A Novel Profiled Side-Channel Attack in Presence of High Algorithmic Noise

Mostafa Taha and Patrick Schaumont
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061, USA
Email: mtaha, schaum@vt.edu

***Abstract*—Understanding the nature of hardware designs is a vital element in a successful Side-Channel Analysis. The inherent parallelism of these designs adds excessive Algorithmic Noise in the power consumption trace, which makes it difficult to mount a successful power attack against it. In this paper, we address this high Algorithmic Noise with a novel profiled attack that is generic and independent of any specific cryptographic algorithm. We propose both a new profiling phase and two new insights in the attack phase. The proposed profiling technique takes the high design parallelism into consideration, which results in a more accurate power model. In the attack phase, we first define two new targeted regions in the power trace, then aggregate the attack results from each of them to get a more powerful attack phase. The proposed attack model has been tested on the 128-bit AES of the widely known DPA Contest (V2) and achieved a stable 80% Global Success Rate (GSR) at 2755 traces.**

I. Introduction

Side-Channel Analysis (SCA) is a subfield of cryptanalysis, where the attacker uses physical observations of the cryptographic module to collect information about the secret key. The instantaneous power consumption of the module depends on the state of its internal registers. The internal registers depend on the secret key. Hence, a link can be built between the power consumption trace and the secret key. The power trace in this context is a side-channel output. Side-Channel outputs are any byproduct that reveal information about the internal state of the hardware including power consumption [1], electromagnetic radiation [2] and execution time [3].

A Side-Channel Attack at a high abstraction level consists of building a model that mimics the power consumption of the targeted module and using this model to estimate a key that generates a similar power consumption as shown in Fig.1. Practically speaking, a complete attack requires five steps. First, the attacker collects a set of actual power traces at a known (random or chosen) plaintext. Second, he analyzes the algorithm searching for a sensitive intermediate variable that depends on both the plaintext and a small segment of the key (subkey). Then, he chooses a subkey guess and calculates the intermediate variables using the same plaintexts. Then, a power model is used to map the calculated intermediate variable into an equivalent power consumption. Finally, he uses a distinguisher to compare the modeled power trace to the actual power trace searching for the subkey that results in the best match. The comparison should focus on the time region where the intermediate variable is expected to show up in the power trace. The attacker should repeat step three to five for every subkey until all the subkeys are revealed.

This research was supported in part by the VT-MENA program of Egypt, and by National Science Foundation Grant no. 1115839.

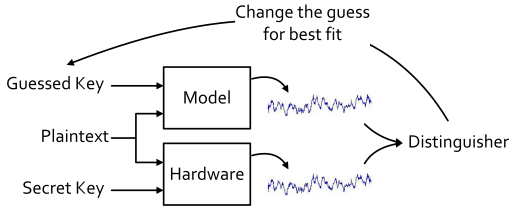

Fig. 1. SCA Attack Model

There are two types of power models used in the literature. Non-profiled models are the most simple and can be the Hamming Weight or Hamming Distance. Performance analysis of those two models along with other non-profiled models can be found in [4]. Profiled models depend on extracting the power model from an actual set of profiling traces collected from an identical device. An overview of the common profiling techniques will be presented in Section II.

The major challenge faced by SCA in plain implementations (without countermeasures) is the noise in measured traces. The dominant source of noise specially in hardware designs of block ciphers is the Algorithmic Noise. Hardware designs in ASIC's and FPGA's are inherently parallel. To achieve a one round per clock, cryptographic algorithms are designed so that all the input bits are processed simultaneously in small parallel logic blocks. While guessing one subkey in the attack, the power consumption of the other running logic blocks will alter the measured trace by unknown values which represents the Algorithmic Noise. Developing a profiled attack model for this noisy environment is the objective of this paper.

In this paper, three of the five attack steps will be revised in Section III. We will start with the definition of 'Targeted Operation'. Then, we propose a novel profiling technique that takes high design parallelism into consideration. The attack phase is also improved with two insights. First, we will define two new targeted attack regions in the power trace. Second, we will aggregate the attack results from each of them to get a more powerful attack phase. The results for a case study are presented in Section IV. Section V presents the conclusion.

II. Related Work

The main contribution in this paper is to introduce a new profiling technique. This section presents an overview of the common profiling techniques that have been used in the literature and how the proposed method is different and more suitable in presence of high Algorithmic Noise.

Chari et al. presented the first profiling technique which is the Template Attack [5]. Their main concern was to build a noise model for the power traces. They used the average and covariance matrix of a set of power traces measured at the same input as a template for this input. The profiling set should be used to build a template for every possible combination of plaintext and subkey. The template will be the basis of a multivariate normal distribution representing the noise model. In the attack phase, the authors used the maximum likelihood to find the template that best matches the attack traces which will lead directly to the correct subkey. Typically, the size of the profiling set should be multiple times of the size of all input combinations which can be reduced by building templates with respect to the intermediate variable directly [6]. The weak point of the Template Attack is the limited number of trace points that can be supported in the covariance matrix. The size of each covariance matrix grows quadratically with the number of trace points. Moreover, the complexity of calculating the matrix inversion which is required in the attack phase, grows almost cubically with the number of trace points (depending on the algorithm used).

Schindler et al. proposed the Stochastic Attack [7] where they used regression analysis to give a different weight to every bit of the intermediate variable, as a way to improve the accuracy of the Hamming Weight (or Hamming Distance) model. The weights are estimated based on approximating the profiling traces to a function of the bits of their corresponding intermediate variables. Schindler et al. assumed that the leakage of each bit is separable and that the system power trace is the sum of the leakages of these individual bits. However, the reality is more complicated and the leakage depends on the relation between bits. This led to the extension of analysis dimensions to cover the effect of each 2 bits (37 dimensions) [8], 3 bits (93 dimensions), and so on up to all the 8 bits in 255 dimensions [9].

The difference between our proposed technique and the two previously mentioned profiling techniques is highlighted with an example in Fig.2. We assume that the targeted device uses four parallel logic blocks, the data width is four bits and the Algorithmic Noise (ϵ) is the dominant source of noise. The source of Algorithmic Noise is the power consumption of other logic blocks running simultaneously with the targeted block. We also assume that:

- $\mathcal{C}$: The ciphertext.
- $\mathcal{K}$: The correct key used to collect traces.
- I_x: The intermediate variable at logic block number x.
- $b_y^{I_x}$: Bit number y of the intermediate variable I_x.
- $l(z)$: The leakage function of z.

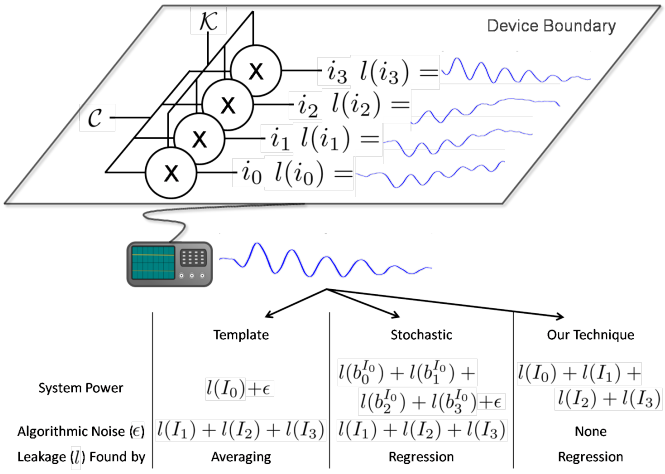

Fig. 2. Comparison between how different profiling techniques expect the system power trace. Profiling in the Template Attack expects the leakage of one intermediate variable, and extracts an estimate of it using averaging. Profiling in the Stochastic Attack expects the aggregate leakage of small building blocks of one intermediate variable, and extracts an estimate of each of them using regression. Our profiling technique expects the aggregate leakage of all the used intermediate variables, and extracts an estimate of each of them using regression.

The leakage function is the one-to-one mapping between the intermediate variable and the corresponding power consumption in the power model. The figure shows the intermediate variable as a function of both a subkey and a small segment of the ciphertext. The targeted device runs several logic blocks simultaneously, and each of them creates one intermediate variable and consumes a certain power. The system power trace is the aggregated sum of the power consumption of all the logic blocks. The Template Attack expects to see the leakage of one intermediate variable. It extracts an estimate of it by averaging over a large number of traces to cancel out the effect of Algorithmic Noise. The averaging should be done with respect to every possible intermediate variable. The Stochastic Attack expects to see the aggregate leakage of individual bits (or combination of bits) of one intermediate variable. It extracts an estimate of the leakage of each bit by using the linear least squares regression. The number of required traces should be much larger than the number of unknown leakage functions to cancel out the effect of Algorithmic Noise.

Note that, all subkeys used in different logic blocks are known in the profiling set, and this information can be used to build a more accurate power model. Our proposed technique understands that the system trace is the aggregate leakage of processing all the intermediate variables. We estimate the leakage function of each intermediate variable (the value itself, not the individual bits) by regression. Hence, our technique is a mix between the previously two mentioned techniques. The design of our profiling technique acknowledges the presence of Algorithmic Noise, and uses it to get a more accurate power model. It is worth mentioning that, the Algorithmic Noise is removed in the profiling phase of our attack. However, this noise will still affect the attack phase.

III. Proposed Attack Model

In this section, we first present several preliminaries that are required by the proposed attack. The actual attack model is presented later on.

A. Targeted Operation

The term of 'Intermediate Variable' is not generic enough, because the Hamming Distance model (for example) uses the XORing between two intermediate variables where the result of XORing is not an intermediate variable by itself. Hence, we propose to use the term 'Targeted Operation' which is more generic and intuitive.

The Targeted Operation is an operation running in the module, where the attacker searches for its power consumption in the recorded trace. The Targeted Operation should depend on both the plaintext and the key with as high diffusion and low confusion as possible. Non-linear diffusion helps SCA in finding the exact secret key as if one bit is miss-guessed, the inspected power consumption will change entirely. Low confusion between subkeys reduces the extensive search space of the entire key to a smaller space of a single subkey. Low confusion can typically be achieved by attacking the first/last round of the cryptographic algorithm. The principle of reducing the key search space and attacking small segments of key is the so called the divide-and-conquer principle [10].

The Targeted Operation has an index and a dimension. The operation index is a number that uniquely identifies the leakage of a single instance of the operation, and that differentiates it from other instances with different leakages. The calculation of the index must be data dependent and include the key as the sole unknown. If the same index was given to more than one instance of the operation, the attacker acknowledges that the leakage of these instances are considered the same. For example, if we targeted the operation of charging a byte register, with the Hamming Weight as the index, we implicitly acknowledge that the instance of charging the register to $0x0F$ has the same leakage as the instance of charging it to $0xAA$ as they both have the same Hamming Weight of 4. The best index size is the one that allocates a different value to every possible leakage value. Template Attack is therefore better than the Hamming Weight as it assigns the byte value itself as the operation index, hence it can differentiate between the aforementioned instances of charging a byte register.

The operation dimension is the number of intermediate variables used to evaluate the index. For example, if we used the Hamming Distance as the index in the previous example, we need to evaluate the input to the register in two consecutive clock cycles in a 2-dimensional operation. The best dimension is the one that represents the actual source of power consumption. For example, if the power consumption depends on switching activity, the 2-dimensional operation should be used.

Considering the same example of charging a byte register as above, if we used the Hamming Distance as the index, we acknowledge that the instance of updating the register value from $0x0F$ to $0x1F$ has the same leakage as updating it from $0x07$ to $0x0F$ as they both have the same Hamming Distance of 1. The index can be improved to the result of XORing the two input values, where the index size will be 256 for this example. This index can now differentiate between the aforementioned two instances. The XORing has been used as the index in the stochastic attack of [9], and will also be used in this paper.

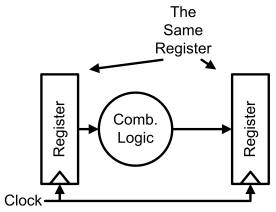

Fig. 3. The Extra Round

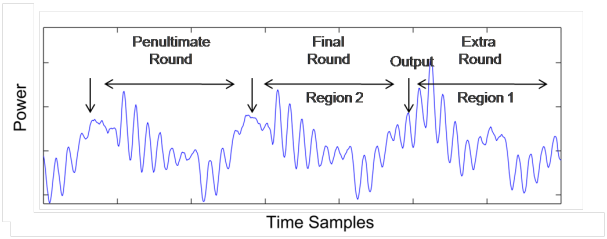

Fig. 4. Attack Regions

B. The Extra Round

Figure 3 shows an implementation of one round of a round-based cryptographic algorithm utilizing one round per clock. The combinational logic shown includes at least an XORing with the key, a non-linear diffusion operation, and a mixing operation for confusion. Once the input is ready in the register, the signal will pass through the combinational logic to generate the output and wait for the clock signal to store it in the same register. In the final round, the circuit behavior will still be the same. Once the final output is stored in the register, the combinational logic will execute and evaluate the output, which is never stored or used. This means that the combinational logic runs on the ciphertext output in an extra round, an operation that is not intended to happen. In this paper, we exploit this point and target the 2-dimensional switching activity of the register and the combinational logic between final round and output. The new attack region is marked at 'Region 1' in Fig.4 along with the time of the last two rounds in a symbolic representation of the power trace. This wide attack region increases the amount of information extracted from each trace for a more powerful attack.

C. Assumptions

In the proposed attack model, we define the unit as the register and the combinational logic circuit following it. The problem in extracting the power model is that the system

power trace is the aggregated sum of all the units. The objective in this paper is to separate them out into different power traces that represent different units.
We follow the same assumption used in Section II, and further assume that:

- R: The output of the penultimate round.
- N_U: The number of parallel units.
- R_u, $\mathcal{C}_u$: The inputs to unit number u.
- I_u: The index at unit number u, $I_u = R_u \oplus \mathcal{C}_u$.
- N_I: The size of the operation index.
- P: The system power trace.
- N_P: The number of profiling traces.

The Targeted Operation is the 2-dimensional operation of updating a unit input from R_u to $\mathcal{C}_u$. The operation index depends only on the unit inputs, hence we assume that the leakage of every unit is the same if they were fed with the same inputs regardless of the location of the unit within the chip. This assumption is fairly reasonable as they were implemented using the same RTL, and are different only for their place and route. This means that we need to have only one set of templates that are reusable over all the processing units.
In the following, we will use a set of profiling traces P to extract an estimate of l for every instance of the unit, which is the unknown leakage model. Then we will use the power model to attack the set of attack traces at an unknown key.

D. Profiling Phase

In the profiling step, we assume that the attacker collects a set of traces using known (chosen or random) keys. The measured power trace will be the aggregated sum of the power consumption of N_U different units

$$P = \sum_{u=0}^{N_U-1} l(I_u) \tag{1}$$

Now consider that, every unit can implement N_I different instances of the operation. We assume that $g(i)$ is the number of units executing in an operation instance with $I = i$. Hence, the power consumption trace will be

$$P = \sum_{i=0}^{N_I-1} g(i)l(i) \tag{2}$$

Note that, the summation of $g(i)$ over i will always be equal to N_U. For N_P profiling traces, the power consumption can be expressed in a matrix format as

$$\mathbf{P} = \mathbf{G} \times \mathbf{L} \tag{3}$$

The size of matrix $\mathbf{P}$ is $(N_P \times C)$, where C is the number of samples covering the attack region. The size of $\mathbf{G}$ is $(N_P \times N_I)$. the size of $\mathbf{L}$ is $(N_I \times C)$. The best solution for $\mathbf{L}$ can be found using the least squares equation

$$\hat{\mathbf{L}} = (\mathbf{G}'\mathbf{G})^{-1}\mathbf{G}' \times \mathbf{P} \tag{4}$$

In case of a large index size and relatively small number of parallel units, the matrix $\mathbf{G}$ will be a sparse matrix with a maximum of N_U/N_I non-zero elements. The first row of $\hat{\mathbf{L}}$ is $\hat{l}(0)$, the second row is $\hat{l}(1)$ and so on. The matrix $\hat{\mathbf{L}}$ will be used as the power model in the next section.

E. Attack Phase

In the attack phase, we target a single key byte and choose a key guess. The key guess along with the input plaintexts can be used to calculate the operation index. Next, the matrix of modeled traces can be build by mapping every operation index to its corresponding model out of the $\hat{\mathbf{L}}$ matrix. Once the matrix of modeled power traces is built, a distinguisher will be used to compare modeled traces to measured ones. In this work, we need to compare a large range of trace points in the attack region hence, we will use the sum of point-by-point correlation which reveals the complexity of calculating the covariance matrix associated with Template Attack. We assume that

- $\mathbf{P}$: The matrix of attack traces.
- $\bar{K}$: The subkey guess used to build the modeled traces.
- $\bar{\mathbf{P}}_{\bar{K}}$: The matrix of modeled traces.
- $D_{\bar{K}}$: The distinguisher result using $\bar{K}$ as the key guess.

We also assume that $\mathbf{P}_c$ and $\bar{\mathbf{P}}_{\bar{K},c}$ are column number c of $\mathbf{P}$ and $\bar{\mathbf{P}}_{\bar{K}}$ respectively. The distinguisher will be

$$D_{\bar{K}} = \sum_{c=0}^{C-1} \frac{cov(\mathbf{P}_c, \bar{\mathbf{P}}_{\bar{K},c})}{\sigma_{\mathbf{P}_c}\sigma_{\bar{\mathbf{P}}_{\bar{K},c}}} \tag{5}$$

The distinguisher should be calculated for all the possible key guesses. The correct key is the one that maximizes $D_{\bar{K}}$.

F. Two Regions

Every point of the power trace gives more information to the attack, only if we know how to use it. The idea of aggregating separate attacks against different points of the trace has been proposed in [11]. In this paper, we will use a similar idea to aggregate the attack against the extra round with the attack against the final round. In this section, we will describe how to build an attack against the final round ('Region 2' in Fig.4), and how to aggregate the attack results from the two regions. Previously, we used the knowledge of the output of the penultimate round R in a 2-dimensional attack against the extra round. Here, we will use the same value in a 1-dimensional attack against the final round. A 2-dimensional attack can not be used here, because calculating the previous value of the register requires passing through another round up which mostly involves a confusion step. Passing through a confusion step requires the knowledge of all the subkeys involved, which violates the divide-and-conquer principle and broadens the search space.
The previously discussed analysis can still be used with only minor changes

- I_u: The index at unit number u, $I_u = R_u$.

The Targeted Operation is the 1-dimensional operation of storing R_u in a unit. The attack points involved in P should

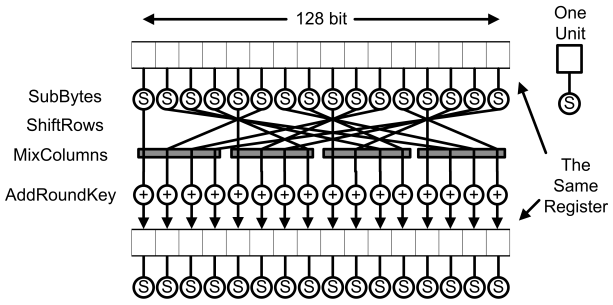

Fig. 5. Last Round of AES

be those samples in time that correspond to the final round. The combined distinguisher will be

$$D_{\bar{K}} = D1_{\bar{K}} + D2_{\bar{K}} \tag{6}$$

where $D1_{\bar{K}}$, is the result of attacking the extra round and $D2_{\bar{K}}$ is the result of attacking the final round. The interesting point about attacking two different regions in the same trace is that the noisy wrong result in a one region will not show up (with high probability) in the other region as well however, the correct key will give high correlation values in both of them.

IV. Results of a Case Study: AES

The DPA contest [12] offers a common set of power traces to make it possible to compare the performance of different SCA attack models. The traces are collected for a hardware 128-bit AES encryption algorithm running on SASEBO-GII board. There are two public sets of traces, a profiling set of 1 million traces, and an attack set of 20,000 traces for 32 different random keys (640,000 traces). Those keys are known, and used by the researcher to characterize his own attack model. In this section, we will target this implementation (which is not our design) to demonstrate the existence of the extra round and the efficiency of the overall attack.

A. The AES

The AES design runs at one round per clock using 16 parallel SBox combinational circuits as shown in Fig.5. More information about the AES encryption algorithm can be found in [13]. The location of MixColumn function is shown in the figure with no effect on the data-path, as there is no MixColumn function in the final round.

In this case, the targeted unit is a one byte register and the SBox combinational logic following it where the number of parallel units N_U will be 16. The targeted operation in the extra round, will be the 2-dimensional operation of updating the unit inputs with the ciphertext. The operation index will be

$$I_u = \mathcal{C}_u \oplus \mathcal{S}^{-1}(\mathcal{SH}^{-1}(\mathcal{C}_u \oplus \mathcal{K}_u)) \tag{7}$$

where $\mathcal{S}^{-1}$ is the inverse of SBox function and $\mathcal{SH}^{-1}$ is the inverse of ShiftRows function. The targeted operation in the final round, will be the 1-dimensional operation of storing R_u in the unit. The operation index will be

$$I_u = \mathcal{S}^{-1}(\mathcal{SH}^{-1}(\mathcal{C}_u \oplus \mathcal{K}_u)) \tag{8}$$

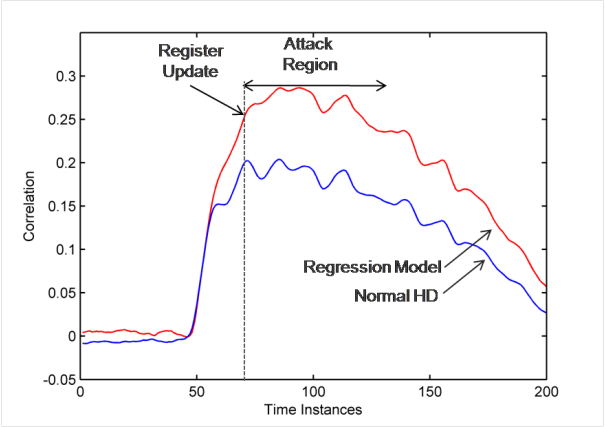

Fig. 6. Proposed Regression Model and the Hamming Distance Model

The size of the operation index N_I in both cases will be 256. The attack regions are identified using the normal Correlation Power Analysis [14] with Hamming Distance Model.

B. The Extra Round

A simple test was conducted to check the existence of the extra round along with the effectiveness of the new modeling method. In this test, we calculated the correlation between the actual measured traces and the modeled traces in the attack region of the extra round. The modeled traces were built using the proposed regression model and the normal Hamming Distance model as a reference. The test was applied to key 1 of the public set, with all the 20000 traces. Figure 6 shows the result of this test. The peak correlation point in the Hamming Distance model is the point of updating the register value with the output ciphertext. The correlation goes down as the signal passes through the SBox circuit. However, the correlation of the proposed regression model achieves higher values in the region of SBox circuit. The correlation values in the figure are relatively high because we assume that we know the subkeys of all the units and aggregate the power models of them to cancel the Algorithmic Noise in this test. However, when we target only one unit in an actual attack, the 15 other units will be considered as Algorithmic Noise. As previously mentioned, The Algorithmic Noise is removed in the proposed profiling phase however, it still affects the attack phase.

C. Performance Metrics

The performance metrics used in this study are the same as those in the DPA Contest (V2):

- Global Success Rate (GSR): The number of traces so that all the subkey bytes are recovered concurrently.
- Partial Success Rate (PSR): The number of traces so that a selected subkey is recovered correctly. Every subkey byte will have its own PSR.
- Partial Guessing Entropy (PGE): The rank of the correct subkey within the result. Every subkey byte will have its own PGE.

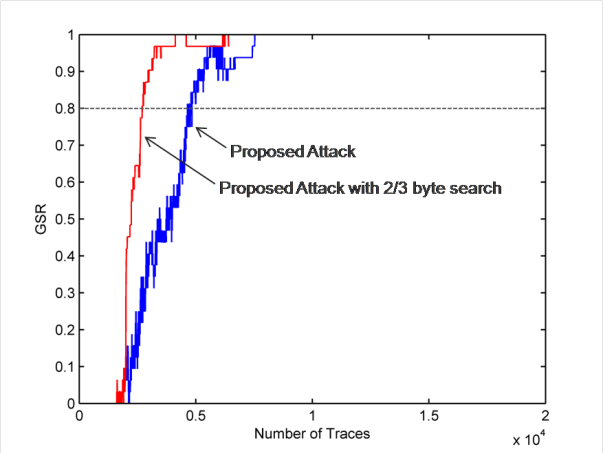

Fig. 7. Global Success Rate

D. Results

The proposed attack model has been used to attack the public set of the DPA Contest (V2). The GSR is shown in Fig.7. The figure shows that the proposed attack achieved 80% GSR at 4641 traces, where the 80% is calculated over the 32 experiments i.e. at least 26 experiments have been recovered completely by this number of traces.

This result reflects a side effect of using one power model to attack all the 16 logic blocks. The extracted power model is considered the average of all of them. However, there may be one (or more) logic blocks that has a shifted model due to differences in placing and routing. Hence, it will be difficult for our proposed attack to recover the correct subkey of this shifted logic block. As the GSR requires recovering all the subkeys concurrently, this shifted subkey will pull the GSR up.

To recover this side effect, we improved our attack with a 2/3 subkey byte search. In this improvement, we targeted the penultimate round with a similar attack. The 2-dimensional operation at the penultimate round involves at least 10 subkey bytes because of the MixColumn step. In the 2/3 subkey byte search, we test all the combinations of the best 3 guess of the worst two bytes (a total of 10 tests). The choice of those worst bytes is done as part of the profiling phase. The GSR of the proposed attack including the 2/3 subkey byte search is also shown in Fig.7. This simple key search improved the 80% GSR from 4641 traces to 2755 traces.

To demonstrate the performance of the proposed complete attack, we compared its performance to the best two submissions of the DPA Contest in Table I. The table shows the previously discussed 80% GSR. It also shows the minimum PSR $> 80\%$, where the 80% is calculated over the 32 experiments and the minimum reflects the worst case subkey out of the 16 subkey bytes. Finally, the table shows the maximum PGE < 10, where the maximum reflects the worst case subkey out of the 16 subkey bytes.

TABLE I
COMPARING RESULTS

Attack	GSR>80%	min PSR>80%	max PGE<10
Our Attack	2,755	2,226	1,420
Li [15]	2,256	2,155	3,181
Heuser [9]	3,589	2,748	1,356

V. CONCLUSION

In this paper, we proposed a profiled attack model in presence of high Algorithmic Noise. We proposed a novel profiling technique that acknowledges the high noise in these designs. We also proposed two new insights in the attack phase. One insight is to exploit the effect of the fixed connections of combinational logic circuits. The other insight is to aggregate the results of attacking two different regions in the same trace.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology CRYPTO'99*, ser. Lecture Notes in Computer Science, 1999, vol. 1666, pp. 789–789.

[2] D. Agrawal, B. Archambeault, J. Rao, and P. Rohatgi, "The EM SideChannel(s)," in *Cryptographic Hardware and Embedded Systems - CHES'02*, ser. Lecture Notes in Computer Science, 2003, vol. 2523, pp. 29–45.

[3] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology CRYPTO'96*, ser. Lecture Notes in Computer Science, 1996, vol. 1109, pp. 104–113.

[4] J. Doget, E. Prouff, M. Rivain, and F. Standaert, "Univariate side channel attacks and leakage modeling," *Journal of Cryptographic Engineering*, vol. 1, no. 2, pp. 123–144, 2011.

[5] S. Chari, J. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES'02*, ser. Lecture Notes in Computer Science, 2003, vol. 2523, pp. 51–62.

[6] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks revealing the secrets of smart cards*. New York, N.Y.: Springer, 2007.

[7] W. Schindler, K. Lemke, and C. Paar, "A stochastic model for differential side channel cryptanalysis," in *Cryptographic Hardware and Embedded Systems CHES'05*, ser. Lecture Notes in Computer Science, 2005, vol. 3659, pp. 30–46.

[8] A. Heuser, W. Schindler, and M. Stoettinger, "Revealing Side-Channel issues of complex circuits by enhanced leakage models," in *ACM/IEEE Design Automation and Test in Europe (DATE'12)*, Mar. 2012.

[9] A. Heuser, M. Kasper, W. Schindler, and M. Stttinger, "A new difference method for Side-Channel analysis with High-Dimensional leakage models," in *Topics in Cryptology CT-RSA'12*, ser. Lecture Notes in Computer Science, 2012, vol. 7178, pp. 365–382.

[10] F. Standaert, T. Malkin, and M. Yung, "A unified framework for the analysis of Side-Channel key recovery attacks," in *Advances in Cryptology - EUROCRYPT'09*, ser. Lecture Notes in Computer Science, 2009, vol. 5479, pp. 443–461.

[11] L. Batina, B. Gierlichs, and K. Lemke-Rust, "Differential cluster analysis," in *Cryptographic Hardware and Embedded Systems - CHES'09*, ser. Lecture Notes in Computer Science, 2009, vol. 5747, pp. 112–127.

[12] *DPA Contest v2 2009/2010*. Telecom ParisTech french University, 2009. [Online]. Available: http://www.dpacontest.org/v2/index.php

[13] J. Daemen and V. Rijmen, *The Design of Rijndael*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[14] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES'04*, ser. Lecture Notes in Computer Science, 2004, vol. 3156, pp. 135–152.

[15] Y. Li, D. Nakatsu, Q. Li, K. Ohta, and K. Sakiyama, "Clockwise collision analysis – overlooked side-channel leakage inside your measurements," Cryptology ePrint Archive, Report 2011/579, 2011, http://eprint.iacr.org/.