Near Collision Side Channel Attacks

Barış Ege[1], Thomas Eisenbarth[2], and Lejla Batina[1]

[1] Radboud University, Nijmegen
The Netherlands
[2] Worcester Polytechnic Institute,
Worcester, MA, USA

Abstract. Side channel collision attacks are a powerful method to exploit side channel leakage. Otherwise than a few exceptions, collision attacks usually combine leakage from distinct points in time, making them inherently bivariate. This work introduces the notion of near collisions to exploit the fact that values depending on the same sub-key can have similar while not identical leakage. We show how such knowledge can be exploited to mount a key recovery attack. The presented approach has several desirable features when compared to other state-of-the-art collision attacks: Near collision attacks are truly univariate. They have low requirements on the leakage functions, since they work well for leakages that are linear in the bits of the targeted intermediate state. They are applicable in the presence of masking countermeasures if there exist distinguishable leakages, as in the case of leakage squeezing. Results are backed up by a broad range of simulations for unprotected and masked implementations, as well as an analysis of the measurement set provided by DPA Contest v4.

Keywords: Side channel collision attack, leakage squeezing, differential power analysis

1 Introduction

Side channel analysis and countermeasures belong to the most active research areas of applied cryptography today. Many variants are known and all kinds of attacks and defenses are introduced since the seminal paper by Kocher et al. [13]. The assumptions for attacks, power and adversary models vary, but all together it can be said that the challenges remain to defend against this type of attacks as an adversary is assumed to always take the next step.

For example, side channel collision attacks exploit the fact that identical intermediate values consume the same power and hence similar patterns can be observed in power/EM measurements. More in detail, an internal collision attack exploits the fact that a collision in an algorithm often occurs for some intermediate values. This happens if, for at least two different inputs, a function within the algorithm returns the same output. In this case, the side channel traces are assumed to be very similar during the time span when the internal collision persists. Since their original proposal [21], a number of works have improved on various aspects of collision attacks, such as collision finding [5] or effective key recovery [10].

There are also different approaches in collision detection. Batina et al. introduce Differential Cluster Analysis (DCA) as a new method to detect internal collisions and extract keys from side channel signals [2]. The new strategy includes key hypothesis testing and the partitioning step similar to those of DPA. Being inherently multivariate, DCA as a technique also inspired a simple extension of standard DPA to multivariate analysis. The approach by Moradi et al. [17] extends collision attacks by creating a first order (or higher order in [15]) leakage model and comparing it to the leakage of other key bytes through correlation. The approach is univariate only if leakages for different sub-keys occur at the same time instance, i.e. for parallel implementations, as often found in hardware. When software implementations are considered, these two sensitive values would leak in different times, therefore other papers pursued the possibility to pursue a similar attack for software implementations in a bivariate setting [8, 23]. Although finding the exact time samples which leak information about the intended intermediate variables increases the

attack complexity, this type of attacks are especially favourable when the leakage function is unknown, or it is a non-linear function of the bits of the sensitive variable [10].

In general, it is desirable for attacks to apply to a wide range of leakage functions. Some strategies are leakage model agnostic, e.g. Mutual Information Analysis [11]. In contrast to this assumption-less leakage model approach, there is also an alternative in choosing a very generic model as in stochastic models approach [20]. We follow this direction in terms of restricting ourselves to leakages that are linear functions of the contributing bits. Nevertheless, in our scenario this is considered merely as a ballpark rather than a restriction.

When univariate attacks are considered such as the one that is proposed in this work, the best way to mitigate is to implement a masking scheme. However, one of the biggest drawbacks of masking schemes is the overhead introduced into implementations. Recently there has been a rising interest in reducing the entropy needed and thereby the implementation overhead by cleverly choosing masks from a reduced set. These approaches are commonly referred to as Low entropy masking schemes (LEMS) or leakage squeezing. In fact, LEMS are a low-cost solution proposed to at least keep or even enhance the security over classical masking [7, 18, 19]. Since the proposal, LEMS have been analyzed from different angles, including specific attacks [24], a detailed analysis of the applicability of made assumptions [12] and problems that may occur during its implementation [16]. Attention to LEMS has been stipulated to a specific version of LEMS, the Rotating S-box Masking (RSM) [18], since it has been used for both DPA contest v4 and v4.2 [3].

Our Contributions The contribution of this work can be summarised as follows:

- We introduce a new way of analysing side channel measurements which is void of strong assumptions on the power consumption of a device.
- The attack that we propose is a non-profiled univariate attack which only assumes that the leakage function of the target device is linear.
- We further extend this idea to analyse a low entropy masking scheme by improving on [24], and we show that our technique is more efficient to recover the key than generic univariate mutual information analysis.
- The proposed attack is applicable to any low entropy mask set that is a binary linear code [4].

Structure The rest of the paper is structured as follows. Section 2 introduces the notation used throughout the work and also the ideas in the literature that leads to our new attack. Section 3 introduces the near collision attack and present simulated results in comparison to other similar attacks in the literature. Section 4 introduces the extension of our idea to a low entropy masking scheme together with a summary of the previous work that it is improved upon. This section also presents comparative results of the extended attack and other attack similar to it in the literature, and a discussion on the attack complexity. Finally, Section 5 concludes the paper with some directions for further research.

2 Backgound & Notation

In this section we briefly summarize side channel attacks and also introduce the notation used throughout the paper.

Side channel analysis is a cryptanalysis method aiming to recover the secret key of a cryptographic algorithm by analyzing the unintended information leakages observed through various methods. In this work, we focus on the information leakage on the power consumption or

electro-magnetic leakage of an implementation. Further, we use the Advanced Encryption Standard (AES) to explain our new attack and run experiments as it is a widely deployed crypto algorithm around the world. This ensures comparability with other works in the literature that use AES for presenting results, but does not hinder generalization to other block ciphers in a natural way.

Correlation based power or EM analysis (CPA) against AES implementations usually focuses on the output of the S-box operation which is the only non-linear element of the algorithm. This non-linearity ensures a good distinguishability between the correct and incorrect key guesses for CPA; the correlation between the observed and the predicted power or EM leakage will be (close to) zero if the key guess is incorrect, due to the highly nonlinear relation between the predicted state and the key. To run a CPA the analyst observes the power (or EM) leakages of the device for each input $x \in X$ and stores it as an observed value $o^x \in O^X$. The next step is to reveal the relation between o^x and x through estimating the power consumption of the target device. Assume that the analyst would like to estimate power consumption with the Hamming weight function ($\mathrm{HW}(x)$) which returns the number of ones in the bit representation of a given variable. In this case, the power estimation for the input value x becomes $P(x, k_g) = \mathrm{HW}(S(x \oplus k_g))$, where k_g is a key guess for the part of the key related to x. Proceeding this way, the analyst forms 256 sets $P_{k_g} = \{P(x, k_g) : x \in X\}$ from the known input values $x_i \in X$ for each key guess $k_g \in \mathbb{F}_2^8$. What remains is to compare the estimated power consumptions P_{k_g} with the set of observations O^X on the power consumption through a distinguisher, in this case through computing the Pearson correlation coefficient $\rho(P_{k_g}, O^X),\ \forall k_g \in \mathbb{F}_2^8$. If the analyst has sufficient data and if the modelled leakage P is close enough to the actual leakage function L of the device (i.e. a linear representative of L), then the correct key k_c should result in a distinguishing value for the Pearson correlation when compared to the wrong key guesses k_w. In case P is not a linear representative of L however, then the correct key may not be distinguishable with this technique. Therefore, the choice of power model determines the strength of CPA.

Collision attacks aim to amend this problem by removing the requirement to estimate L in an accurate manner. Linear collision attacks against AES use the fact that if there are two S-box outputs equal to each other, then their power consumption should be the same [5]. If two S-box outputs for inputs x_i and x_j are equal to each other, then

$$S(x_i \oplus k_i) = S(x_j \oplus k_j) \tag{1}$$

$$\Rightarrow \quad x_j \oplus k_j = x_j \oplus k_j \tag{2}$$

$$\Rightarrow \quad x_i \oplus x_j = k_i \oplus k_j \tag{3}$$

when S is an injective function. Since the AES S-box is bijective, collisions as above reveal information about the relation of a pair of key bytes. Detecting collisions can be a challenging task, and therefore Moradi et al. [17] proposes to use Pearson correlation for collision detection by comparing pairs of vectors which have a fixed difference in their input bytes, which in turn represents the difference between the corresponding key bytes as explained above. After running a linear collision attack (referred to as the '*correlation enhanced collision attack*'), the analyst can reduce the key space radically and solve the remaining linear system of equations to recover the entire key. Hence the only challenge remains to be finding the time instances where the targeted leakages occur, which can be a time consuming task depending on the amount of samples the analyst acquires for analysis.

Although the resulting work load for brute forcing the key is reduced significantly by a linear collision attack, to further reduce the work load, Ye et. al [23] proposes a new collision attack (namely the '*non-linear collision attack*') to directly recover a key byte rather than a linear relation of two key bytes. Rather than looking for a collision in the same power measurement, non-linear collision attack looks for a linear relation between the input of the S-box for a

plaintext value x and an S-box output value of another input x' which are related to the same key byte as:

$$x' \oplus k = \quad S(x \oplus k) \tag{4}$$
$$x' = S(x \oplus k) \oplus k \ . \tag{5}$$

Therefore, the collision can be tested by building a hypothesis for k and whenever a collision is detected, the correct key for that byte is immediately revealed. Even though the key byte can be recovered directly with this attack, the intrinsic problem here remains and that is the challenge to find the two leaking samples which refer to the leakage of such values. Next section presents the univariate solution to this problem which removes the requirement of strong leakage assumptions to be able to mount a side channel attack similar to side channel collision attacks.

3 Side Channel Near Collision Attack

In this chapter we introduce the univariate non-profiled attack, namely the side channel near collision attack (NCA) with an example to an AES implementation. NCA is very similar to other collision attacks in the sense that a priori knowledge of the leakage function is not required to mount it. However, unlike collision attacks proposed up until now, near collision attack exploits the existence of very similar but yet distinct values that are computed when the inputs are assumed to be selected at uniformly random from the entire set of inputs: $\mathbb{F}_2^8$. This brings up an implicit power model assumption that the power consumption should be linearly related to the bits of the sensitive value that is computed in the device. In comparison to the popular Hamming weight model, this implicit power model assumption is a much weaker one and therefore makes the attack more powerful against a wider range of platforms and devices with different leakage functions.

The main idea of a near collision attack (NCA) is to separate the measurements into two vectors and statistically compare how these two vectors are related to each other. Assuming that the S-box output leaks in the measurements, for any input byte x_0, another input value x_1 is computed for the same byte and for a key guess k_g as:

$$S(x_1 \oplus k_g) = S(x_0 \oplus k_g) \oplus \Delta(t) \tag{6}$$
$$x_1 = S^{-1}(S(x_0 \oplus k_g) \oplus \Delta(t)) \oplus k_g \tag{7}$$

where $\Delta(t)$ is an 8-bit value with a '1' at the t^{th} bit position and '0' elsewhere. If the key guess is correct, then the S-box outputs have only one bit (XOR) difference. If the key guess is not correct, then the outputs will have a random (XOR) difference in between. Note that this property holds due to AES S-box's strength against differential cryptanalysis. Proceeding in this way, one can form a pair of vectors, X_0 and X_1 such that

$$X_0 = [x_0^i \in \mathbb{F}_2^8 : i \in \{1, \ldots, 128\},\ S(x_0^i \oplus k_g) \wedge \Delta(t) = 0] \tag{8}$$

where $\wedge$ is the bit-wise AND operation, and X_1 is formed from each element of X_0 through the relation given in Eqn. (7). This way the whole set of values in $\mathbb{F}_2^8$ are separated into two vectors and now they can be used to generate a statistic for k_g which in turn can be used to distinguish the correct key from others.

For $t = 8$, the observed values corresponding to the sets X_0 and X_1 can be visualized in Figure 1 for an incorrect and a correct key guess under the assumption that the Hamming weight of a value leaks in observations. The difference between the observed values is also included in the plot for ease of comparison. As it is clearly visible from Figure 1, when the

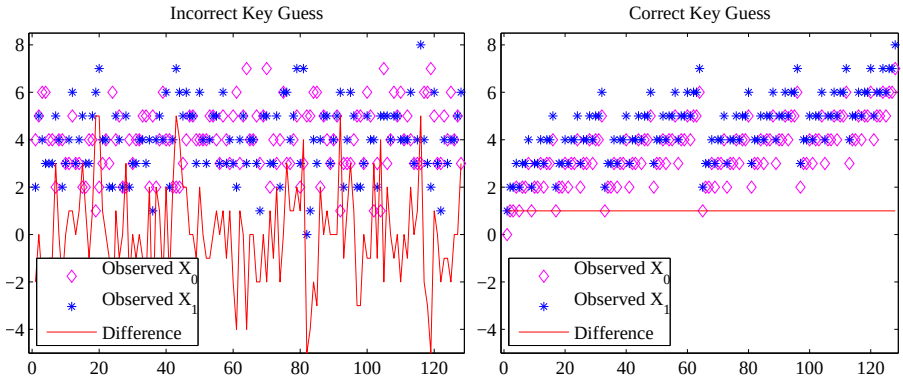


Fig. 1. Simulation values in sets X_0 and X_1 for incorrect(left) and correct(right) key guesses.

key guess is correct, there is a clear linear relation between the vectors of observed values O^{X_0} and O^{X_1} corresponding to X_0 and X_1 respectively. Therefore Pearson correlation coefficient $(\rho(O^{X_0}, O^{X_1}))$ can be used as a statistical distinguisher in this case to recover the key.

For real measurements, this attack can be implemented in a known plaintext setting by computing the mean of the observed values $\mu(O^{x_0^i})$ and $\mu(O^{x_1^i})$ for each input value x_0^i and x_1^i to reduce noise as in [17]. Furthermore, the attack can be run on larger than 128 value vectors to reduce multiple times for different values of t to compute the byte difference $\Delta(t)$, and the resulting correlation coefficients can be added together for each key guess k_g to constitute one final value to better distinguish the key.

3.1 Simulated Experiments on Unprotected AES Implementation

We have run simulated experiments to assess the capabilities of the near collision attack (NCA) and its efficiency in comparison to other similar attacks in the literature. To evaluate how our attack reacts to noise, we have fixed the number of traces and conducted experiments with various signal to noise ratio values (SNR $= \frac{var(signal)}{var(noise)}$, where signal and noise are computed as defined in [14]).

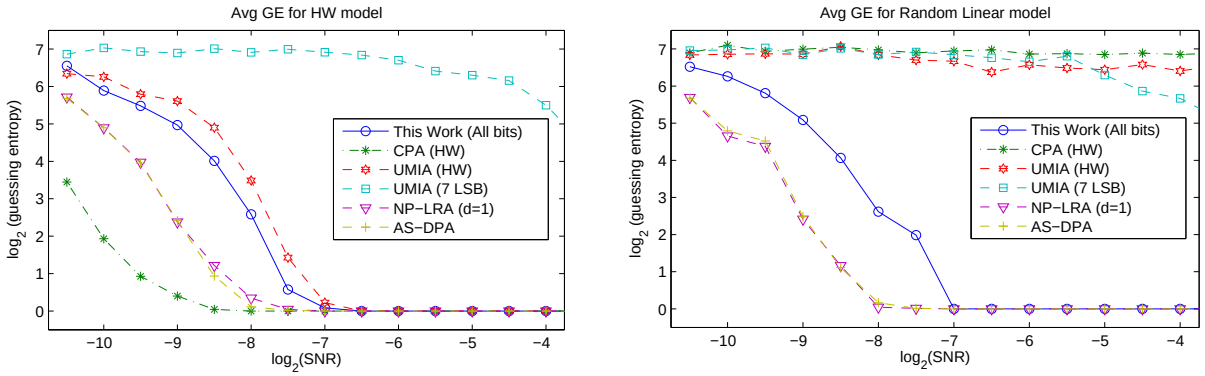
An important note here is that we have use scaled simulated values to mimic the measurements collected from an oscilloscope. Usually when simulated measurements are analysed, the fact that the simulations provide unnaturally optimistic measurements is neglected. Since this may lead to misleading simulation results which cannot be reproduced in real life, we have chosen to filter the simulated traces and scale them to the resolution of an 8-bit oscilloscope, therefore producing 256 unique values for traces. Note that, depending on the noise level the simulated traces can cover a large range of values. Therefore we have chosen to scale the values in a way such that the maximum and minimum values (128 and -127) are assigned to values $(\mu + 3 \times \sigma)$ and $(\mu - 3 \times \sigma)$ respectively, where μ is the mean, and σ is the standard deviation of the simulated traces. The rest of the values are distributed equally over the sub-ranges which are of equal size.

As to measure the robustness of our technique against different linear leakage functions, we have used two ways to compute the simulated traces:

(a) The first method computes the Hamming weight of the S-box output (HW model).
(b) The second method is a weighted linear function of the bits of the S-box output, where the weight values are picked uniformly at random in the range $[-1, 1] \subset \mathbb{R}$ (Random linear model).

For comparison, we have selected the popular non-profiling univariate attacks, namely: correlation power analysis (CPA) [6], absolute sum DPA (AS-DPA) [1], non-profiled linear regression attack (NP-LRA) [9], and univariate mutual information analysis (UMIA) [11]. We have included CPA with Hamming weight model to have a basis for comparison as it is a popular choice for doing side channel analysis. The choice of AS-DPA and NP-LRA are to have a comparison with attacks which also have weak assumptions on the leakage model; AS-DPA assumes that each bit of the sensitive variable contribute significantly to the power consumption, where NP-LRA usually limits the algebraic order of the leakage function. For this work, we have restricted the basis functions of NP-LRA to linear relations (the case $d = 1$ in [9]), so that it would be a fair comparison to our work. Furthermore, we have included the leakage model dependent UMIA with Hamming weight model (UMIA-(HW)), and the leakage model agnostic variant UMIA which measures the mutual information between the least significant 7 bits of the sensitive variable and power measurements (UMIA-(7 LSB)).

We have run the experiments with 10 000 traces to put all methods on fair ground. Note that MIA requires a large number of traces as its distinguishing ability depends on the accuracy of the joint probability distribution estimations between the sensitive variable and power traces. We have computed the guessing entropy [22] over 100 independent experiments for each SNR value considered. Figure 2 presents the results of these experiments. As it is visible in Figure 2 (a)



(a) (b)

Fig. 2. SNR vs Guessing Entropy values computed over 100 independent experiments with 10 000 traces for perfect HW leakage (a), and random linear leakage (b).

which shows results for Hamming weight leakage function, CPA has an obvious advantage over all other methods. When the second leakage function is considered however (Figure 2(b)), the attacks using relaxed assumptions on the leakage function outperforms CPA. We also clearly see that MIA cannot deal with high levels of noise as efficiently as NCA, AS-DPA and NP-LRA.

Finally, if we only consider the attacks which have fewer assumptions on the leakage function, absolute sum DPA and the non-profiled linear regression attack seems to be able to deal with noise more efficiently when compared to NCA in an unprotected setting. Section 4 explains how the near collision approach of looking for small differences in the sensitive values can lead to a significant improvement over the state of the art attack against low entropy masking schemes.

3.2 Implementation Efficiency of NCA

Although near collision attack has the advantage of having reduced assumptions on the target device, this comes at a price, namely in computation time. For each key guess, the analyst should

find the measurements which have a particular value in its corresponding plaintext. Although this can be a cumbersome operation, this does not scale up when the analyst has to run the analysis on multiple samples of each collected measurement. To give a more accurate idea on the timing cost of NCA, Table 1 summarizes the average running time of each attack that is run in the previous section. The table presents average running time of each attack on 10 000 traces. All attacks are implemented as Matlab scripts executed in Matlab 2015a run on a PC with a Xeon E7 CPU. Note that the performance numbers assume the traces to be already loaded into memory in all cases.

Table 1. Average timing results from 100 independent experiments.

Technique	Time (sec.)
NCA	5.2727
CPA (HW)	1.0285
AS-DPA	1.1568
NP-LRA (d=1)	2.7621
UMIA (HW)	1.4130
UMIA (7 LSB)	6.6153

Looking at the results presented in Table 1 and also Figure 2, AS-DPA seems to be the best choice for the analyst in the tested cases in terms of running time and the ability to deal with Gaussian noise. However, even AS-DPA and NP-LRA are more efficient in the unprotected case, to the authors' best knowledge, these techniques do not scale to univariate attacks against low entropy masking schemes.

4 Near Collision Attack Against LEMS

A rather effective countermeasure against first order attacks such as introduced in the previous sections of this work is to use a masking scheme. However, one of the biggest drawbacks of masking schemes is the overhead introduced to the implementations. Low entropy masking schemes (LEMS) are a solution proposed to keep the security of classical masking [7, 18, 19] but reducing the implementation costs significantly. In this section, we argue how near collision attack idea can be extended to low entropy masking schemes. In particular, we focus on the mask set that is also used in the DPA Contest v4 traces:

$$M_{16} = \{\texttt{00}, \texttt{0F}, \texttt{36}, \texttt{39}, \texttt{53}, \texttt{5C}, \texttt{65}, \texttt{6A}, \texttt{95}, \texttt{9A}, \texttt{A3}, \texttt{AC}, \texttt{C6}, \texttt{C9}, \texttt{F0}, \texttt{FF}\} \ .$$

4.1 Leaking Set Collision Attack

Leaking set collision attack is based on the observation that two sensitive variables which are masked with the mask set M_{16} lead to the same 16 leaking (masked) values if they are the bit-wise complement of each other [24]. Following this observation, the authors propose to compare the so-called leaking sets, the measurements corresponding to the 16 masked values, for each input x and it's pair x' computed as

$$x' = S^{-1}(S(x \oplus k) \oplus (\texttt{FF})_{16}) \ . \tag{9}$$

Once the input pairs per key guess are computed, the analyst collects the observed values O^x and $O^{x'}$ corresponding to x and x'. If the key guess is correct, O^x and $O^{x'}$ should have the same distribution. If the key guess is not correct however, the resulting distributions will differ significantly, thanks to the AES S-box's good resistance against differential attacks. For comparing

the distributions of these two sets, authors of [24] propose to use the 2-sample Kolmogorov-Smirnov (KS) test statistic. As KS test measures the distance between two distributions, the correct key guess should result in a lower KS test statistic than the incorrect key guesses do.

4.2 Leaking Set Near Collision Attack

We now define the '*leaking set near collision attack*' (LS-NCA) as a combination of the LSCA idea proposed in [24] and the near collision attack (NCA) introduced in Section 3. Leaking set near collision attack can be summarized as an extension of the idea explained in Section 4.1 to the entire mask set of M_{16}. Similarly, we use the same observation that some input values lead to the same distribution in the S-box output as a direct result of the properties of the mask set that is used. As the authors of [24] point out in their work, whenever a sensitive value x is protected with the mask set M_{16}, the value $x \oplus (\texttt{FF})_{16}$ also results in the same values after applying the mask set. A further observation on the mask set M_{16} is that it is a closed set with respect to the XOR operation. In other words, XOR of any two elements in the set M_{16} results in another element of the mask set:

$$m_i \oplus m_j \in M_{16},\ \forall m_i, m_j \in M_{16}\ . \tag{10}$$

This means that a sensitive variable x protected with the mask set M_{16}, and another sensitive variable $y(i) = x \oplus m_i,\ m_i \in M_{16}$ leads to the same masked values. It is easy to see that the property exploited in [24] is one particular case of the observation given in Eqn. (10). Therefore, rather than directly comparing two similar distributions, if one collects all data from the input values which lead to the same distribution in the same set, this will lead to an equally reliable statistical analysis with less data. One should note that once all data that contribute to the same distribution are collected together, it is no longer possible to make a comparison between different sets and expect the same distribution. Therefore, we utilize a similar approach as we have done in Section 3 and look for sensitive values with 1-bit differences for comparison.

Leaking set near collision attack can be summarized as follows:

1. Generate the (disjoint) subsets of inputs (x) of which the S-box outputs contribute to the same distribution:
$$D^{x_i}_{M_{16}} = \{x : x = S^{-1}(S(x_i) \oplus m),\ \forall\ m \in M_{16}\}\ .$$
2. Make a key guess k_g.
3. For each input byte $x \oplus k_g$ which contribute to the same distribution (e.g. $x \oplus k_g \in D^{x_i}_{M_{16}}$), collect the corresponding measurement sample in a set $O^{x_i}(k_g)$.
4. Use 2-sample Kolmogorov-Smirnov(KS) test to check how similar the distributions of $O^{x_i}(k_g)$ and $O^{x_j}(k_g)$ are, where $S(x_i) \oplus S(x_j) = \Delta(t),\ \forall t \in \{1, ..., 8\}$.
5. Store sum of all 2-sample KS test statistics for each k_g.

Note that in Step 4, only the sets which have a 1-bit difference in between are used for 2-sample KS-test statistic calculation. In fact, sets with more than one bit difference in their S-box outputs might have the same Hamming weight, which in turn leads to similar (but not the same) distributions. Therefore, we expect the correct key to lead to a large distance between the two distributions. In case of an incorrect key guess however, each of the 16 elements in the set $D^{x_i}_{M_{16}}$ will lead to 16 distinct values after the S-box, therefore resulting in a distribution which spans the entire space $\mathbb{F}_2^8$. Sets with only one bit difference however will always result in different distributions. For instance, if the device leaks the Hamming weight of a value it computes, comparing sets with more than one bit difference would introduce noise in the cumulative KS-test statistic as values $(\texttt{05})_{16}$ and $(\texttt{03})_{16}$ have a 2-bit XOR difference in between, but have

the same Hamming weight. Further note that doing the analysis on 1-bit different sensitive values limits the analysis to 64 calls to the 2-sample KS-test, therefore saves running time when the device leaks the Hamming weight of the sensitive variable. On the other hand, if the leakage function is an injection, all $\binom{16}{2} = 120$ combinations should be compared cumulatively. Here, using only the sets with 1-bit difference for comparison can be thought of a method similar to using mutual information analysis (MIA) by estimating power consumption with the Hamming weight model, since there is an implicit leakage function assumption that there is no inter-bit interaction in the leaking variable. In fact, if the leakage function is non-linear, the improvement gained through using only 1-bit different sets for comparing distributions would be less pronounced.

Unlike LSCA (outlined in Section 4.1), the cumulative test statistic now results in much smaller values for the incorrect key guesses. This is due to the fact that a wrong key guess (k_w) results in a random sampling of the set $\mathbb{F}_2^8$ and taking into account that 16 masks in M_{16} results in 16 distinct values for each sample in the set, the resulting $O^{x_i}(k_w)$ has a cardinality much closer to $|\mathbb{F}_2^8|$. However, this does not diminish the distinguishability of the correct key from other candidates. In the case where the key guess is correct (k_c), the set $O^{x_i}(k_c)$ will have around 16 unique values (the exact number can increase due to noise in the measurement setup). Now that we have a much smaller sampling of the set $\mathbb{F}_2^8$, a comparison of distinct sets $([O^{x_i}(k_c), O^{x_j}(k_c)],\ i \neq j)$ is more meaningful, and in fact the cumulative 2-sample KS-test statistic value results in a larger value than the one obtained for a wrong key guess as the distributions are definitely different.

Note that this mask set is an example of the mask sets that are generated as a linear code [4]. As binary linear codes have the intrinsic property of being closed sets with respect to the XOR operation, any mask set that is a binary linear code is vulnerable to the attack explained in this section.

4.3 Simulated Experiments on AES Implementation with LEMS

In this section we present results of our simulated experiments with different SNR values and also with two different leakage functions as it is done in Section 3.1. In our simulations, we compare our attacks to the previously proposed univariate, non-profiled attacks: univariate MIA (UMIA) and LSCA that is recalled in Section 4.1. To compare the efficiency of the attacks in terms of the expected remaining work to find the key, we use the guessing entropy metric [22]. For each experiment using a random linear model as the leakage function, we generate 8 values which are picked uniformly at random from $[1, 10] \subset \mathbb{R}$. Unlike the simulations presented in Section 3.1, we have chosen to use a linear leakage function which is slightly different, and favour the attacks assuming that each bit of the sensitive value contributes to the leakages. Although this may not always be the case in real life, we choose to use this leakage function as it is a favourable leakage model for MIA using the Hamming weight model. We present results that show the proposed technique in Section 4.2 is more efficient than MIA in terms of handling the noise in an unknown linear leakage model setting even when the leakage model favours MIA.

The experiments are carried out with various SNR values and the results are presented in Figure 3 for both Hamming weight model and the random linear leakage model we computed for each experiment. Note that the attack which assumes a linear leakage model and computes only 64 comparisons is marked as 'Linear', and the attack which computes all possible 120 comparisons is marked as 'ID' for identity model. A quick look at Figure 3 shows that, similar to the case in near collision attack proposed in Section 3, the attack is indifferent to changes in the leakage model as long as it stays linear. Moreover, if the leakage model is a random linear function of the bits of the sensitive variable, univariate MIA fails to recover the key for leakage

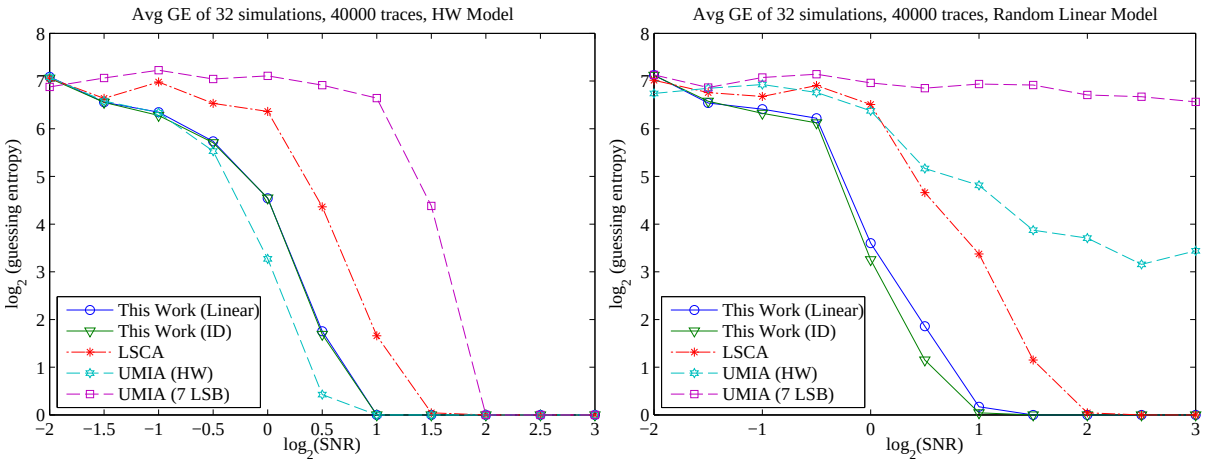


Fig. 3. Efficiency of the proposed attacks in comparison with previous works for various SNR values.

models with a high variance of its weight values. However when the leakage model follows a strict Hamming weight leakage, then univariate MIA seems to handle noise more efficiently than both of our proposed approaches to analyse the low entropy masking scheme at hand.

4.4 Experiments on DPA Contest v4 Traces

This section shows the efficiency of our attacks compared to the similar previously proposed methods in the context of a real world scenario. For reproducibility of our results, we used DPA Contest v4 traces which are EM measurements collected from a smart card having an ATMega-163 microcontroller which implements an LEMS against first and second order attacks [19]. The implementation uses the mask set M_{16} given in Section 10.

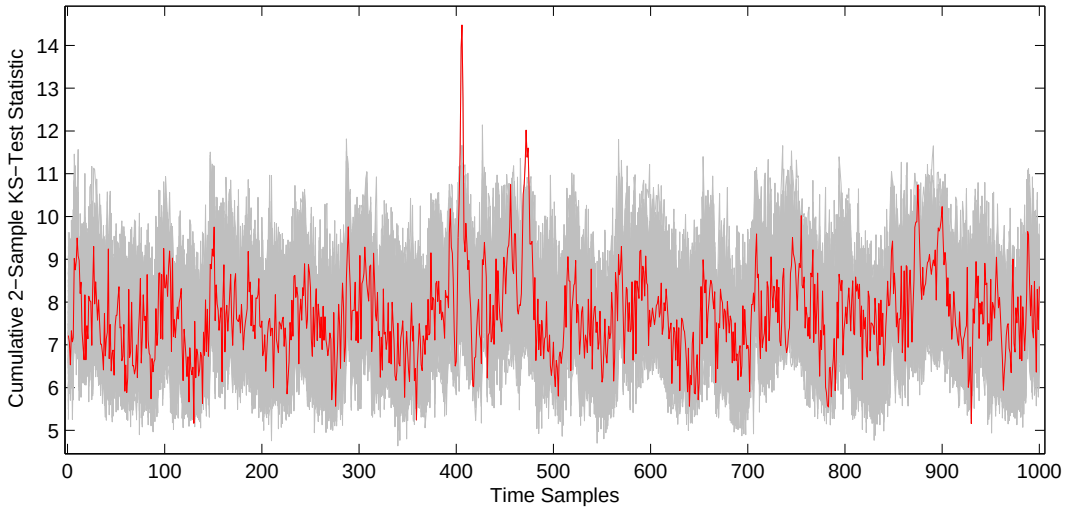


Fig. 4. Leaking set near collision attack results VS time samples.

Figure 4 presents an analysis in time domain which reveals that even with 1000 traces, it is possible to recover the key with high confidence. To further test the reliability of our technique

on the DPA Contest v4 traces, we have focused our analysis on the time samples where each of the 16 S-box outputs lead to the highest signal-to-noise ratio (SNR computed following the definition in [14]) for computing guessing entropy and the results of the analysis are presented in Figure 5.

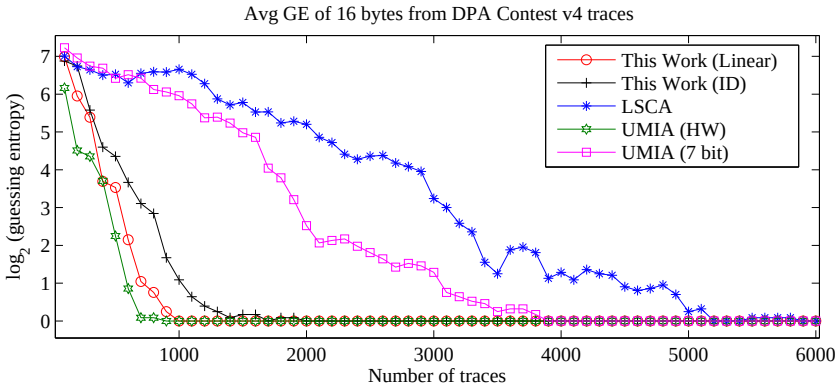


Fig. 5. Logarithm of average partial guessing entropy for various univariate attacks on DPA Contest v4 traces.

First thing to notice in the figure is that LSCA has some room for improvement even when compared to a generic univariate MIA (UMIA (7-bit) in Figure 5). On the other hand, when MIA is applied with a more accurate power model (in this case the Hamming weight model), the gap is rather large. When the leaking set near collision attack proposed in this work is considered, it is easy to see that the one which does not assume any power model ('ID') performs twice as efficient in terms of the number of traces required to recover the full key when compared to the generic univariate MIA ('7 LSB'). Moreover, when the leakage function is assumed to have a linear relation with respect to the bits of the leaking value, the results are almost identical to the ones from a univariate MIA which models the power consumption as the Hamming weight of the leaking value. One should note that Hamming weight model is rather accurate in this case as SNR values (computed with Hamming weight model) vary between 3 and 5 for the points taken into consideration for the analysis.

4.5 Implementation Efficiency of the Attack

Similar to the near collision attack, leaking set near collision attack also requires to find the traces in the measurement set which correspond to a set of input bytes. Although this operation is computationally heavy when applied to a large trace set, it does not get worse when multiple samples are needed to be analysed. The analyst can group all the traces corresponding to a leaking set and then compute 2-sample KS test statistic for each sample of a pair of leaking sets.

As in Section 3, we have run simulated experiments to assess the time required to run the proposed attacks in comparison to the other attacks run in this section. Table 2 presents the average running times of each attack applied to the chosen low entropy masking scheme. Timings presented in the table are average running times over 100 independent experiments that are run over 10 000 traces. Similar to the experiments before, all attacks are implemented as Matlab scripts executed in Matlab 2015a run on a PC with a Xeon E7 CPU. Note that the performance numbers assume the traces to be already loaded into memory in all cases.

Table 2. Average timing results from 100 independent experiments.

Technique	Time (sec.)
LS-NCA (Linear)	13.7144
LS-NCA (ID)	15.4003
LSCA	8.6176
UMIA (HW)	1.5648
UMIA (7 LSB)	7.6951

Looking at the results presented in Table 2 and taking into consideration that the leaking set near collision attacks (LS-NCA) require less number of traces, they are the strongest attacks against software implementations of LEMS.

5 Conclusions

In this work, we introduced a new way of analysing side channel traces, namely the side channel near collision attack (NCA). Unlike the collision attacks proposed in the literature, NCA is intrinsically univariate and only assumes the leakage function to be linear. Simulations show that NCA is indifferent to changes in the linear leakage function.

Furthermore, we present a new attack, leaking set near collision attack, against the low entropy masking scheme used in DPA Contest v4 [19]. This attack improves the attack proposed in [24] by fully exploiting the properties of the used mask set, and combining it with the NCA approach. As the proposed attack is univariate, it is especially of interest for software implementations of low entropy masking schemes. Simulations show that in case the leakage function diverges from a perfect Hamming weight leakage but yet stays a linear function, our attack overpowers univariate MIA.

It should be noted that not only the mask set M_{16}, but all mask sets which have a linear relation in between (as proposed in [4]) are vulnerable to the attack presented in this paper.

Application of the proposed analysis methods to non-linear leakage functions remains a research direction to follow as a future work.

References

1. Agrawal, D., Rao, J., Rohatgi, P.: Multi-channel attacks. In: Walter, C., Ko, ., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2003, Lecture Notes in Computer Science, vol. 2779, pp. 2–16. Springer Berlin Heidelberg (2003), http://dx.doi.org/10.1007/978-3-540-45238-6_2
2. Batina, L., Gierlichs, B., Lemke-Rust, K.: Differential Cluster Analysis. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2009, Lecture Notes in Computer Science, vol. 5747, pp. 112–127. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-04138-9_9
3. Bhasin, S., Bruneau, N., Danger, J.L., Guilley, S., Najm, Z.: Analysis and Improvements of the DPA Contest v4 Implementation. In: Chakraborty, R., Matyas, V., Schaumont, P. (eds.) Security, Privacy, and Applied Cryptography Engineering, Lecture Notes in Computer Science, vol. 8804, pp. 201–218. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-12060-7_14
4. Bhasin, S., Carlet, C., Guilley, S.: Theory of masking with codewords in hardware: low-weight dth-order correlation-immune Boolean functions. Cryptology ePrint Archive, Report 2013/303 (2013), http://eprint.iacr.org/
5. Bogdanov, A.: Multiple-Differential Side-Channel Collision Attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems CHES 2008, Lecture Notes in Computer Science, vol. 5154, pp. 30–44. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85053-3_3
6. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004, Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer Berlin Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-28632-5_2

7. Carlet, C., Danger, J.L., Guilley, S., Maghrebi, H.: Leakage Squeezing of Order Two. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012, Lecture Notes in Computer Science, vol. 7668, pp. 120–139. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-34931-7_8
8. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved Collision-Correlation Power Analysis on First Order Protected AES. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems CHES 2011, Lecture Notes in Computer Science, vol. 6917, pp. 49–62. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-23951-9_4
9. Doget, J., Prouff, E., Rivain, M., Standaert, F.X.: Univariate side channel attacks and leakage modeling. Journal of Cryptographic Engineering 1(2), 123–144 (2011), http://dx.doi.org/10.1007/s13389-011-0010-2
10. Gérard, B., Standaert, F.X.: Unified and Optimized Linear Collision Attacks and Their Application in a Non-profiled Setting. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems — CHES 2012, Lecture Notes in Computer Science, vol. 7428, pp. 175–192. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-33027-8_11
11. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems CHES 2008, Lecture Notes in Computer Science, vol. 5154, pp. 426–442. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85053-3_27
12. Grosso, V., Standaert, F.X., Prouff, E.: Leakage Squeezing, Revisited (2013)
13. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology. pp. 388–397. Springer-Verlag, London, UK (1999)
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
15. Moradi, A.: Statistical Tools Flavor Side-Channel Collision Attacks. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology — EUROCRYPT 2012, Lecture Notes in Computer Science, vol. 7237, pp. 428–445. Springer Berlin Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-29011-4_26
16. Moradi, A., Guilley, S., Heuser, A.: Detecting Hidden Leakages. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) Applied Cryptography and Network Security, Lecture Notes in Computer Science, vol. 8479, pp. 324–342. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-07536-5_20
17. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-Enhanced Power Analysis Collision Attack. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010, Lecture Notes in Computer Science, vol. 6225, pp. 125–139. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15031-9_9
18. Nassar, M., Guilley, S., Danger, J.L.: Formal Analysis of the Entropy / Security Trade-off in First-Order Masking Countermeasures against Side-Channel Attacks. In: Bernstein, D., Chatterjee, S. (eds.) Progress in Cryptology — INDOCRYPT 2011, Lecture Notes in Computer Science, vol. 7107, pp. 22–39. Springer Berlin Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-25578-6_4
19. Nassar, M., Souissi, Y., Guilley, S., Danger, J.L.: RSM: A Small and Fast Countermeasure for AES, Secure Against 1st and 2Nd-order Zero-offset SCAs. In: Proceedings of the Conference on Design, Automation and Test in Europe. pp. 1173–1178. DATE '12, EDA Consortium, San Jose, CA, USA (2012), http://dl.acm.org/citation.cfm?id=2492708.2492999
20. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: Rao, J., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems — CHES 2005, Lecture Notes in Computer Science, vol. 3659, pp. 30–46. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/11545262_3
21. Schramm, K., Wollinger, T., Paar, C.: A New Class of Collision Attacks and Its Application to DES. In: Johansson, T. (ed.) Fast Software Encryption, Lecture Notes in Computer Science, vol. 2887, pp. 206–222. Springer Berlin Heidelberg (2003), http://dx.doi.org/10.1007/978-3-540-39887-5_16
22. Standaert, F.X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, Lecture Notes in Computer Science, vol. 5479, pp. 443–461. Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-01001-9_26
23. Ye, X., Chen, C., Eisenbarth, T.: Non-Linear Collision Analysis. In: Saxena, N., Sadeghi, A.R. (eds.) Radio Frequency Identification: Security and Privacy Issues, pp. 198–214. Lecture Notes in Computer Science, Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-13066-8_13
24. Ye, X., Eisenbarth, T.: On the Vulnerability of Low Entropy Masking Schemes. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications, pp. 44–60. Lecture Notes in Computer Science, Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-08302-5_4