The ISC Int'l Journal of Information Security
ISeCure

January 2022, Volume 14, Number 1 (pp. 47–55)
http://www.isecure-journal.org

A Time Randomization-Based Countermeasure Against the Template Side-Channel Attack **

Farshideh Kordi [1], Hamed Hosseintalaee [1], and Ali Jahanian [1]
[1] *Shahid Beheshti University Faculty of Computer Science and Engineering Tehran, Iran.*

ARTICLE INFO.

Article history:
Received: December 19, 2020
Revised: May 19, 2021
Accepted: June 12, 2021
Published Online: September 6, 2021

Keywords:
Side-Channel Attacks, Template Attack, Time Shuffling Countermeasure, Profiled Attacks

Type: Short Paper

doi: 10.22042/isecure.2021.262658.592

dor: 20.1001.1.20082045.2022.14.1.4.4

ABSTRACT

The template attack is one of the most efficient attacks for exploiting the secret key. Template-based attack extracts a model for the behavior of side channel information from a device that is similar to the target device and then uses this model to retrieve the correct key on the target victim device. Until now, many researchers have focused on improving the performance of template attacks, but recently, a few countermeasures have been proposed to protect the design against these attacks. On the other hand, researches show that regular countermeasures against these attacks are costly. Randomized shuffling in the time domain is known as a cost-effective countermeasure against side-channel attacks that are widely used. In this article, we implemented an actual template attack and proposed an efficient countermeasure against it. We focus on the time shifting method against template attack. The results show that template attack is very susceptible to this method. The performance of attack on an AES algorithm is considerably reduced with this method. We reported the analysis results of our countermeasure.The performance of the attack can be determined according to various criteria. One of these criteria is the success rate of the attack. According to these results, template attack will be hardened significantly after the proposed protection such that the grade of the key recovery increases from 1 with 350K traces in unprotected design to 2^{100} with 700K traces in the protected circuit. This security improvement gains in the cost of about 7% delay overhead.

1 Introduction

Executing an encryption algorithm may leak side-channel information such as power, electromagnetic, time, etc. Side-channel attacks try to break cryptosystems by exploiting this information, which is leaked by the hardware implementation. Side-channel attacks (SCA) are known as serious threats to modern cryptographic implementations and protection against SCA is a critical threat to the security of embedded systems [1]. Generic SCA algorithm consists of two steps; identifying leaked information and creating a model that fits this information. An important SCA branch is power attacks that use power traces as leaked information to mine the secure data inside a system. The power leakage is made of two parts, dynamic and static power. Dynamic power is a widely

* Corresponding author.
**This article is an extended/revised version of an ISCISC'17 paper.
Email addresses: `f.kordi@mail.sbu.ac.ir`, `hamed.talaie8@gmail.com`, `jahanian@sbu.ac.ir`
ISSN: 2008-2045 © 2020 ISC. All rights reserved.

used method as a source of power attacks.

Power side-channel attacks categorize as *Profiled* and *Non-profiled* categories. The only assumption for non-profiled attacks is that the attacker must have a similar device to collect power consumption. Non-profiled attacks are a wide range of methods such as SPA, DPA and CPA [2]. The simplest method for this category is SPA in which attackers observe power consumption for a key-related operation without considering any statistical method [2]. CPA is an algorithm used to do power analysis that categorize as a statistical attack. In this attack, the Pearson correlation coefficient uses to extract the maximum dependency between the key and the data. In this attack is assumed that hypothetical key is correct when the correlation between the Hamming weight model for this key assumption and the real power consumption of all hypothetical keys is most significant [3].

Another powerful category of power attack is profiled attack that precisely models the system's noises as well and uses this model for the attack. In this kind of attack, as mentioned, accessing to a device similar to the attacking device is assumed [4]. Profiled attacks include template, stochastic and machine-learning-based attacks. Template attack (TA) was first offered by Chari *et al.* in 2002. This method extracts the most helpful information from each power consumption trace [5] and [6]. TAs consist of two steps; *Profiling* and *Attacking* steps. In the profiling step, an attacker needs the same device to build key-dependent templates. In the attacking step, the probabilities of key-dependent noise are calculated for each power trace captured from the targeted device. Then, the secret key can be disclosed by estimating the maximum likelihood method [7]. In the TA method, like other profiled attacks, a precise noise model is extracted from the power consumption traces. Typically, the Gaussian model is considered a noise model [8]. Stochastic attack (SA) is a type of TAs that is in the profile stage instead of the Gaussian model uses linear regression [9]. The main problem with implementing TA is the scale of the covariance matrix, which is a factor of the number of sampling points. To mitigate this problem, A subset of sample points (points of interest or POI) must be chosen [10]. Points of interest are those points that contain the most key dependency. Since TA is hard to defend compared to other power SCA, this paper analyzes the TA. Countermeasures against the SCAs are constituted of hiding and masking in software implementations. The masking method is an effective countermeasure against SCAs that makes unpredictable the intermediate values of prevention dependencies between these values and the power consumption. Intermediate values are covered by random values for every execution. In the masked implementation, the security is related to the randomness degree of the masks [11]. The Higher-Order DPA attacks can break the masking technique [12].

The hiding method consists of shuffling the order of the operations or inserting random delays with dummy operations to hide the relationship between the power consumption and secret data [13]. This paper is an extension of work originally presented in ISCISC [14]. Randomized shifting in the time domain is prposed as an effective countermeasure against template attacks in this article. This countermeasure makes it almost impossible to select POIs. Based on the disadvantages of TAs, the attack will be so tricky because we have to select all the sample points. This article consists of the following sections. Section 3 provides a brief description of how TA attacks are performed. Section 4 describes the proposed countermeasures, and Section 5 includes an analysis of the experimental results. Finally, in section 6 a conclusion of the article is presented.

2 Related Work

Mangard *et al.* [11] have researched that TAs can defeat protected algorithms by the masking method. According to this article, masking does not improve implementation security in the template-based attack. The reported results confirm that the template-based DPA attack is broken the masked implementation with 15 power traces in the practical experiment.

Masking usually implies significant performance overheads. The easiest method of defense against such attacks is inserting random delays. Inserting random delays is categorized as one of the Hiding countermeasures methods. Hiding countermeasures propose lower security than masking countermeasures but their implementation offers lower costs [3]. By inserting random delays, the alignment of the traces are broken, thus increasing the number of traces needed for the attack. The misalignment of the traces can be a powerful countermeasure against SCA [4].

In another work by Mangard *et al.* was described as a combination of masking and randomization methods to resist the TAs and the higher order attack. Increased resistance means that for a successful attack, more traces need to be collected [15]. Another countermeasure against TAs was suggested by Barenghi *et al.* [16]. In this paper, an architectural countermeasure against profiled attacks has presented that differentiates the behavior of leaked information for similar devices, thus disrupting the reuse of an extracted model for a similar one.

An essential assumption in a profiled attack is that the leaked information behavior extracted of a sample device prepares a suitable descriptions for the behavior of another sample of a similar device. This method prevents the possibility of reusing the profile obtained from one device for another similar one. However, it does not prevent profile access and a successful attack on the primary devices [16].

Authors of [17] mentioned that the performance of the TA is highly influenced by factors such as misalignment of traces, a large amount of data and selection of interesting points. In [17], a method based on Convolutional Neural Networks was proposed for profiled attacks. Deep learning techniques are introduced as an effective type of profiled attacks. SCA-based deep learning is similar to TA but it uses deep learning techniques to find the maximum likelihood instead of using multivariate Gaussian distributions [18]. Most countermeasures are designed to protect the design against classic SCAs and cannot protect the design from deep learning attacks.

In [18], a countermeasure is proposed against side-channel attacks based on deep learning. It should be noted that the proposed countermeasure can also increase the resistance of classical side-channel attacks. In this paper, the noise instructions is inserted into the code as countermeasure. An important difference with other countermeasures is the selection of more appropriate noise inserting instructions and identification of the accurate location of the noise [18].

As mentioned earlier, TA has been introduced as a highly beneficial method in the category of SCAs. In this paper, we focused on protecting the AES algorithm against template attacks by hiding countermeasures. However, time randomization has been widely used for protecting encryption algorithms against DPA and CPA. However, analysis of the impact of this method on TAs has not been reported, while analyses show that real implementation of TAs (e.g. using the interesting points) is very sensitive to any time-shifting. For this purpose, we propose an efficient random time shuffling on critical subset points of the AES encryption algorithm (that are selected randomly) to filter the interesting points for obtaining successful TAs. We will validate the effectiveness of countermeasures against TAs. Our analyses lead to important conclusions about the function distribution of the mean vector and points of interest.

3 Template Attack

We appreciate that you have selected our journal for publishing your paper. We set up some rules to standardize the papers we receive in order to make it easier for the author and the editor. Please consider these rules in writing your manuscript.

3.1 Profiling Stage

At this step, the attacker collects a large amount of power consumption traces from a similar device to make the profiles. The Hamming Distance (HD) regularly examines the relationship between power consumption and bit switches from 0 to 1 or 1 to 0. The Hamming Weight (HW) is the number of “1” bits in the binary sequence. On the other hand, based on the microcontrollers’ assumption, the HW and HD models are almost equal [19]. The S-Box output can be used to build 256 classes directly. This method is more recommended because each of these 256 classes is related to the information of each guessed key directly [20]. In this paper, we build templates according to the HW model with power traces to improve the success rate of the TAs. As a result, 9 templates is built in this step. The S-Box output of the first round of the AES algorithm is usually chosen to collect the power traces because the S-Box is a non-linear function within an AES encryption algorithm. The HW of the S-Box output byte is calculated as:

$$H = HW(S(p \oplus k)) \tag{1}$$

S is the S-Box, p is plaintext, and k is a corresponding key in the AES encryption algorithm.

The templates are built based on hamming weight $(T_i) = \{T_0, T_1, \ldots, T_8\}$ in order to characterize the device. In template attacks, the total number of $N \times M$ states may occur which N is the number of traces and M is the number of sample points that are used. Each sample includes two parts, signal and noise. According to the result of HW, each power trace is mapped to one template. Consequently, 9 templates are built because an 8 bit may have HW 0 to 8.

Each template consists of a mean vector $(\overline{\mu})$ and a covariance matrix $(\mathbf{C})$. The $\overline{\mu}$ vector and the covariance matrix C determine the noise model for all profiles. Vector $\overline{\mu_i}$ is defined as the mean vector of the traces belongs to template i. It can be calculated as:

$$\overline{\mu_i} = \frac{1}{n_i}\sum_{j=1}^{n_i}\overline{t_{i,j}} \quad \forall\ i \in 0,1,\ldots,8 \tag{2}$$

where $\overline{t_{i,j}}$ is j^{th} power trace vector of the template i and n_i is the number of traces in an i^{th} template. The covariance matrix of an i^{th} template $(\mathbf{C_i})$ is defined as:

$$C_i = \frac{1}{(n_i - 1)}\sum_{j=1}^{n_i}(\overline{t_{i,j}} - \overline{\mu_i})^T(\overline{t_{i,j}} - \overline{\mu_i}) \quad \forall\ i \in 0,1,\ldots,8 \tag{3}$$

Accordingly, positive covariance shows that both dimensions increase together. Negative covariance means that as one dimension increases, the other dimension decreases and zero covariance indicates that two dimensions are independent of each other.

Eventually, the attacker can create a multivariate Gaussian noise model according to the mean vector and the covariance matrix per template.

3.2 Attacking Stage

At this stage, the key is recovered with a small amount of traces that is collected from the device under attack. The maximum likelihood methods determine the correct key bytes. A multivariate normal distribution model can be extracted using the two parameters of mean and covariance obtained from the previous step.

First of all, the new power trace is given to be identified, then the attacker guesses a key-value by following methods.

- Considering a two-dimensional matrix based on the number of traces of the attacking step and guessed keys.
- The probability density function is calculated according to each guess of the keys and the plaintext for each template. Then, the probability density function (PDF) is inserted into the previous step matrix corresponding to its plaintext.
- Finally, the sum of the matrix columns is computed and one vector with 256 elements is generated based on the guessed key.
- The best guess key is the maximum value of the vector in the previous step.

The PDF function is calculated as follow:

$$f_{i,j}(\overline{t_{i,j}};(\overline{\mu_i},C_i)) = \frac{1}{\sqrt{(2\pi)^n \det(C_i)}} exp\left(-\frac{1}{2}(\overline{t_{i,j}}-\overline{\mu_i})^T (C_i)^{-1}(\overline{t_{i,j}}-\overline{\mu_i})\right) \tag{4}$$

The attacker approximate the secret key which maximizes the likelihood:

$$\overline{k} = argmax f_{i,j}(t_{i,j}|k) \tag{5}$$

The number of sample points is significant in TA attack and a wide range of samples lead to excessive computational loads. For building templates in the first experiment, 350000 traces are recorded with random input plaintexts with random keys. The second set of traces is a small number (approximately 50). We calculate and sort the value of the probability distribution function for all the hypothetical key values and determine the grade of the key recovery based on these sorted values.

3.3 Feature Selection

Points of interest (POI) are those points that give the maximum correlation between the power model and power traces. The advantages of applying POIs are as follows:

- It decreases the number of sampling points per trace, which leads to a reduction in calculations per profile.
- Numerical problems related to the inverse of the covariance matrices will remarkably reduce the performance of TAs. The ill-condition error occurs due to a large number of data in the inverse matrix [21].

As a result, selecting points that defines the template is an essential part of the attack. With this method, the template size will be smaller and the profiling stage will lead to better performance. There are various techniques to extract interesting points. Sum of squared differences (SOSD) and correlation power analysis (CPA) are the efficient methods. The SOST-based and Principal Component Analysis method (PCA) will lead to choosing the best interesting points [16]. According to another paper, CPA and the SOST methods will conduct to the best efficiency [10]. SOSD can be computed as follows.

$$SOSD = \sum_{i=1}^{n_i} \sum_{j=1,j\neq i}^{n_j} (\overline{\mu_i} - \overline{\mu_j})^2 \tag{6}$$

By normalizing the variance parameter in the SOSD calculation, the quality level can be increased. This normalized SOSD is called SOST that is computed as follows. SOST is calculated as:

$$SOST = \sum_{i=1}^{n_i} \sum_{j=1,j\neq i}^{n_j} \left(\frac{((\overline{\mu_i} - (\overline{\mu_j})}{\sqrt{(\frac{\delta_i}{m_i})^2 + (\frac{\delta_j}{m_j})^2}} \right) \tag{7}$$

where δ_i is the variance of template $i \in \{0, 1, \ldots, 8\}$, m_i is the number of traces for template i and $\overline{\mu_i}$ is the mean vector of this template.

To find POIs, we calculated the SOST such that the points having higher SOST values, indicate POIs and built the templates for these points. The rest of the sampling points are removed in the power traces and the templates are built based on the interesting points.

4 The Proposed Countermeasure

Randomize time-shifting in cryptographic algorithms has been proposed as an effective countermeasure against SCAs. The basic idea is to remove the data dependency of the intermediate value and the secret key. This means that the power consumption characteristic changes and the attacker cannot find the dependency between the data and the key. First of all, we have implemented the TA and we have applied it to an unprotected AES algorithm on an ARM

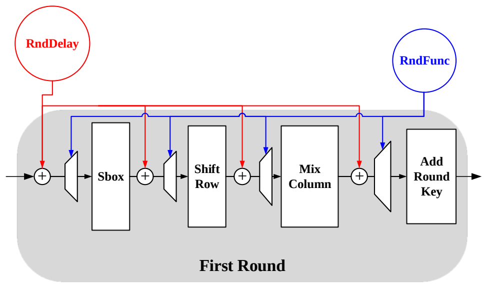


Figure 1. First Round of AES to insert random delay.

microcontroller. In this implemented attack, using a maximum of 15 power traces, we can retrieve the correct key.

In the following steps, a single net of the design is selected and its delay is shuffled randomly. Then multiple random delays are inserted on multiple random places in order to apply to the AES implementation.

In multiple random delays, two random factors are used to apply the random delay in a random place as follows.

- The first factor specifies the place where the delay should be applied it. Depending on the random value generated, the delay can be added before each of the SubByte(), ShiftRow(), MixColumn(), and AddRoundKey() functions for each AES round. RandFunc determines this value according, to Figure 1.
- The second factor determines the delay that must be added to the random location. This value is determined by the RandDly variable in Figure 1.

The rand () function is used to generate random numbers with uniform distribution, which is used twice. In the first step, a random number is produced in the range of [0, 3], which indicates the location of the delay. In the next step, a random number is produced in the range of [0, 10] indicating the amount of delay. In this implementation, there is no need to use the random number generator in the hardware. The whole implementation is done in software and this is the advantage of it. The codes for this section are shown in Algorithm 1.

The key point is that the delay length should be shortened to minimize the performance overhead. In order to achieve such a result, Having several short random delays is more effective, as recommended in [22] and [13], because finding and eliminating many short delays are more complicated than long delays. It should be noted that the length of individual delays in single and multiple delays are in the distance [0, 10]. In the single delay method, the delay is inserted before the SubByte function in AES-128 based on

Algorithm 1 Pseudocode for AES with random delay inserted.

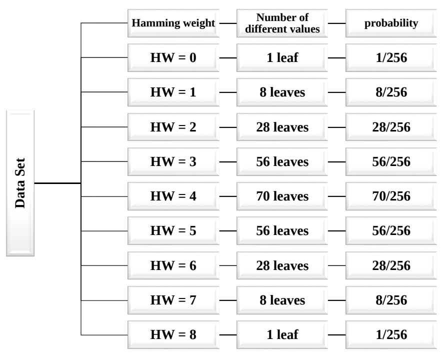
```
1: RndFunc = PRNG
2: RndDly = PRNG
3: state = M
4: for 1 through 10 do
5:     if RndFunc is 0 then
6:         RndDly
7:         SubByte(state)
8:     else if RndFunc is 1 then
9:         RndDly
10:        ShiftRow(state)
11:    else if RndFunc is 2 then
12:        RndDly
13:        MixColumns(state)
14:    else if RndFunc is 3 then
15:        RndDly
16:        AddRoundKey(state)
17:    end if
18: end for
```

random function. This countermeasure makes selecting POIs very hard and forcing the attacker to analyze the vast number of points for two steps (profiling and matching steps) of TAs. We can eliminate POIs with this method. To achieve such a result, we need a wide range of sample points for profiling and attacking phases for each trace. Additionally, we require much more power traces for building templates and successful key recovery. According to the description of POIs in section 2, TAs will be more challenging. The most used metrics for attacks and countermeasure is the number of traces for the profiling step(the value N of M sample points) [22].

5 Experimental Results

We extracted the power consumption traces on a microcontroller during an AES-128 operation. Our target device is an STM32F407 processor. Sampled data was acquired with a 300MHz and 2Gsamples/S oscilloscope. In the profiling step, significant power traces were acquired from the profiling device for the random values of plaintext and the key to building templates. In the profiling step, we have to find the value of the Hamming weight of each power trace and map to 9 templates according to Figure 2. For example, the maximum number of classes we need to control simultaneously is related to classes with a Hamming weight of 4, where 70 values are possible.

In the attacking step, A small amount of traces is used to have a successful attack. Our results show that the main parameter that influences the likelihood of success of an attack against a time shuffling technique is the number of places that might be chosen for delay shuffling. In the first step of our experi-



Data Set: Hamming weight	Number of different values	probability
HW = 0	1 leaf	1/256
HW = 1	8 leaves	8/256
HW = 2	28 leaves	28/256
HW = 3	56 leaves	56/256
HW = 4	70 leaves	70/256
HW = 5	56 leaves	56/256
HW = 6	28 leaves	28/256
HW = 7	8 leaves	8/256
HW = 8	1 leaf	1/256

Figure 2. Hierarchical of HW templates

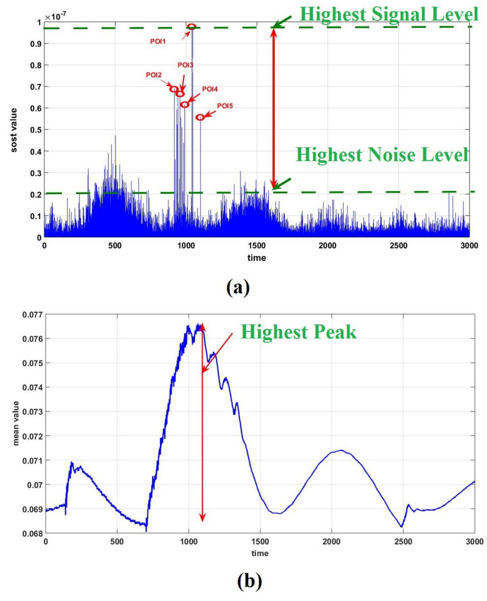


Figure 3. The result of TA against the unprotected AES (a) SOST (b) mean vector for HW=6

ment, $N = 350000$ traces each of which contains $P = 3500$ sample points are used for training (Template generation) and the number of interesting points is $N_{poi} = 5$. We applied the SOST method to pick significant points in time belong to the first round of AES. We required at most 15 traces for a successful attack against AES-128 without countermeasure.

Figure 3 depicts the statistical information of unprotected implementation. This figure comprises two sub-figures; Figure 3-a represents the SOST for TAs against AES-128 encryption without countermeasure and Figure 3-b shows the mean vector from 1 of 9 HW of the profiled templates. POIs (POI0 to POI5 in Figure 3-a) are the highest peaks in SOST. Moreover, the difference between the signal and noise is shown by the red vertical line.

Afterward, AES is protected by shuffling the delay at the point before the SubByte function. Then, we performed our attack on AES with a countermeasure. Figure 4 shows the results of the proposed method

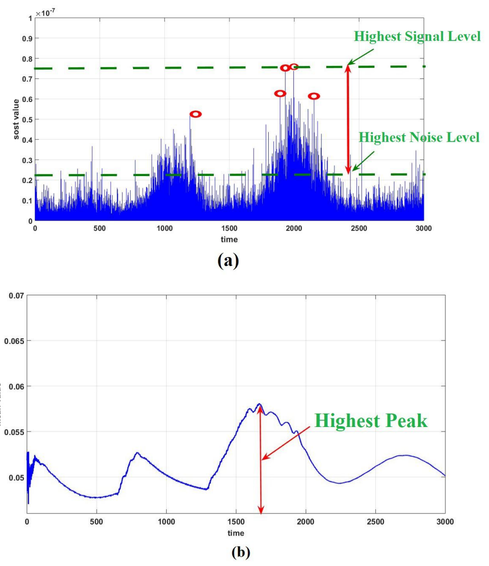


Figure 4. The TA result against the AES with the delay shuffling on one node(a) SOST (b) Mean vector for HW=6

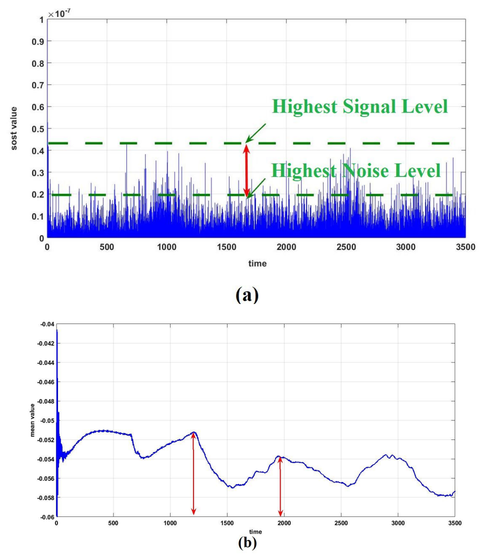


Figure 5. The result of TA against the AES with multiple random delays (a) Sum of squared t-differences (sost) for AES encryption with multiple random delays (b) mean vector from one of nine HWs template profiled

after inserting one random delay. In this experiment, the number of traces used to train is $N = 450000$ and the number of interesting points is $N_{poi} = 50$ that is shown just 5 points for example.

As shown in Figure 5, the difference between the noise level and the signal level is significantly smaller than Figure 4. The difference between SOST results for protected and unprotected implementations is getting further. This means that the protected circuit is hardened against the attack. Figure 5 represents the results of the protection method after shuffling of delay on multiple nodes.

Table 1. Success rate depending on the number of traces on protected and unprotected AES

Design	#traces	#Sample pints in a trace	The rank of The correct key
Unprotected	350000	5	1
Protected	400000	6000	$2^{106.3}$
	500000	6000	$2^{103.9}$
	600000	6000	$2^{101.1}$
	700000	6000	$2^{100.3}$

As shown in Figure 5, the noise level and the signal level are getting closer to each other because of the trace misalignment that resulted in the applied random delay shuffling. On the other hand, the plotting's shape means vector in Figure 5-b represents that applied time shuffling has a more chaotic outcome and there is no impressive general point. It shows that the correlation between the leaked information and the secret key is mitigated considerably. On the other hand, it is inferred from Figure 5 that the POIs selection methods are not helpful due to the closer difference between the signal and noise. In this situation, the attacker must consider more sample points of a trace and more trace data for a successful attack.

In the following, we try to present a quantitative evaluation of the proposed method. The amount of traces needed for the profiling stage to find the correct key(or result in an acceptable grade of the key) is a widely used metric to show the quality of protection against a specific attack [4]. Table 1 represents the number of traces and corresponding resulted in the grade of the key recovery for protected and unprotected AES algorithms. We sort all full subkeys according to the maximum likelihood method based on the probability distribution that is explained in section 2 and determine the grade of the key recovery. It is worth noting that the probability of success significantly reduces in protected implementation and it was not possible to have a successful attack.

As is seen in Table 1, the grade of the key recovery is raised exponentially in the protected design such that the successful attack is not possible practically. Moreover, by increasing the number of profiling traces, this grade does not decrease considerably (e.g. $2^{106.26}$ for 400K traces to 2^{100} for 700K traces). In this experiment, the sampling rate is 2GHz and the time per sample is $0.5ns$. Insertion of multiple delays generates time overhead that is calculated corresponding to increased points. In this method, the number of sample points is increased from 35000 to 37500. As a result, the time overhead for the time shuffling method is near 7%.

Finally, it is worth noting that for the following reasons, we have used and analyzed the time randomization method for TAs.

- Previous articles have not reported the analysis of time randomization for TAs, while TAs are very sensitive to the time-shifting method and this analysis is very informative.
- The numerical obstacles are the critical vulnerability of the TA because it leads to poorer classification performance. One of the advantages of TAs is the profiling stage. In this paper, we evaluate that we need $350,000$ profiling power traces to have a successful attack for unprotected AES. Each trace contains $35,000$ sample points per execution of AES. With such a considerable amount of data and complex computation is required about 40 to 50 GB of memory to do the profiling stage. Based on what was mentioned and citations, we should choose POIs to boost the efficiency of the profiling stage. On the other hand, we will show how POIs have been removed by inserting random delays. However, the time randomization method will apply an enormous value of noise into signals and detection of the peak will be pointless because it will not contain useful information. We must be used $35,000$ sample points per power trace and more than profiling power traces are required for a protected AES. Dealing with this computational complexity will be impossible. It can be concluded that profiled attacks are so sensitive to this countermeasure and our result confirms this sensitivity.

6 Conclusion

n this paper, we focus on TAs and propose a countermeasure against it. We introduced the countermeasure based on time randomization to filtering interesting points. We performed three experiments. Firstly, power traces extracted from a software implementation of the AES without countermeasures. Secondly, power traces are extracted from AES by inserting one random delay. Thirdly, power traces are extracted from AES by inserting multiple random delays. We ran the feature selection technique (SOST) on three methods and showed how POIs had been removed. Results showed our countermeasure counteracts the key recovery effort based on template attacks. Time randomization as a countermeasure also results in 7% time overhead and the grade of the key recovery is raised exponentially in the protected design (e.g. $2^{106.26}$ for 400K traces to 2^{100} for 700K traces). As future work, it would be interesting to implement random shuffling on AES that is one of the time randomization techniques.

References

[1] C. Archambeau, E. Peeters, F. X. Standaert, and J. J. Quisquater. Template attacks in principal subspaces. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4249 LNCS:1–14, 2006.

[2] B. Timon. Non-profiled deep learning-based side-channel attacks. *International Journal of Network Security*, pages 1–34, 2018.

[3] K. M. Abdellatif, D. Courousse, O. Potin, and P. Jaillon. Filtering-based cpa: A successful side-channel attack against desynchronization countermeasures. *ACM International Conference Proceeding Series*, pages 29–32, 2017.

[4] Y. Zhou and F. X. Standaert. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized resnet model for side-channel attacks. *Journal of Cryptographic Engineering*, 2019.

[5] S. chari, J. R. Rao, and P. Rohatgi. Template attacks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2523:13–28, 2003.

[6] L. Lerman, S. F. Medeiros, N. Veshchikov, C. Meuter, G. Bontempi, and O. Markowitch. Semi-supervised template attack. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7864 LNCS:184–199, 2013.

[7] M. Bar, H. Drexler, and J. Pulkus. Improved template attacks. *Constructive Side-Channel Analysis and Secure Design*, pages 81–89, February 2010.

[8] M. O. Choudary and M. G. Kuhn. Efficient, portable template attacks. *IEEE Transactions on Information Forensics and Security*, 13(2):490–501, 2018.

[9] W. Schindler, K. Lemke, and C. Paar. A stochastic model for differential side channel cryptanalysis. *CHES*, 3659:3046, Sept 2005.

[10] G. Fan, Y. Zhou, H. Zhang, , and D. Feng. How to choose interesting points for template attacks more effectively? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9473:168–183, 2015.

[11] E. Oswald and S. Mangard. Template attacks on masking—resistance is futile. *CT-RSA*, LNCS 4377:243–256, February 2007.

[12] H. Maghrebi, S. Guilley, and J. L. Danger. Leakage squeezing countermeasure against high-order attack. *WISTP*, 6633 LNCS:208–223, 2011.

[13] M. Renauld. Cryptanalysis of the ches 2009/2010 random delay countermeasure. *CHESS*, 280141:29–41, December 2013.

[14] F. Kordi, H. Hosseintalaee, A. Jahanian, and A. Legay. Cost-effective and practical countermeasure against the template side channel attack. *2020 17th International ISC Conference on Information Security and Cryptology (ISCISC)*, 10239:126–131, September 2020.

[15] C. Herbst, E. Oswald, and S. Mangard. An aes smart card implementation resistant to power analysis attacks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3989 LNCS:239–252, April 2006.

[16] A. Barenghi, W. Fornaciari, G. Pelosi, and D. Zoni. Scramble suit: A profile differentiation countermeasure to prevent template attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2019.

[17] E. Cagli, C. Dumas, and E. Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. *Cryptographic Hardware and Embedded Systems-CHES*, 10529 LCNS:45–68, 2017.

[18] R. Gu, P. Wang, M. Zheng, H. Hu, and N. Yu. Adversarial attack based countermeasures against deep learning side-channel attacks. *arXiv - CS - Cryptography and Security*, 2020.

[19] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. *In International workshop on cryptographic hardware and embedded systems*, pages 16–29, 2004.

[20] S. Picek, A. Heuser, A. Jovic, and A. Legay. Hierarchical classification for machine learning side-channel attacks. *AFRICACRYPT*, 10239:61–78, April 2017.

[21] L. Lerman, G. Bontempi, and O. Markowitch. Side channel attack: an approach based on machine learning. *Second International Workshop on Constructive SideChannel Analysis and Secure Design*, pages 29–41, 2011.

[22] J. S. Coron and I. Kizhvatov. An efficient method for random delay generation in embedded software. *CHESS*, 5747 LNCS:156–170, 2009.

Farshideh Kordi received his B.S. degree in computer hadrware engineering from University of Tehran, Tehran, Iran in 2007 and the M.S. degrees in computer engineering from Shahid Beheshti University of Technology, Tehran, Iran in 2020. His current research interests are hardware security and FPGA-based system design.

Hamed Hosseintalaee received his B.S. degree in computer hardware engineering from Khajeh Nasir Toosi University of Technology, Tehran, Iran in 2014 and the M.S. degrees in computer engineering from Shahid Beheshti University of Technology, Tehran, Iran in 2016. His current research interests are hardware security and VLSI design automation.

Ali Jahanian received his B.S. degree in computer engineering from University of Tehran, Tehran, Iranin 1996 and the M.S. and Ph.D. degrees in computer engineering from Amirkabir University of Technology,Tehran, Iran in 1998 and 2008, respectively. He joined Shahid BeheshtiUniversity, Tehran, Iran in 2008. His research interests are hardware security and biochips design.