Template attacks on nano-scale CMOS devices

Bastian Richter1 · Amir Moradi1

Received: 23 April 2019 / Accepted: 8 April 2020 / Published online: 7 May 2020 © The Author(s) 2020

Abstract
Profiler attacks are widely considered to be the most powerful form of side-channel analysis attacks. A common form is known as Gaussian template attacks which fit a Gaussian distribution to better model the behavior of the target device. Since profiler attacks build the model based on a device identical to the target device, manufacturing variances are an important factor for the success of such attacks. With shrinking the feature size, the influence of manufacturing variation on the power consumption of integrated circuits increases. It has been warned that this issue might render template attacks less effective. We evaluate this assumption on an ASIC design manufactured in 40 nm technology. We characterize the introduced variation and show that these can be easily mitigated. By performing attacks on multiple samples of the same ASIC, we show that template attacks on small technology sizes are still successful.

Keywords Template attacks · Nano-scale devices · Side-channel analysis attacks · Profiled side-channel attacks

1 Introduction

For today’s embedded systems dealing with cryptographic primitives and secrets involved in cryptographic operations, side-channel analysis (SCA) attacks are considered as one of the most serious threats. Especially differential power analysis (DPA) attacks as introduced by Kocher et al. [15] and later extended to correlation power Analysis (CPA) attacks [3] have been proven to be powerful tools to extract secrets from cryptographic devices when the attacker has physical access to the target [10,23,24]. Along the same line, compared to that using power consumption, measuring the electromagnetic emanation (EM) of the device can lead to stronger attacks [11] since EM signals can be localized and are usually less influenced by other irrelevant parts of the circuit.

In general, such multi-query attacks are conducted under a black-box scenario, where no (or little) information about the device-under-test (DUT) is known. In contrast, profiling SCA attacks have access to and full control over a device identical to the DUT, with which the power consumption (or EM) characteristics of the device can be studied. Such a device is referred to as profiling device. In short, during the profiling phase, the attacker collects as many measurements as required from such a device, whose entire intermediate values are known. Using such profiles, the attack is conducted on the DUT by means of very small number of measurements (ideally a single one). The first and the most popular method in this area is the Gaussian template attack (TA) introduced by Chari et al. [6]. During the profiling phase, based on the value of a chosen intermediate value, the attacker estimates multivariate Gaussian distributions (so-called profiles) using the measurements collected from the profiling device. Later, the attacker makes use of the profiles to predict the targeted intermediate value in each SCA measurement collected from the DUT. The Gaussian distribution is the most commonly used distribution for this kind of attack although it is also possible to base it on other distributions. Due to the multivariate nature of this method, it can recover the secret using less number of measurements compared to the attacks under black-box scenario. Profiling attacks are even able to directly target the secret values which are independent of the given inputs. For example, such attacks can target the key chunks (e.g., bytes) when they are transferred from memory to registers, e.g., for key schedules. Since key schedule is independent of the cipher input (plain-
text/ciphertext), the associated leakages cannot be exploited by multi-query DPA/CPA attacks. Apart from Gaussian template attacks, there are also other forms of profiling attacks which make use of machine learning techniques like support vector machine (SVM) [14,16] or deep neural networks (DNN) [17,19]. Notably, DNNs have shown very promising results when applied in cases where the implementation is protected by means of temporal noise, i.e., randomized or jitter-full clock [4].

The steady shrinking of CMOS feature sizes steadily increases the speed and power efficiency of modern devices. The smaller structures consume significantly less dynamic power due to the lower resulting capacitances in the gate and in the shorter connections [33]. So far mostly dynamic power consumption contributed to the information leakage exploited by SCA attacks. However, due to the increase in static power consumption in smaller technology nodes, it can also be considered as a source of information leakage [21,22,26]. Djukanovic et al. [9] and Bellizia et al. [2] also examined the application of multivariate attacks to static leakage by first investigating the influence of temperature on the information leakage and then creating multiple measurements with different temperatures for one attack to acquire multiple dimensions. The decrease in dynamic power consumption may complicate power analysis attacks on circuits build using the newest technologies, which can be considered as a positive development.

More importantly, process variation is more intensively observed in new smaller technology nodes. Since the concept behind the profiling SCA attacks relies on the similarity of the profiling device and the DUT (and their corresponding power/energy characteristics), severe process variation has been expected to increasingly hinder successful profiling SCA attacks. Renaud et al. [27] examined the power variability of a nano-scale chip and its influence on SCA attacks. Based on practical experiments on a 65 nm circuit, they concluded that variability of power consumption pattern of different chips would make it very challenging to successfully conduct template attacks. Hence, it is expected that by decreasing the feature size in the future and more intensive process variation such attacks will get more and more difficult. According to personal communication with the authors, the device under test in [27] was a single AES S-box implemented as a fully combinatorial circuit without any register at input/output or control logic. The 8-bit input and 8-bit output signals of the S-box were provided as physical I/O pins of the chip. Although the authors put external register banks (on PCB level) at the input and output of the chip, this leads to very dominant changes in power consumption curves when the input of the S-box alters. Such changes are due to the activity of the energy-consuming I/O cells and fan out of the chip and are seen as strong noise in the measurement. Further, ASIC samples are slightly different in their package, e.g., in the length of bonding wires. Since no register is packed into the targeted ASIC, the changes on S-box input and output pins lead to various amount of power consumption in different ASIC samples. This can justify the variety that the authors have observed in [27].

It is noteworthy that susceptibility of devices to profiling attacks are usually examined under the worst-case scenario. More precisely, a single device is used in both profiling and attack phases. Under such circumstances, it is examined whether the attack is successful even if the DUT and profiling device have a very similar (ideally identical) power consumption characteristics. In this work, we conduct Gaussian template attacks on the AES encryption function implemented by a 40 nm ASIC standard library. In contrast to the worst-case scenario, we try to evaluate the real-world applicability of such profiling attacks by examining 11 ASIC sample chips. This also includes analyzing a full AES implementation compared to the single S-box of [27]. Our experiments also enabled examining different intermediate values and models for the templates which turned out to highly affect the results. As a side note, every ASIC sample includes seven AES encryption cores with identical netlist but with different placement and routing. This allowed us to quantify how strongly the routing influences such profiling attacks, when profiling and attack devices are not from the same placement and routing. This can be of high interest when dealing with selling and cloning third-party IP cores. In short, we found that the attacks are still easily possible. Based on our experimental results (only valid on the underlying technology of our prototyped ASIC samples), the increasing process variation in modern technology nodes does not strongly affect the success and feasibility of profiling attacks. To be more precise, such variations are easily compensated by already-available portability methods like mean compensation [20] usually used to compensate other variations, e.g., in the measurement setup.

2 Background

2.1 Template attacks

In Gaussian template attacks, it is assumed that the adversary is able to obtain a device identical to the DUT [6]. This enables a profiling phase in which the attacker uses the profiling device to build multivariate Gaussian models for the leakage associated with intermediate values. The built model can then be applied to attack the DUT with a few number of measurements.
2.1.1 Profiling phase

Following the notation of [7], we build multivariate Gaussian distributions for the leakages associated with the value \( k \in \mathcal{S} \) with \( \mathcal{S} \) being an intermediate value of the calculation. This can either be a certain value or a model like Hamming Distance (HD) of consecutive values stored in a register. For each value in \( \mathcal{S} \), we measure \( n \) traces, each represented as

\[
x = (t_1, \ldots, t_m)
\]

with \( m \) sample points \( t_i \in \mathbb{R} \). The \( m \) sample points in \( x \) are usually selected or compressed from the originally measured traces (this is discussed in more details in Sect. 2.1.3). \( \mathbf{X}_k \in \mathbb{R}^{n \times m} \) is then a matrix of measurements for the value \( k \), with \( \mathbf{x}_{k,i} \) representing the \( i \)-th row of matrix \( \mathbf{X}_k \). We can then compute the parameters needed to describe the multivariate Gaussian distribution which are the sample mean vector \( \bar{x}_k \in \mathbb{R}^m \) and the sample covariance matrix \( S_k \in \mathbb{R}^{m \times m} \) for each value \( k \in \mathcal{S} \):

\[
\bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_{k,i}
\]

\[
S_k = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_{k,i} - \bar{x}_k)(\mathbf{x}_{k,i} - \bar{x}_k)^T
\]

2.1.2 Attack phase

In the attack phase, the profile \( \bar{x}_k \) and \( S_k \) for a selected \( k \in \mathcal{S} \) is applied on a single trace \( y \) measured from the DUT by calculating the Gaussian probability density function pdf:

\[
\text{pdf}(y \mid \bar{x}_k, S_k) = \frac{1}{\sqrt{(2\pi)^m |S_k|}} \exp \left( -\frac{1}{2} (y - \bar{x}_k)^T S_k^{-1} (y - \bar{x}_k) \right). \tag{3}
\]

Repeating this for all \( \forall k \in \mathcal{S} \) results in a set of values that can be used as a discriminant score to rank the \( k \) candidates. Knowing the input (or output) associated with the attack trace \( y \), each key candidate is assigned to a category of \( k \) and therefore receives its pdf score. Since often a single attack trace is not sufficient to recover the key, the result of multiple attack traces should be combined. In this context, it is reasonable to calculate the logarithm of the pdf instead:

\[
\log \text{pdf}(y \mid \bar{x}_k, S_k) = -\frac{m}{2} \log 2\pi - \frac{1}{2} \log |S_k| - \frac{1}{2} (y - \bar{x}_k)^T S_k^{-1} (y - \bar{x}_k). \tag{4}
\]

It offers the advantage that the scores acquired by different attack traces can simply be summed up. It also prevents some numerical instabilities which can occur during the computation. When exploiting the leakages associated with an intermediate value which not only depends on the secret key but also on the algorithm input (or output), the multiple attack traces can be recorded for different inputs to exploit the full distribution. This process is usually called Template DPA [18].

2.1.3 Points of interest

The selection of points of interest, i.e., the points which contain the highest amount of information associated with the chosen intermediate value, is a crucial step in the preparation of template attacks. Although Gaussian multi-variate templates can benefit from correlated noise in additional points, there is a sweet spot of added points which has to be found. Including additional non-informative points can degrade the matching with the calculated templates. Another point is that the computational complexity quadratically depends on the number of points used for the templates. In order to select these points, the attacker can either use some metrics to directly select certain points of the traces or use data-dimensionality reduction methods to compress a part of the trace to a small amount of derived points.

There are several metrics available for directly selecting the points of interest. Estimating the difference-of-means (DOM) over different \( k \) values has been proposed in the original work introducing Gaussian template attacks [6]. Other methods include (1) sum of squared differences (SOSD) [12] which amplifies larger differences and prevents the cancellation of smaller signals with alternating signs and (2) the points with maximum correlation as the result of a CPA. Leakage detection tests like the \( t \)-test [28] might also be used but have to be modified for detecting the leakage of specific intermediate values and not the whole computation. Alternatively, signal-to-noise ratio (SNR) [18] can be used, that is defined as the ratio between the variance of the categories’ means and the variance of traces within the same category:

\[
\text{SNR} = \frac{\text{Var}_{\forall k \in \mathcal{S}}(\bar{x}_k)}{\text{Var}_{\forall k \in \mathcal{S}}(\text{Var}_{\forall i}(\mathbf{x}_{k,i}))}, \tag{5}
\]

with \( \text{Var}(\cdot) \) standing for variance and \( \text{E}(\cdot) \) for expected value.

Dimensionality reduction is most of the times performed by means of principle component analysis (PCA) [1] or linear discriminant analysis (LDA) [29]. While PCA maps the input dimensions (points of the traces) to orthogonal dimensions with maximum variance, LDA maximizes the ratio between inter-class and intra-class variance in the consecutively added orthogonal dimensions.\(^1\)

\(^1\) #1.

In our experiments, we did not use any dimensionality reduction. In order to allow us to compare the results with\(^2\) #1.

\(^2\) #1.
those of [27] and to not distort the results by being dependent on the preprocessing step, we select the points of interest by means of SNR (more details in Sect. 3).

2.2 Known improvements

There are different factors which can cause problems when building the templates or lead to a mismatch between the created templates and the measurements collected from the DUT. These might lie in minimal changes, e.g., environmental, during the measurements or numerical and statistical issues when building the templates with too few traces. Below, we shortly restate the common techniques known to compensate such effects.

2.2.1 Pooled covariance matrix

If too few traces are used to calculate the covariance matrix, it can happen that it is singular and thus not invertible (see Eqs. (3) and (4)). This often occurs if the categories of the TA are not equally likely, and thus some categories have not many profiling traces. As an example, this can happen when the input of the profiling device is randomly selected while the categories are defined based on HD of an intermediate value. This can cause the covariance to not be well estimated and thus inaccurate. Because in classical TA a Gaussian noise is assumed for each sample point, templates with the same points of interest often exhibit the same or a very similar covariance. Hence, it is possible to build a pooled covariance matrix for all templates instead separated ones for each category. In our experiments, since we worked with a high number of training traces, we did not encounter the aforementioned problems. Thus, we calculated separate covariance matrices for each category.

2.2.2 Mean adjustment

The main factors which interfere with the matching are parasitic resistance and capacitance introduced by the measurement setup. These parameters can be different in case of the profiling device and the DUT. Also, tolerances in the manufacturing or the packaging of the chip can have an influence, e.g., increased resistance due to longer bond wires. It has been shown in [8] that the main difference between measurements of different devices on the same setup is usually a DC offset.

In order to tolerate this, the attacker can shift the attack traces to make their mean the same as that of the profiling traces.

\[ x_{\text{adjusted}} = x + (\bar{x}_{\text{train}} - \bar{x}_{\text{attack}}) \]  

Here, \( \bar{x}_{\text{train}} \) is the mean over all training traces and \( \bar{x}_{\text{attack}} \) over all attack traces. The attacker then uses \( x_{\text{adjusted}} \) for the attack. When performing a template DPA, it is initially not known how many traces are needed to succeed. Hence, it is possible to either record a set of traces of a predefined size and calculate its mean. But the mean then includes information used from traces which might not be used for the rest of the attack so the number of traces used for the attack is actually not correct. Thus, we decided to estimate \( \bar{x}_{\text{attack}} \) incrementally and updated it for each trace added to the attack set.

The requirement to calculate the attack device’s mean trace restricts the attack to cases in which enough attack traces recorded with different inputs are available to properly approximate the mean, i.e., a reasonably uniform sample over the different classes should be acquired to not get a bias in the mean approximation. For example, a one-trace attack is not possible if mean adjustment is needed. Methods based on dimensionality reduction can help here as they might be able to compensate the mean difference within one trace.\(^3\)

2.3 Key ranking/enumeration

Performing a TA results in probabilities (logarithmic when Eq. (4) is used) for each candidate for a key portion (e.g., a byte). The question is then how difficult it would be to perform a search for the full key, e.g., when 16 different TAs are performed each of which for a byte of a round key of an AES encryption. The first problem is to find an algorithm which searches through the key space in an optimal way based on the probabilities resulting from the TAs. A key enumeration algorithm is presented in [31] that performs such a key search. Unfortunately, launching an actual attack is very time-consuming and might be infeasible if the remaining entropy is too high. Thus, to estimate how difficult an attack might be in the future or with higher computational power, an algorithm is needed to estimate the rank of the correct key in a security evaluation scenario. Such a ranking algorithm is presented in [32]. This algorithm works by carving boxes in the key space thereby approximating the volume of the key space with higher probability than the correct key. The algorithm continues until all boxes defined by the sub-key candidates are processed or a given tightness of the bounds is achieved.

For the evaluation of our experiments, we used Algorithm 2.1 introduced in [13]. Suppose that the target key is split into \( N_p \) portions, e.g., in 16 parts in case of the AES-128 round key. The algorithm operates on \( N_p \) histograms generated over the probabilities (or log Pr) as the result of \( N_p \) template attacks, given the set of measurements \( Y \). To this end, the distance between minimum and maximum of the entire probabilities (or log Pr) is divided into \( N_{\text{bin}} \) bins.

\(^3\) #1.
For each of the $N_p$ key parts, the bins’ values in the respective histogram $H_i$ are incremented for each key candidates probability falling in the respective range. This is done for each of the $N_p$ key parts and thus results in $N_p$ histograms. As input, the algorithm receives such histograms $H_i \leq i \leq N_p$ and the probability of the correct key $Pr[k^*|Y]$ given the measurements $Y$. The histograms are iteratively convoluted to each other, and the correct key rank is estimated by summing up the values in the bins representing higher probabilities than that of the correct key. By increasing $N_{\text{bin}}$, the bounds of the estimation can be tightened. This algorithm offers a very fast estimation of the remaining entropy, which is the logarithm to the base 2 of the rank, with tight bounds. As we later use the average ranking returned by the algorithm to calculate the entropy, it is more precisely the remaining guessing entropy [30] which directly relates to the average remaining workload of the side-channel attacker. However, the results of a ranking algorithm are only relevant if there is a corresponding key enumeration algorithm for which the correct key would achieve the calculated rank. In this case, the corresponding enumeration algorithm was presented by Poussier et al. [25].

Algorithm 2.1: Rank estimation [13]

Input: number of key portions $N_p$, histograms $H_i \leq i \leq N_p$, probability $Pr[k^*|Y]$

Output: estimated rank of $k^*$

1. $H = H_i$
2. for $i \in \{2, \ldots, N_p\}$ do
3. \hspace{1em} $H = \text{conv}(H, H_i)$
4. return $\sum_{j \geq \text{bin}(Pr[k^*|Y])} H(j)$

3 Practical experiments

3.1 Target

The target of our experiments is an implementation of the AES block cipher on an ASIC prototype which has been manufactured in a 40 nm technology and is bonded into a JLCC68 package. Each ASIC sample contains 7 AES cores which are synthesized from the same RTL design and thus have the same netlist (except for the adjustment of drive strengths) but differ in their placement and routing. The cores are placed next to each other in a defined area as illustrated by the different colors in Fig. 1.

The underlying implementation follows a byte-serial architecture, i.e., only one instance of the S-box is implemented. Trivially, in order to hold the cipher state, the design contains a 128-bit register (marked as Data Reg in Fig. 2) where each byte can be addressed individually. The S-box, which is based on Canright’s design [5], is split up by two registers Z and C before and after the inversion which enables pipelining the operations. After the plaintext is byte-serially loaded into the Data Reg, the SubBytes-Operation is performed byte-wise in a particular order fitting to ShiftRows operation. This is enabled by the pipeline structure formed by the registers in the S-box module. Since the corresponding SubKey byte is XORed to the S-box input, in total the AddRoundKey, SubBytes and ShiftRows operations are performed in 18 clock cycles. Afterwards, MixColumns is initiated by storing four bytes of one column into the MixColIn register. The Quarter...
MixColumn module calculates one byte of the MixColumns output, which is stored back into the Data Reg. By rotating the MixCol In, other bytes of the MixColumns output are calculated. In total, the entire MixColumns operation is performed in 32 clock cycles (8 clock cycles per columns).

Further, the same S-box module is used by the KeySchedule module (not shown in Fig. 2) to calculate the next SubKey. Hence, 6 clock cycles are spent during the KeySchedule to perform four required S-box lookups, and 12 clock cycles for the XOR operations on the remaining SubKey bytes. In sum, except the last one (which only needs 36 clock cycles due to the missing MixColumns operation), every cipher round (including the KeySchedule) needs 68 clock cycles, and a full encryption terminates in 648 clock cycles excluding the required clock cycles to load the plaintext bytes and send out the ciphertext bytes.

For practical measurements, we made use of a single toolkit board which hosts our packaged ASIC samples by means of a socket. In order to measure SCA leakages of each ASIC sample, we just exchanged the chip. The SCA traces have been recorded by a Teledyne-Lecroy HD06054 digital sampling oscilloscope with a sampling rate of 1.25 GS/s. For the entire measurements, the AES core was clocked at 4 MHz by an internal oscillator. After some initial tests, we decided to use a Tektronix TC-2 AC current probe placed in the VDD path, since it provided measurements with less noise compared to measuring the voltage drop over a shunt resistor. Additionally, the output signal of the current probe was amplified by a Mini-Circuits ZFL-1000LN+ amplifier (with 20 dB gain).

We had access to 11 ASIC samples of the fabricated design. For each of the 7 cores in every ASIC sample, we recorded 10 million profiling traces with random plaintext and random key. Additionally, for each (core, chip) combination, we collected 1000 sets of attack traces, each of which containing 1000 traces measured for a fixed (but arbitrary selected) key while plaintext was provided randomly. In other words, we collected 10 million profiling traces and 1 million attack traces while each 1000 attack traces belong to a unique key. This enables us to first examine under which model (to define the categories in TAs) the measured traces show high dependency to the intermediate values. Then, we can apply this model in inter-chip TAs to evaluate how manufacturing variability and the setup affect the portability of the templates. More precisely, by inter-chip attacks we perform TAs on core \( x \) of chip \( y \) while the profiles have been made using the traces measured from the same core \( x \) of chip \( z \neq y \). We further evaluate the influence of placement and routing of the target core by performing intra-chip attacks. It means that we conduct TAs on core \( x \) of chip \( y \) using the profiles constructed from core \( z \neq x \) of the same chip \( y \).

3.2 Model selection

Since the AES architecture processes the S-boxes serially, the update of the Data Reg is an obvious candidate for an 8-bit model. At the beginning of the first cipher round, the register contains the plaintext which will then be overwritten by the S-box lookup in the ShiftRows order. Therefore, the resulting model is the HD between such consecutive values, i.e., \( \text{HW}(P_i \oplus SR_i) \), with \( P_i \) and \( SR_i \) the \( i \)-th byte of plaintext and ShiftRows output, respectively. Considering the largest combinatorial circuit of the design (i.e., the S-box), the HD of consecutive S-box output values again in ShiftRows order \( \text{HW}(SR_i \oplus SR_{i+1}) \) is also expected to represent a valid model. There are also \( C \) and \( Z \) registers in the S-box module (see Fig. 2) which enable the pipelining. Thus, we also examined the HD of consecutive values in these two registers, i.e., \( \text{HW}(C_i \oplus C_{i+1}) \) and \( \text{HW}(Z_i \oplus Z_{i+1}) \) ShiftRows order. For completeness, we also added the HW of the S-box output \( \text{HW}(SB_i) \) to the list of our considered models. Another large combinatorial circuit in the underlying AES core is the Quarter MixColumns module. Since it processes 32-bit key-dependent intermediate values, we did not consider its leakage into our list. In addition to the HW in the aforementioned models, we considered their pure 8-bit values as well, i.e., without the HW operator.

In order to compare the considered models, by considering a single S-box lookup (i.e., one \( i \) index in the aforementioned models) we estimated the SNR following the concept illustrated in [18]. The result are shown in Fig. 3 and indicate that the HD between the plaintext and the ShiftRows output stands out with the highest SNR of 0.68, more than twice as high as the second best model with 0.32. The same is observed for other S-box lookups, i.e., other \( i \) indices. For each byte, the best model depends on only one key byte. This implies that using such a model in an attack, since the

![Fig. 3 SNR of different models for one S-box lookup](image-url)
plaintext is known we have a direct 8-bit key candidate in comparison with the second best model \(HD(SR_i, SR_{i+1})\), where two key bytes should be guessed. All models where the HW operator is not used showed smaller SNR compared to their corresponding HW model.

### 3.3 Attack in worst-case scenario

When a single device is used for both profiling and attack, it defines a baseline of what is the best result which can be achieved by an attacker, i.e., the worst-case scenario with respect to vulnerability. Consequently, we use this settings by analyzing a unique core in a single chip for profiling and tuning the parameters for the subsequent attacks.

#### 3.3.1 Points of interest

Another important step after choosing a model is the selection of Points of Interests (POIs) corresponding to the selected model. Since we do not apply a dimensionality reduction algorithm, we chose our POIs based on the SNR. We chose either the \(n\) points with the highest SNR or additionally consider a minimum distance of \(d\) time samples between any two selected points. This can efficiently be performed by first calculating the templates for a high number of POIs, e.g., 200 points, and keeping them ordered by SNR. Then, in the attack phase, we can pick the points which fulfill our requirements, e.g., the 20 points with highest SNR and minimum distance of 3 and adjust the templates. Such an adjustment is very efficiently done by copying the corresponding elements of the mean vector and the covariance matrix. Only the inverse of the covariance matrix needs to be recalculated.

As stated in Sect. 3.2, we made use of HD between plaintext and ShiftRows output as the model to build the templates. For the selection of POIs, we examined different parameters to find an optimal combination. We checked the number of points from 200 down to 100 in steps of 25 and from 100 down to 5 in steps of 5 points, always selecting the points with the highest SNR. Our experiments indicate that 100 points lead to the best result for our targeted chip. We also tested the cases with a minimum distance distance between the points from 2 up to 5 points, which did not improve the attacks. Hence, we build our templates by means of 100 points exhibiting the highest SNR for the model \(HW(P_i \oplus SR_i)\). The POIs were selected on Chip 1 on Core 1 and then kept for the remaining tests.

#### 3.3.2 Influence of environmental noise

In order to quantify the influence of variations in the measurement setup including the temperature, we first collected the profiling and attack traces for all cores in each chip directly after each other. We refer to these set of measurements as ‘old’. This means that the setup ran with different chips for many days and has been dis- and reconnected multiple times between the measurements (to swap the chips). After finishing the measurements for all cores and all chips, we recorded one more set of profiling traces for the first core of the first chip, which we recall as ‘new.’ This means that (as given in Sect. 3.1) we have 1000 sets of attack traces (each containing 1000 traces for a fixed key), while two sets of ‘old’ and ‘new’ profiling traces are available (each consisting of 10 million traces).

We have considered four cases to perform the attacks; two cases trivially correspond to ‘old’ and ‘new’ profiling traces. This is repeated by applying the mean adjustment explained in Sect. 2.2.2. For each case, we performed 1000 different attacks and applied the key ranking algorithm given in Sect. 2.3 to obtain the remaining entropy. Figure 4 shows the average of remaining entropy for all four cases over the number of used attack traces.

Without mean adjustment, the attack does not work well when using the ‘new’ set of profiling traces. It achieves an average remaining entropy of 113 bits after 1000 attack traces. In comparison, when the ‘old’ profiling traces are used, the attack needs 830 traces to achieve a remaining entropy of less than 1 bit. The entropy already reaches 8 bits after 350 traces, where the correct key can be found in a space of \(2^8\) by an enumeration algorithm, e.g., [25]. When the mean is adjusted, the attacks in both cases are significantly improved. The remaining entropy of less than 1 bit is achieved

![Fig. 4 Remaining entropy of attacks using ‘old’ and ‘new’ profiling traces with and without mean adjustment (MA), averaged over 1000 attacks](image-url)
after 120 traces using the ‘old’ profiling traces and after 140 by the ‘new’ set. Hence, for the following experiments we assume that the differences caused by the measurement setup are mostly compensated by mean adjustment.

### 3.4 Inter-chip attack

After assessing the worst-case scenario, we proceed with the real-world scenario where the training and attack traces do not belong to the same device. To this end, we concentrated on a unique core in all 11 chips. In order to get an intuition about the degree of the variability between different chips, Fig. 5a shows the corresponding 11 mean traces estimated for a certain category of the underlying model, i.e., \( \text{HW}(P_i \oplus S R_i) \). For comparison, Fig. 5b presents the mean traces for all 9 different categories belonging to a single chip. This indeed is the signal that we are trying to exploit in the attacks. As the figures show, the variability between the chips is greater than the actual exploitable signal.

![Fig. 5](image)

**Fig. 5** Mean traces of one clock cycle, categories defined by the model \( \text{HW}(P_i \oplus S R_i) \)

![Fig. 6](image)

**Fig. 6** Remaining entropy of inter-chip attacks, a unique core on different chips, using profiling traces of a certain chip, attack traces of all 11 chips, averaged over 1000 attacks. Black, dashed line is with identical profiling and attack chip

![Fig. 7](image)

**Fig. 7** Number of required traces to achieve a remaining entropy of less than 1 bit in inter-chip attacks on a unique core, with mean adjustment, averaged over 1000 attacks
Figure 6a shows the result of the attacks on a unique core on all chips using the templates build based on a single chip. Directly applying the templates to the attack traces results in widely different results even using 1000 attack traces. This indeed confirms the assumed problems of transferring the templates to other chips. Therefore, as formerly discussed, the mean adjustment technique (see Sect. 2.2.2) should be applied during the attack phase, which can drastically improve the results. Doing so, the remaining entropy for all attacks similarly reached 0 bit after around 110–120 attack traces, independent of whether the profiling and attack devices are different or the same (see Fig. 6b). This confirms that applying the mean adjustment not only compensates for the differences in the measurement setup and environmental noise but also for the manufacturing variation of the chip.

Subsequently, by concentrating on a single core we repeated this scenario for all combinations, i.e., all chips for profiling and all chips to measure the attack traces. Figure 7 lists the number of traces required in attacks to achieve a remaining entropy of less than 1 bit for all combinations of profiling and attack chips. It clearly highlights the influence of the measurement variations. While the attacks on, e.g., chip 1 and 7 seem to be successful with 100 to 120 traces, the attacks on chip 8 seem to strongly deviate from the others and require 160 to 200 traces. Also, the templates built based on chip 11 seem to perform worse compared to all other cases.

It is noteworthy that we have repeated the same experiments on all other 6 AES cores of the targeted chips. Due to the their similarity to the presented results, we omit showing the corresponding outcomes.

3.5 Intra-chip attack

Another interesting aspect is whether cores with the same architecture and even the same netlist exhibit similar leakage characteristics when placed and routed differently. To examine this, we performed template attacks on all cores using the templates built based on a certain core. Ordinarily performing the attacks did not show a successful result. When the attack core is not the same as the profiling core, the remaining entropy is still higher than 100 bits, as can be seen in Fig. 8a. Applying mean adjustment makes the key recovery feasible again. While the attack on the same core as profiling needs around 110 traces to reach the entropy of 0, between 350 and 1000 traces are needed to reach the same remaining entropy when attacking other cores (see Fig. 8b). Hence, a major component of the difference between the cores’ leakage is a DC offset which gets compensated by the mean adjustment. It also means that the significant components of the power consumption are only partially affected by the routing, and the relative difference between the values is still approximated.

Note that another possible compensation factor is to additionally adjust the standard deviation [20]. This might be useful to decline the difference between the cores’ leakage characteristics even more. In contrast to mean adjustment, the variances should be estimated for each category of the underlying model. This leads to a low number of samples for each category since the aim in TAs is to make use of a very low number of attack traces. Hence, the variance adjustment is not necessarily helpful if the number of attack traces is limited, as it is the case in our experiments.

4 Conclusion

We have shown that template attacks are still a high risk for integrated circuits manufactured in small technology nodes like 40 nm. While the manufacturing variation in our samples
clearly leads to variation in the power consumption even more than the actual data-dependent leakage, this can be easily accounted for by adjusting the mean of the attack measurements. This way the attacks on different chips achieved nearly the same performance as on the profiling circuits. Our results\(^9\) differ from Renauld et al. [27] who concluded that template attacks will be very challenging for small technology nodes after testing it on a 65 nm prototype with no internal registers around the s-box. It highlights the importance of evaluating the whole cipher design or at least putting registers around the circuits as the I/O pins can be a major source of noise.

Additionally, we have shown that template attacks on the same RTL design but with different routing seem to be possible with mean adjustment. Although these do not perform as good as on the same core, the attack is still feasible.

4.1 Future work

In this work, we intentionally kept the conducted template attack as basic as possible, to better show the variation introduced by the manufacturing. Aside from the mean adjustment, we did not apply any other preprocessing. However, often some form of dimensionality reduction is performed on the traces to reduce the number of samples points and to make the attack more robust against noise. One future aspect might be to also consider methods like PCA [1] or LDA [29].\(^{10}\) Even smaller technologies can further increase the variability and there is also the question whether at some technology node other effects might introduce variabilities which cannot be easily corrected.

Based on the results of our attacks on different placement and routing, it might be interesting to see whether it is possible to use devices which are of the same chip family but not identical for profiling, assuming that these use the same RTL design for the underlying cipher.

Acknowledgements

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Archambeau, C., Peeters, E., Standaert, F., Quisquater, J.: Template attacks in principal subspaces. In: CHES 2006, Lecture Notes in Computer Science, vol. 4249, pp. 1–14. Springer (2006)
2. Bellizia, D., Djukanovic, M., Scotti, G., Trifiletti, A.: Template attacks exploiting static power and application to CMOS lightweight crypto-hardware. Int. J. Circuit Theory Appl. 45(2), 229–241 (2017)
3. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: CHES 2004, Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer (2004)
4. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures—profiling attacks without pre-processing. In: Cryptographic Hardware and Embedded Systems—CHES 2017 Proceedings, Lecture Notes in Computer Science, vol. 10529, pp. 45–68. Springer (2017)
5. Canright, D.: A very compact s-box for AES. In: Cryptographic Hardware and Embedded Systems—CHES 2005 Proceedings, Lecture Notes in Computer Science, vol. 3659, pp. 441–455. Springer (2005)
6. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Cryptographic Hardware and Embedded Systems—CHES 2002, Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer (2002)
7. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Smart Card Research and Advanced Applications CARDIS 2013, Lecture Notes in Computer Science, vol. 8419, pp. 253–270. Springer (2013)
8. Choudary, O., Kuhn, M.G.: Template attacks on different devices. In: COSADE, Lecture Notes in Computer Science, vol. 8622, pp. 179–198. Springer (2014)
9. Djukanovic, M., Bellizia, D., Scotti, G., Trifiletti, A.: Multivariate analysis exploiting static power on nanoscale CMOS circuits for cryptographic applications. In: AFRICACRYPT, Lecture Notes in Computer Science 10239, pp. 79–94 (2017)
10. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.T.M.: On the power of power analysis in the real world: a complete break of the keeloq code hopping scheme. In: Wagner, D.A. (ed.) Advances in Cryptology—CRYPTO 2008 Proceedings, Lecture Notes in Computer Science, vol. 5157, pp. 203–220. Springer (2008)
11. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Cryptographic Hardware and Embedded Systems—CHES 2001, Lecture Notes in Computer Science, vol. 2162, pp. 251–261. Springer (2001)
12. Gierlichs, B., Lemié-Rust, K., Paar, C.: Templates vs. stochastic methods. In: Cryptographic Hardware and Embedded Systems—CHES 2006 Proceedings, Lecture Notes in Computer Science, vol. 4249, pp. 15–29. Springer (2006)
13. Glowacz, C., Grosso, V., Poussier, R., Schütt, J., Standaert, F.: Simpler and more efficient rank estimation for side-channel security assessment. In: Fast Software Encryption FSE 2015, Lecture Notes in Computer Science, vol. 9054, pp. 117–129. Springer (2015)
14. Hospodar, G., Gierlichs, B., Mulder, E.D., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. J. Cryptogr. Eng. 1(4), 293–302 (2011)
15. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: CRYPTO 1999, Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999)
16. Lerman, L., Poussier, R., Botttempi, G., Markowitz, O., Standaert, F.: Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In: Constructive Side-Channel Analysis and Secure Design, COSADE, 2015., Lecture Notes in Computer Science, vol. 9064, pp. 20–33. Springer (2015)

\(^9\) \#2.

\(^{10}\) \#2.
17. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Security, Privacy, and Applied Cryptography Engineering SPACE 2016 Proceedings, Lecture Notes in Computer Science, vol. 10076, pp. 3–26. Springer (2016)

18. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks—Revealing the Secrets of Smart Cards. Springer, Berlin (2007)

19. Martinasek, Z., Hajny, J., Malina, L.: Optimization of power analysis using neural network. In: Smart Card Research and Advanced Applications CARDIS 2013, Lecture Notes in Computer Science, vol. 8419, pp. 94–107. Springer (2013)

20. Montminy, D.P., Baldwin, R.O., Temple, M.A., Laspe, E.D.: Improving cross-device attacks using zero-mean unit-variance normalization. J. Cryptogr. Eng. 3(2), 99–110 (2013)

21. Moos, T., Moradi, A., Richter, B.: Static power side-channel analysis of a threshold implementation prototype chip. In: Design, Automation and Test in Europe Conference and Exhibition, DATE 2017, pp. 1324–1329. IEEE (2017)

22. Moradi, A.: Side-channel leakage through static power—Should we care in practice? In: Cryptographic Hardware and Embedded Systems—CHES 2014 Proceedings, Lecture Notes in Computer Science, vol. 8731, pp. 562–579. Springer (2014)

23. Moradi, A., Kasper, M., Paar, C.: Black-box side-channel attacks highlight the importance of countermeasures—an analysis of the xilinx virtex-4 and virtex-5 bitstream encryption mechanism. In: Topics in Cryptology—CT-RSA 2012—The Cryptographers’ Track at the RSA Conference 2012, Lecture Notes in Computer Science, vol. 7178, pp. 1–18. Springer (2012)

24. Oswald, D., Paar, C.: Breaking mifare desfire MF3ICD40: power analysis and templates in the real world. In: Cryptographic Hardware and Embedded Systems—CHES 2011 Proceedings, Lecture Notes in Computer Science, vol. 6917, pp. 207–222. Springer (2011)

25. Poussier, R., Standaert, F., Grosso, V.: Simple key enumeration (and rank estimation) using histograms: An integrated approach. In: Cryptographic Hardware and Embedded Systems—CHES 2016 Proceedings, Lecture Notes in Computer Science, vol. 9813, pp. 61–81. Springer (2016)

26. Pozo, S.M.D., Standaert, F., Kamel, D., Moradi, A.: Side-channel attacks from static power: When should we care? In: Proceedings of the 2015 Design, Automation and Test in Europe Conference and Exhibition, DATE 2015, pp. 145–150. ACM (2015)

27. Renaud, M., Standaert, F., Veyrat-Charvillon, N., Kamel, D., Flan- dre, D.: A formal study of power variability issues and side-channel attacks for nanoscale devices. In: Advances in Cryptology—EUROCRYPT 2011—30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Lecture Notes in Computer Science, vol. 6632, pp. 109–128. Springer (2011)

28. Schneider, T., Moradi, A.: Leakage assessment methodology—a clear roadmap for side-channel evaluations. In: Cryptographic Hardware and Embedded Systems—CHES 2015 Proceedings, Lecture Notes in Computer Science, vol. 9293, pp. 495–513. Springer (2015)

29. Standaert, F., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Cryptographic Hardware and Embedded Systems—CHES 2008 Proceedings, Lecture Notes in Computer Science, vol. 5154, pp. 411–425. Springer (2008)

30. Standaert, F., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Advances in Cryptology—EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 443–461 (2009)

31. Veyrat-Charvillon, N., Gérard, B., Renaud, M., Standaert, F.: An optimal key enumeration algorithm and its application to side-channel attacks. In: Selected Areas in Cryptography SAC 2012, Lecture Notes in Computer Science, vol. 7707, pp. 390–406. Springer (2012)

32. Veyrat-Charvillon, N., Gérard, B., Standaert, F.: Security evaluations beyond computing power. In: Advances in Cryptology—EUROCRYPT 2013 Proceedings, Lecture Notes in Computer Science, vol. 7881, pp. 126–141. Springer (2013)

33. Wong, H.P., Frank, D.J., Solomon, P.M., Wann, C.H.J., Welser, J.J.: Nanoscale CMOS. Proc. IEEE 87(4), 537–570 (1999)

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.