly next.

1) Simple Power Attack (SPA): An SPA attack can be carried out by merely observing changes in power usage throughout the execution of the target operation (e.g., RSA encryption). It's worth noting that in this attack no particular mathematical computations are required. The attack on the unprotected RSA implementation based on the multiply-square algorithm is a well-known example [4]. Observing the peak power values of the square and multiply operations during encryption and decryption allows the attacker to retrieve the key [4].

2) Differential Power Attack (DPA): In DPA attacks [4], the attacker selects a small portion of the key (i.e., 8-bits for AES), divides the traces in two sets for 256 hypothetical key values based on a single bit at the output of the SBOX and selects the key belonging to the two sets where the mean difference between them is the highest. This process is repeated until the full key is recovered.

3) Correlation Power Attack (CPA): Correlation power side channel attacks [14] work as follows. The attack on AES starts similarly as DPA, but instead of creating two sets based on single bit at the output of the SBOX, the used key is estimated using the Pearson coefficient correlation, which is computed using Equation 1. In the equation, $h_{k,i}$ represents the hamming weight/distance of the i^{th} intermediate operation (e.g., SBOX in AES, square and multiply in RSA), k the subkey value of the encryption/decryption execution, $t_{k,j}$ the sample point j within the sub-trace k, and n the number of traces. In asymmetric algorithms, the key is recovered in a bit-by-bit, in contrast to symmetric algorithms where this is determined by the width of the SBOX output (which equals 8 for AES).

$$r_{i,j} = \frac{\sum^{n}(h_{k,i} - \mu_{h_i})\dot{(}t_{k,j} - \mu_{t_j})}{\sqrt{\sum^{n}(h_{k,i} - \mu_{h_i})^2 \dot{\sum}^{k}(t_{k,j} - \mu_{t_j})^2}} \qquad (1)$$

4) Collision Power Attack: Collision attacks [15] aim at situations where two encryptions with different inputs and an unknown key will produce the same intermediate values (e.g. hamming weight/distance). If an adversary can identify from the power consumption two encryption operations that contain this occurrence, the collision can be exploited. Since a collision only exists for a subset of the potential key space, each successful collision allows the attacker to narrow the key search space.

5) Zero Value Attack: Zero value attacks [16] take advantage of the fact that known plaintext (e.g. setting it to zero) will lead to the leakage of information that exposes the secret key. These type of attacks mainly target implementations that contain countermeasures and aim to remove the randomization generated by those countermeasures. An example of such an attack can be found in multiplicative masking where zero input values cannot be randomized using multiplicative mask [18].

6) Machine learning Attack: Machine learning attacks [17] use the leakage model to distinguish between traces and derive the key from them. These attacks mainly target asymmetric algorithms such as RSA and ECC. For example, k-means is a clustering algorithm [17] that is commonly used to apply such attacks. Starting with an initial guess/prediction, it splits the training set into k distinct clusters. For each collected trace, it iteratively detects the nearest cluster center (centroid) and updates the centroids based on the mean of all training instances assigned to it until no changes occur anymore. To put it another way, the aim is to discover a partitioning that minimizes the total cluster variance. To determine the distance between two traces, the squared Euclidean distance can be used. Once the clusters are created, the partial traces that belong to the two clusters represent either a square or multiplication operation. Once these operations have been defined, the key can be recovered in a bit-by-bit fashion [17].

B. Profiled Attacks

In contrast to non-profiled attacks, in a profiled attack an adversary uses a similar or identical device under his control to create a leaking template known as the **profiling phase**. After that, the attacker correlates the power traces of the target device and compares them with the template to recover the key; this phase is also known as **extraction phase**. Both phases are explained next.

C. Profiling Phase

In this step, the adversary uses a similar or identical device that he or she completely controls to develop a behavioral model of the targeted device. This phase consists of the following steps:

Step 1: In this step, the adversary looks for a device that behaves in a similar way as the target device.

Step 2: In this step, the adversary selects and defines the point of attack (e.g., the output of *SubByte* operation in AES algorithm or *square* and *multiply* functions in RSA).

Step 3: In this step, the adversary measures several power traces of the chosen target point of attack.

Step 4: In this step, the adversary assigns a label to each trace acquired in Step 3. Depending on the cryptographic methodology employed, the label can be computed in a variety

of ways. The hamming weight/distance of the SBOX output is the most commonly used label in AES algorithm [6]. Asymmetric algorithms, on the other hand, often use the main operations as labels, e.g., *square* and *multiply* in RSA [19].

Step 5: Finally, The adversary designs/builds a template to characterize the traces. The model is build from the traces and labels collected in Steps 3 and 4, respectively.

D. Extraction Phase

During this phase, the adversary attempts to extract the secret information from the target device by applying the steps below:

Step 1: in this step, the adversary locates the device of attack.

Step 2: in this step, the adversary identifies the point of attack, i.e., the operation used to extract the labels during the profiling phase. For instance, the output of SBOX function in AES.

Step 3: in this step, the adversary measures several power traces that contain the point of attack. This step requires the traces to be sliced when asymmetric algorithms are used (e.g., slicing *square* and *multiply* operations in RSA). Slicing is not required for symmetric algorithms; there each power trace is represented with a single label.

Step 4: in this step, the adversary guesses the label value of each measured trace of Step 3. For AES algorithm, the labels can be seen as the results of the hamming weight/distance, while for asymmetric algorithm the labels represent the executed function (e,g., *square* and *multiply* in RSA).

Step 5: Finally, the adversary derives the secret key from the obtained labels. The key is retrieved in a bit-by-bit fashion when asymmetric algorithms from the identified operations of the previous step. The returned bits must be concatenated from left to right or right to left, depending on the methodology employed to recover the whole key. Symmetric algorithms need an additional steps, as the leakage model results (e.g., hamming weight) must be converted to a sub-key value. This additional step is depicted in Algorithm 1. The subkey is obtained after calculating the likelihood of key values. To predict a $subkey$ value, the likelihood of all potential subkey scenarios from $subkey = 0$ to 255 are evaluated by computing the leakage model results for each plaintext/ciphertext.

There are many examples of profiled attacks both for symmetric and asymmetric algorithms. In this section we selected the most famous ones (i.e., template based attacks (TBA) [20], machine learning Attacks(ML-SCA) [21], and deep learning based side channel attacks (DL-SCA) [6]). Note that there are many variations proposed. Next each of them will be briefly described.

1) Template-based Attack: In this attack, the *multivariate normal distribution* is used to create a profile. The profile consists of multiple covariance matrices C and mean vectors m of the points of interest of the collected power traces. First, the measured traces are grouped based on their Hamming weight/distance (HW/HD) value. Next the covariance and mean are computed for selected samples (i.e., the points of interest) within the traces for every HW/HD group. They are identified by C_h m_h for HW/HD with value h.

During the attack phase, the adversary uses the probability distance to correlate measured power traces with the profile.

Algorithm 1 Symetric Algorithms: Key Extraction

procedure KEY_EXTRACT($Prediction_{set}, pt_{array}$)
  $P_k[0, 255]$ = key probability
  $Prediction$ = the results of the trained model on the attack traces
  pt = is the plaintext used in the encryption process.
  for each sub-key **do**
    $P_k[0, 255] = 0$
    for j in trace-set **do**
      $X_{0,255} = predict(trace)$
      for k=0 to 255 **do**
        $HW_k = HW(SBOX[pt[j] \bigoplus k])$
        $P_k[k] = P_k[k] + log(Prediction_j[HW_k])$
      end for
    end for
    $guess_{subkey} = max(P_k)$
  end for
end procedure

This is shown in Equation 2. In the equation, h denotes the template number (i.e., the corresponding HW/HD set) and t an attack trace. The value of the leaking model is determined by the template that produces the highest results. Note that the traces used for attack must be aligned with the traces used for profiling.

$$f(t) = \frac{1}{\sqrt{(2\pi)^n \times det(C_h)}} \times exp(-\frac{1}{2} \times (t - m_h)' \times C_h^{-1} \times (t - m_h)) \quad (2)$$

2) Machine Learning Attack: In machine learning (ML) attacks [21], the multivariate normal distribution is replaced by ML techniques such as Support Vector Machine (SVM). SVM is a binary classifier. First a feature selection method is used to reduce the dimension (i.e., trace length) of the power trace. Thereafter a classifier is used to learn the features. During the profiling phase, the classifier creates two hyperplanes in a high-dimensional space with the goal of classifying the data. The data separation takes place in such a way that the hyperplanes are furthest away from each other. During the extraction phase, the classifier is used to classify the attack traces based on the distance to both planes. The percentage of correct classifications among the power traces from the test sets are used to determine the success rates. Note that SVMs are designed to perform a binary classification and hence can be used in three ways to perform an attack on symmetric algorithms [22]: (1) separate the results of hamming weight/distance to two groups (i.e., less than or greater than 4), (2) separate the results of the hamming weight/distance based on even or odd, and (3) separate the results of hamming weight/distance based on the value of the fourth least significant bit.

3) Deep Learning: During the profiling phase, the attacker builds and trains a neural network. The attacker must first specify the neural network's structural parameters (such as depth, width, and activation function). After that, training is performed on traces that have labels attached to them. The attacker separates the dataset (i.e., traces and their labels) into

a training set (usually 80 percent to 90 percent of the total dataset) and a validation set. The attacker ends the training when the training and validation accuracy is high enough. In extraction phase, the attacker applies the traces collected from the target device to the trained neural network. The results obtained from the neural network are subsequently used to extract the key.

III. Side Channel countermeasures

Several countermeasures to power attacks have been suggested over the last two decades. As shown in Figure 2, these countermeasures can be classified based on two metrics: technique (i.e,. obfuscation and balancing) and implementation level (i.e., software, hardware architecture, circuit/implementation, and technology). Next, the different countermeasures will be discussed based on their technique.

A. Obfuscation

Countermeasures based on obfuscation attempt to randomize the power behaviour irrespective of the performed operation. There are many examples of such techniques at different implementation levels available in the literature. At software level, one of the famous examples of obfuscating the power consumption for mainly symmetric algorithms is using masking [23]. Masking works by splitting the algorithm calculations' sensitive intermediate operations into $d+1$ random shares in such a way that analyzing d shares reveals no information about the secret value. Other examples of software level obfuscating are random order execution [24], random delay insertion [25], message and/or exponent blinding [19] and SBOX confusion [26]. In [24], random instructions with random register accesses are inserted between the original instructions sequence of the encryption/decryption process, which changes the power behaviour each time. In [25], the power behaviour is altered by inserting random NOP instructions which causes misalignment in the power traces. In [19], the key and/or the message of asymmetric algorithms are randomized in each execution. In [26], the SBOX of AES algorithm is implemented using a neural network which confuses the power behaviour of the leakage model. Note that this countermeasure, unlike the others, targets the leakage model. In hardware, similar to software, masking [27] is the most popular countermeasure. Another example of a hardware based countermeasure is random delay insertion [28], where the delay is inserted by logic gates. Note that the other software level techniques can be also implemented in hardware. At circuit level, the power consumption can be obfuscated by modifying the logic cells as is the case for masked dual-rail pre-charged logic [29] or by having an additional source in the system to injected noise [30]. At technology level, emerging devices such as memristors can be used for obfuscation by exploiting cycle-to-cycle variation.

B. Balancing

The goal of balancing techniques is to keep the power usage as stable as possible during sensitive operations. Similarly to obfuscation techniques, there are many examples of such techniques studied in the literature at every implementation level (i.e., software, hardware architecture, circuit, and technology). One of the famous countermeasures in software is Montgomery multiplication [31] where both operations of asymmetric algorithms (e.g., square and multiply in RSA and double and addition in ECC) are executed in the Montgomery domain. This results in a similar power behaviour for both operations. Hence, it is harder for an attacker to distinguish between them. Another example of a balancing technique at software level for asymmetric algorithms is multi-bit blinding [32]. This technique always executes the same sequence of operations regardless of the key bit values, by considering two bits at a time and re-order their operations. For symmetric algorithms, a multi-core can be used [33] where two encryptions are executed on different cores simultaneously. One with original message while the other one with its complementary message. At the hardware level, the same techniques used in software can be implemented. A clear example can be seen in the duplicate design [34], where instead of having two software encryptions, two actual hardware implementations are used to run the message and its complementary at the same time. At circuit level many techniques were proposed such as *power equalizer* [35], *dual-rail logic* [36], and *Adiabatic Logic* [37]. In *power equalizer* [35], the power is balanced using the on-chip power supply. In *dual-rail logic* [36], the power is balanced by redesigning logic cells such that they take both the input and their complement values as inputs. In *adiabatic logic* [37], the power is balanced by designing CMOS cells in such a way tha they both charge and discharge at the same time to disguise power irregularities. At technology level, researchers are exploring emerging technologies such as Memristor [38] to minimize the power leakage, which increases the attack difficulty. Note that circuit and technology level techniques can be applied to both symmetric and asymmetric algorithms.

IV. Leakage Assessment Styles

There are currently several options to evaluate countermeasure implementations. They can be grouped into three categories based on their style: evaluation-style, conformance-style, and formal style. Each style is briefly explained next.

A. Evaluation Style

In evaluation-style testing, power traces are tested using actual side channel attack scenarios, such as those described in Section II. They show whether the implementations are resistant to such attacks or not. The attacks can be performed in a profiled or unprofiled manner as discussed in Section II. The attacks can be performed after the chip is manufactured using off-the-shelf security tools and equipment (e.g., equipment of Rambus [39] and Riscure [40]) or during the design process using CAD-based solutions [41].

B. Conformance Style

On the other hand, conformance-style testing examines whether traces are compliant with specific leakage criteria without taking actual attacks into account. Test Vector Leakage Assessment (TVLA) [42] and signal-to-noise ratio (SNR) analysis [43] are two examples of this form of analysis. Due to space limitations, we focus only on TVLA.

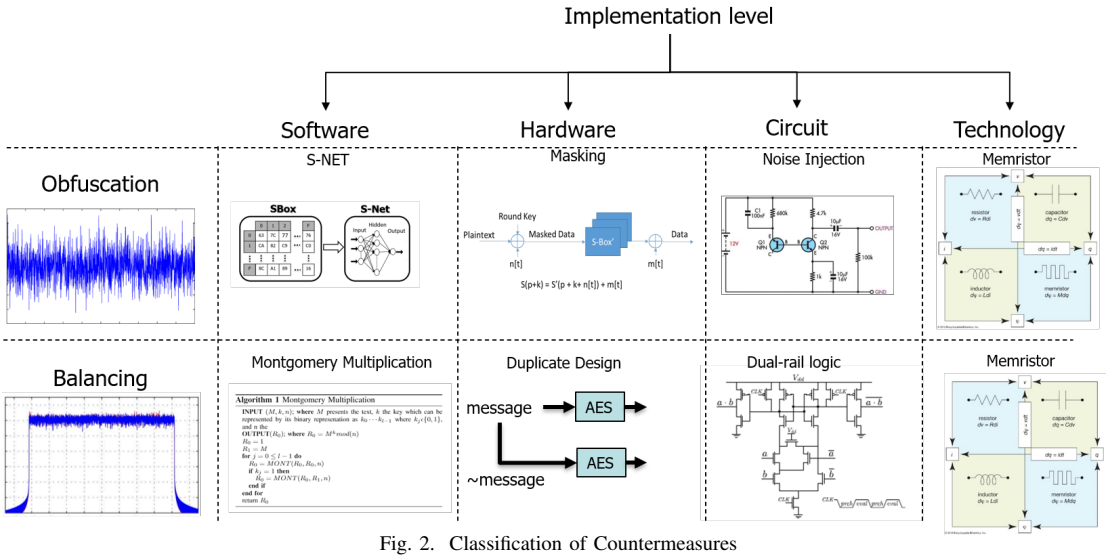

Fig. 2. Classification of Countermeasures

TVLA is based on Welch's t-test, which examines whether two populations have similar distribution. Welch's t-test is used in to identify whether power traces of an encryption/decryption algorithm execution leak information about the secret key. The leakage is measured using two sets of power traces, one with fixed plaintext/ciphertext and the other with random plaintext/ciphertext. Note that the key value is the same in both sets. Equation 3 shows the equation used to perform this test. In the equation, $\bar{X}_1$, S_1^2, and N_1 represents the mean, the variance, and the total number of used *fixed plaintext/ciphertext* traces, respectively, while $\bar{X}_2$, S_2^2, and N_2 represents the mean, the variance, and the total number of used *random plaintext/ciphertext* traces, respectively.

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}} \tag{3}$$

C. Formal Style

The aim of formal verification-based testing is to analyze the leakage of an implementation mathematically. Formal verification examples can be found in [44, 45]. In [44] the authors use formal verification to verify hardware masking countermeasures. In [45], the authors present a satisfiability modulo theories (SMT) solver to evaluate software masking countermeasures.

V. Discussion and Conclusion

This paper presented a short survey of on power side channel attacks, proposed countermeasures and leakage assessment styles. Based on our study, we conclude the following.

Power attacks: Researches are continuously improving power side channel attacks and explore ways to enhance the attack resolution of current attacks. For example, the use of Generative Adversarial Networks [46] as a method to enhance the profiling phase by producing more power traces without the need to measure them is one of the most recent proposed techniques. Also new developments are proposed for non-profiled attacks. One example is the employment of deep learning approaches like using auto-encoders [47] as a processing tool to improve accuracy. Note that the development of attacks is not limited to attack analysis techniques; measuring and accessing the target device are also being looked at for more realistic circumstances. For example, the authors in [48] propose a remote side channel attack by accessing the on-chip analogue-to-digital converter.

Limitations of countermeasures: *Software countermeasures* typically come with a large performance overhead. The obfuscation based randomization methods have a limited impact and are complex to implement, while it is very difficult to create balancing based countermeasures. *hardware design countermeasures* perform slightly better than software countermeasures in terms of security and performance. However, they increase the area of the design which is not always affordable, as is the case for constraint IoT devices. In addition, they are difficult to debug in case a vulnerability is found. *Circuit level countermeasures* require special skills and experience for proper implementation. In addition, they typically increase the area and the power consumption (except for adiabatic logic which still lacks a proper security analysis). Finally, *technology level countermeasures* are not well explored as they mostly depend on immature emerging technologies. Moreover, they are costly to implement and require special skills.

Limitations of Leakage assessment: Unfortunately, it is impossible to evaluate a design against all the existing attacks using evaluation-style testing; furthermore it is also impossible to predict future attacks. Performing attacks with post-manufacturing traces is the most accurate way to do this and in case the security is not satisfied, the chip has to be redesigned and re-manufactured, affecting the cost and the design time considerably. CAD-based solutions, on the other hand, can solve the post-manufacturing by generating traces during the design stage already by using simulations for example. However, this is a time-consuming task and the traces are not as accurate as real ones. Conformance-style leakage assessment methods (i.e., TVLA and SNR) provide a fast and easy way of assessing the power traces.

Unfortunately, these methods have been proven that they are not very accurate [49]. Formal style testing is a new way of performing a pre-manufacturing power leakage assessment. However, this methods works only for a single countermeasure technique (i.e., masking). Furthermore, when the number of mask shares increases, the formal verification's worst-case time complexity increase exponentially.

ACKNOWLEDGMENTS

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

REFERENCES

[1] M. R. Mayhew, M., "An overview of hardware-level statistical power analysis attack countermeasures." 2017.
[2] M.-C. Hsueh, T. Tsai, and R. Iyer, "Fault injection techniques and tools," *Computer*, 1997.
[3] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology – CRYPTO*, 1996.
[4] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology – CRYPTO*, M. Wiener, Ed., 1999.
[5] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *2009 IEEE International High Level Design Validation and Test Workshop*, 2009.
[6] H. Maghrebi, "Deep Learning based Side Channel Attacks in Practice," *IACR Cryptol. ePrint Arch.*
[7] J. Danial *et al.*, "Scniffer: Low-cost, automated, efficient electromagnetic side-channel sniffing," 2020.
[8] F. Koeune and F.-X. Standaert, "A tutorial on physical security and side-channel attacks," in *FOSAD*, 2004.
[9] V. Lomné *et al.*, "Side channel attacks," 2011.
[10] L. Zhang, L. Vega, and M. Taylor, "Power side channels in security ics: Hardware countermeasures," 2016.
[11] K.-y. Chen *et al.*, "A Survey of Side-Channel Attack and Security Assessment for Cryptographic Equipment," *PoS*, 2017.
[12] G. S. . G. T. Hettwer, B., "Applications of machine learning techniques in side-channel attacks: a survey," 2020.
[13] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, 2020.
[14] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES*, 2004.
[15] K. Schramm *et al.*, "A collision-attack on aes," in *Cryptographic Hardware and Embedded Systems - CHES*, M. Joye and J.-J. Quisquater, Eds., 2004.
[16] T. Akishita and T. Takagi, "Zero-value point attacks on elliptic curve cryptosystem," in *Information Security*, 2003.
[17] J. Heyszl *et al.*, "Clustering algorithms for non-profiled single-execution attacks on exponentiations," in *Smart Card Research and Advanced Applications*, 2014.
[18] J. D. Golić and C. Tymen, "Multiplicative masking and power analysis of aes," in *Cryptographic Hardware and Embedded Systems - CHES*, 2003.
[19] M. Carbone *et al.*, "Deep learning to evaluate secure rsa implementations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019.
[20] S. Chari, J. R. Rao, and P. Rohatgi, "Template Attacks," *CHES*, 2003.
[21] L. Lerman, G. Bontempi, and O. Markowitch, "Power analysis attack: an approach based on machine learning," *Int. J. Appl. Cryptogr.*, 2014.
[22] G. Hospodar *et al.*, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, 2011.
[23] M. Masoumi, P. Habibi, and M. Jadidi, "Efficient implementation of masked aes on side-channel attack standard evaluation board," in *2015 International Conference on Information Society (i-Society)*, 2015.
[25] J.-S. Coron and I. Kizhvatov, "Analysis and improvement of the random delay countermeasure of ches 2009," in *Cryptographic Hardware and Embedded Systems, CHES*, S. Mangard and F.-X. Standaert, Eds., 2010.
[24] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "A smart random code injection to mask power analysis based side channel attacks," in *Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, 2007.
[26] A. Aljuffri *et al.*, "S-net: A confusion based countermeasure against power attacks for sbox," in *Embedded Computer Systems*, 2020.
[27] H. Maghrebi *et al.*, "A first-order leak-free masking countermeasure," in *Topics in Cryptology – CT-RSA 2012*. Springer Berlin Heidelberg, 2012.
[28] Y. Lu, M. O'Neill, and J. McCanny, "Evaluation of random delay insertion against dpa on fpgas," *ACM Trans. Reconfigurable Technol. Syst.*, 2010.
[29] T. Popp and S. Mangard, "Masked dual-rail pre-charge logic: Dpa-resistance without routing constraints," in *Cryptographic Hardware and Embedded Systems – CHES 2005*. Springer Berlin Heidelberg, 2005.
[30] L. Zhang, L. Vega, and M. Taylor, "Power side channels in security ics: Hardware countermeasures," *ArXiv*, 2016.
[31] H. Nozaki *et al.*, "Implementation of rsa algorithm based on rns montgomery multiplication," in *Cryptographic Hardware and Embedded Systems — CHES*, 2001.
[32] A. Aljuffri *et al.*, "Multi-bit blinding: A countermeasure for rsaagainst side channel attacks," in *IEEE VLSI Test Symposium*, 2021.
[33] Z. Chen and P. Schaumont, "Virtual secure circuit: Porting dual-rail pre-charge technique into software on multicore," *IACR Cryptol. ePrint Arch.*, 2010.
[34] M. Doulcier-Verdier *et al.*, "A side-channel and fault-attack resistant aes circuit working on duplicated complemented values," in *2011 IEEE International Solid-State Circuits Conference*, 2011.
[35] C. Wang *et al.*, "Power profile equalizer: A lightweight countermeasure against side-channel attack," in *2017 IEEE International Conference on Computer Design (ICCD)*, 2017.
[36] J.-L. Danger *et al.*, "Overview of dual rail with precharge logic styles to thwart implementation-level attacks on hardware cryptoprocessors," in *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*, 2009.
[37] H. Thapliyal, T. Varun, and S. D. Kumar, "Adiabatic computing based low-power and dpa-resistant lightweight cryptography for iot devices," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2017.
[38] M. Masoumi, "Novel hybrid cmos/memristor implementation of the aes algorithm robust against differential power analysis attack," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.
[39] Rambus, "DPA Workstation Platform," Available at: https://www.rambus.com/security/dpa-countermeasures/dpa-workstation-platform/, accessed: 2021-05-08.
[40] Riscure, "Inspector Side Channel Analysis," Available at: https://www.riscure.com/security-tools/inspector-sca, accessed: 2021-05-08.
[41] A. Nahiyan *et al.*, *CAD for Side-Channel Leakage Assessment*, 2021.
[42] G. Becker *et al.*, "Test Vector Leakage Assessment (TVLA) methodology in practice," Available at: https://pdfs.semanticscholar.org/, 2011, accessed: 2019-09-23.
[43] S. Mangard, "Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness," in *CT-RSA*, 2004.
[44] R. Bloem *et al.*, "Formal verification of masked hardware implementations in the presence of glitches," in *Advances in Cryptology – EUROCRYPT 2018*, 2018.
[45] H. Eldib, C. Wang, and P. Schaumont, "Formal verification of software countermeasures against side-channel attacks," 2014.
[46] P. Wang *et al.*, "Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks," *IACR Cryptol. ePrint Arch.*, 2020.
[47] D. Kwon, H. Kim, and S. Hong, "Improving non-profiled side-channel attacks using autoencoder based preprocessing," *IACR Cryptol. ePrint Arch.*, 2020.
[48] M. Martínez-Rodríguez, I. M. Delgado-Lozano, and B. B. Brumley, "Sok: Remote power analysis," Cryptology ePrint Archive, Report 2021/015, 2021.
[49] "Leak me if you can : Does tvla reveal success rate ?" 2017.