Non-Linear Collision Analysis

Xin Ye, Cong Chen, Thomas Eisenbarth

Worcester Polytechnic Institute, Worcester, MA, USA
{xye,cchen,teisenbarth}@wpi.edu

Abstract. As an unsolved issue for embedded crypto solutions, side channel attacks are challenging the security of the Internet of things. Due to the advancement of chip technology, the nature of side channel leakage becomes hard to characterize with a fixed leakage model. In this work, a new non-linear collision attack is proposed in the pursuit of the side channel distinguishers with minimal assumption of leakage behavior. The attack relies on a weaker assumption than classical DPA: it does not require a specific leakage model. The mechanism of collision generation enables independent recovery of partial keys so that for the first time the collision attack can be fairly compared with other standard side channel distinguishers. The efficiency of this attack has been verified by experiments on an unprotected microcontroller implementation of AES. Its immunity to modeling errors is confirmed through simulation of a broad range of leakage functions.

1 Motivation

Side channel attacks (SCA) such as Power and EM analysis remain as a major concern for embedded cryptographic systems. The mostly wireless connection of devices and appliances makes security and hence reliable embedded crypto engines a necessity for the entire Internet of things. Only affordable countermeasures and robust evaluation methods can assure a widespread protection against SCA. In general, SCA achieves its key recovery objective through exploring the data dependency between side channel observables and the internal state or the system. Such data dependency has usually been described with a particular leakage model by the classical Differential Power Analysis (DPA) [8] and Correlation Power Analysis (CPA) [4]. Models range from Hamming weight/distance models to more complicated toggle count models depending on the a-priori knowledge about the implementation. Consequently, the error from leakage modeling assumption or the lack of detailed a-priori knowledge can aggravate or even prevent successful attacks. Recent studies [13, 9, 6] call for generic distinguishers that do not rely on a-priori knowledge about the implementation and have minimum assumption on the leakage distribution. Although non-parametric statistic methods such as Mutual Information Analysis [7] and Kolmogorov-Smirnov test [20] are well suited to estimate the unknown leakage behavior, the cost is a huge loss of efficiency: many more measurements are needed for probability density estimation or empirical distribution comparison. Whitnall et. al. showed in [21]

that generic univariate attacks with a leakage model exist only for a very limited selection of target functions. It is indicated that profiled attacks such as template attacks [5] and stochastic modeling attacks [14] are necessary for security evaluation. Although those attacks achieve great efficiency, the requirement of the profiling stage is sometimes demanding except for evaluating labs.

An alternative side channel strategy are *side channel collision attacks* [16] where the adversary recovers the key with the combined benefit from the algebraic property and the leakage similarity of internal collisions. Another attractive feature of collisions is the *self-templating* property: instead of estimating or assuming a leakage model, leakages observed from different queries are directly compared. In other words, side channel collision attacks do not even require a leakage model. This satisfies the need of generic side channel distinguishers that assume as little about the leakage function as possible.

Contribution In this work we propose a new side channel collision attack to recover secret information without prefixing a leakage model or estimating leakage distributions. The attack derives side channel collisions between internal states that do not have a simple linear relationship. The approach allows us to collide the same partial state at two different stages, e.g. the input and output of an S-Box, and hence retrieve the secret information by exploiting the bivariate leakage samples reflecting the two stages. Results are verified experimentally and through simulation. Of independent importance is the quantitative analysis of the sensitivity of collision attacks to leakage mismatch in the colliding states. The proposed attack is efficient, immune to leakage modeling errors and robust against high inhomogeneity of the leakage behavior of non-linear collisions.

The rest of the paper is organized as follows: After a review of related prior work in collision attacks in Section 2 we explain in Section 3 how to exploit non-linear collisions for more efficient key recovery. Section 4 details on experiments as well as simulations and highlights the applicability and convenience of non-linear collision attacks.

2 Background

We briefly revisit the existing proposals in side channel collision attacks.

2.1 Collision Attack

Side channel collision attacks were introduced in [16] against DES and extended in [15] against AES. Works of [1–3, 22] further improved the collision attacks for different scenarios. They have a common definition of collisions being the same internal state computed from different inputs. A collision tells the adversary that targeted key parts satisfy certain algebraic equalities which are employed to reduce the space of valid key hypotheses. Collision attacks take advantage of both side channel leakage and the algebraic property of the cipher and hence can

recover the key with fewer traces. However, such benefit stands on two prerequisites: (1) the adversary should have chosen plaintext capability as mentioned in [15]; (2) the adversary can detect the occurrence of collision with as low probability of false positive decision as possible. This is because the algebraic property can be used only after a collision is successfully detected and it is easy to understand that using a wrongly detected collision yields misuse of invalid algebraic property and hence risks missing the correct key.

2.2 Linear Correlation Collision Attack

In [11], an interesting algorithm has been proposed to attack AES using correlation enhanced linear collision. It is different from the classical collision attack since it does not use collision *detection* to reduce the total number of valid key hypotheses. In fact, it works more like classical DPA/CPA style attacks that firstly make hypothesis and then use distinguisher to determine the correct key that actually *generates* collisions. But unlike classical DPA/CPA, the linear correlation collision attack (LCCA) does not recover each subkey directly, but instead it tests hypothesis of the difference between subkeys as shown in Figure 1(a). More specifically, if the adversary aims at recovering the difference $\Delta = k_a \oplus k_b$ between subkey k_a and k_b at byte a and b, she needs to test all possible hypotheses δ of the subkey difference. For each hypothesis δ, the adversary computes the correlation $\rho(M_a^X, M_b^{X\oplus\delta})$ between the averaged leakage trace M_a^X of the byte-a-plaintext $X_a = X$ and the averaged leakage trace $M_b^{X\oplus\delta}$ of the byte-b-plaintext $X_b = X \oplus \delta$. Upon completion of all hypotheses, the adversary makes the decision of the hypothesis that gives highest correlation, i.e. $\delta^* = \text{argmax}_\delta\{\rho(M_a^X, M_b^{X\oplus\delta})\}$. The attack works because when testing the correct hypothesis $\delta = \Delta = k_a \oplus k_b$, the Sbox outputs of the two bytes cause collisions as seen from below.

$$\begin{aligned} & X_a \oplus X_b = X \oplus X \oplus \delta = \Delta = k_a \oplus k_b \\ \Longleftrightarrow\ & X_a \oplus k_a = X_b \oplus k_b \\ \Longleftrightarrow\ & S(X_a \oplus k_a) = S(X_b \oplus k_b) \end{aligned}$$

Therefore the averaged leakage traces M_a^X and $M_b^{X\oplus\Delta}$ gives high correlation. If a wrong hypothesis $\delta \neq \Delta$ is assumed, the above equalities do not hold any more, neither have collisions be generated. Therefore a wrong hypothesis results in low correlation.

3 Non-Linear Collision Attack

Non-linear collisions take advantage of the fact that processing two internal states of the same value yields similar leakage behavior—especially for software implementations. The concept of exploitable collisions is extended so that they occur for different internal states, even if processed under different operations. We first explain the idea of generating non-linear collisions and then detail how

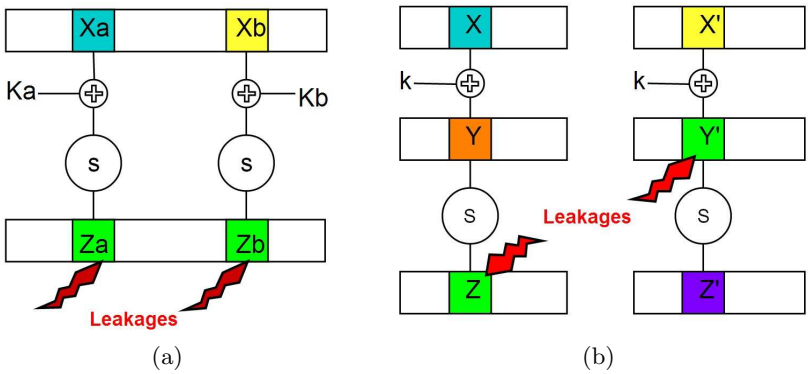


Fig. 1. Simplified schematic for Linear Correlation Collision Attack (LCCA) (a) VS Non-Linear Collision Attack (NLCA) (b). While LCCA exploits linear collision of two *different* state bytes at the same stage in the cipher round, NLCA can exploit the non-linear collision of the *same* state byte at two different stages of the cipher round.

to exploit them and use them to build a side channel distinguisher called Non-Linear Collision Attack (NLCA). Its validity, complexity and relation to other side channel attacks are also discussed.

3.1 Existence of Non-Linear Collisions

Let two internal states of the target implementation be denoted by Y and Z for the NLCA. The first state $Y = f_k(X)$ is the output of a function of the plaintext X with the secret key k. The keyed function f_k is part of the crypto algorithm that is executed in the target device. For notational convenience, we use $f_k^{-1}(Y)$ to denote the set of all pre-images of plaintexts that lead to the internal state Y. The second state $Z = \phi(Y)$ is mapped through an intermediate non-linear function ϕ from the predecessor state Y. It is clear that the state Z is a functional composition output, represented as $Z = \phi \circ f_k(X)$. Note that both of Y and Z should produce observable side channel leakage to be exploitable by the side channel adversary. We use L_Y and L_Z to denote the observed leakages for processing the two respective states Y and Z.

The goal of NLCA is to generate collisions between state Y and state Z and to exploit them by detecting the correlated leakage behavior. That is, for a given plaintext X, we want to find another X' such that the induced internal states Y, Y', Z, Z' satisfy the cross-state collision of either $Y' = Z$ or $Z' = Y$. Without loss of generality, we explore the first type $Y' = Z$, i.e.

$$f_k(X') = \phi \circ f_k(X) \tag{1}$$

Clearly, if X' is chosen as one of the pre-images of $\phi \circ f_k(X)$, then it is a solution to equation (1). In other words, $X' \in f_k^{-1}(\phi \circ f_k(X))$ implies that the internal state $Y' = f_k(X')$ is guaranteed to be colliding with the internal state

$Z = \phi \circ f_k(X)$. Hence the observed leakage behavior of L_Z and $L_{Y'}$ can be expected to be very similar.

3.2 Building a Non-linear Collision Attack

We now show how this idea can be used and converted to a side channel attack on AES. The described approach can be easily adjusted to target many other block ciphers. We choose the non-linear operation ϕ as the first round[1] SubBytes. More precisely, we only consider ϕ as a single Sbox $S(\cdot)$ in the following context. The states Y and Z are then the input and output of the same Sbox respectively. The function f_k is the initial key addition (xor) operation. Figure 1(b) visualizes the idea of NLCA in this setting. The cross state collision in equation (1) becomes $X' \oplus k = S(X \oplus k)$ and clearly it has a unique solution

$$X' = k \oplus S(X \oplus k) \tag{2}$$

In other words, if the AES encryption algorithm is executed with plaintexts X and X' computed from equation (2), the produced side channel leakages $L_{Y'}$ and L_Z (with $Y' = X' \oplus k$ and $Z = S(X \oplus k)$) will be closely correlated. The adversary, however, does not know the subkey k and therefore cannot directly plug it into the equation and find such X'. Nevertheless, all possible subkey hypotheses can be checked to find the correct subkey k. Algorithm 1 shows the detailed procedure for the attack on AES. Basically, the adversary makes a total of 256 subkey hypotheses $g \in \{0,1\}^8$. For each hypothesis g, she computes $X'_g = g \oplus S(X \oplus g)$ for all possible plaintext bytes X. The resulting list of plaintext pairs X and X'_g is assumed to generate cross-state collisions $Z = Y'_g$, under this hypothesis g. The respective average leakage signals $L_Z, L_{Y'_g}$ are stored in vectors α, β_g. The Pearson correlation coefficient $\rho(\alpha, \beta_g)$ between them is finally computed for testing the subkey hypothesis g. After testing all subkey hypotheses, the adversary picks the subkey hypothesis k^* that yields the highest correlation coefficient and determines it as the correct subkey k, i.e. $k^* = \text{argmax}_g \{\rho(\alpha, \beta_g)\}$.

Validity If the hypothesis is correct, i.e. $g = k$, the computed $X'_g = X'_k$ has the same format as in equation (2). It follows that

$$\begin{aligned} & X'_g = g \oplus S(X \oplus g) = k \oplus S(X \oplus k) \\ \iff & X'_g \oplus k = S(X \oplus k) \\ \iff & Y'_g = Z \end{aligned}$$

Hence the respective mean signals α, β_g of the observed leakage should be similar and have high correlation. However if the hypothesis is wrong, i.e. $g \neq k$, then the above equations do not hold anymore. Hence Y'_g does not collide with Z and their respective leakage should only give low correlation.

[1] It can easily be translated to last round SubBytes with known ciphertexts.

Algorithm 1 Non-Linear Collision Attack on AES

Input: Number of Traces q, plaintext-byte values $X = [X_1, ..., X_q]$ Leakages $L_Y = [L_{Y,1}, ..., L_{Y,q}]$ and $L_Z = [L_{Z,1}, ..., L_{Z,q}]$
Output: Subkey Decision k^*

1: **for** x = 0 to 255 **do**
2: $U_x = \{i \mid X_i = x, i \in [1 : q]\}$ ▷ the set of indices where plaintext is x
3: $\alpha[x] = \text{avg}\{L_{Z,i} \mid i \in U_x\}$ ▷ mean leakage for processing Z
4: $\gamma[x] = \text{avg}\{L_{Y,i} \mid i \in U_x\}$ ▷ mean leakage for processing Y
5: **end for**
6: **for** g = 0 to 255 **do**
7: **for** x = 0 to 255 **do**
8: $x'_g = g \oplus S(x \oplus g)$ ▷ x and x'_g cause hypothetical collision $z = y'_g$
9: $\beta_g[x] = \gamma[x'_g]$ ▷ get the leakage for processing Y'_g
10: **end for**
11: $R[g] = \rho(\alpha, \beta_g)$ ▷ Pearson correlation coefficient
12: **end for**
13: $k^* = \text{argmax}_g \{R[g]\}$
14: **return** k^*

Adaptable with Higher Order Statistical Moments Generic distinguisher has low assumption on the leakage distribution. In certain scenario, leakage cannot be captured with the first order statistical moment (empirical mean) but is able to be detected through higher order moments (e.g. empirical variance, skewness, etc) as pointed out by [10]. The proposed non-linear collision attack can easily be extended to capture such hidden leakages. The adjustment is on line 3 of Algorithm 1. The original vector α is used to precompute the mean signal (i.e. 1st order moment) of leakage L_Z. That is

$$\alpha[x] = \text{avg}\{L_{Z,i} \mid i \in U_x\} = \frac{1}{|U_x|} \sum_{i \in U_x} L_{Z,i}$$

with U_x defined in line 2 of the algorithm. The d-th order moment ${}_d\alpha$ of leakage L_Z can also be precomputed for any integer $d > 1$

$${}_d\alpha[x] = \frac{1}{|U_x|} \sum_{i \in U_x} (L_{Z,i} - \alpha[x])^d$$

Similarly ${}_d\gamma$ can be computed on line 4 to store the d-th order moment of leakage L_Y. Finally, one can finish the changes by replacing the first order moment terms α, β_g in line 9 and 11 with d-th order ${}_d\alpha, {}_d\beta_g$ respectively. The adjusted algorithm can then distinguish subkey hypothesis using higher order statistical moments. A detailed description of the methods as well as the benefits can be found in [10].

3.3 Comparison with other SCA

In the following we explore possible benefits and drawbacks of NLCA when compared to other attacks.

Comparing NLCA with DPA,CPA The big difference between NLCA and DPA,CPA lies in the fact that NLCA does not rely on a particular leakage model, e.g. Hamming weight model. DPA and CPA correlate leakage sample to the leakage model of hypothesis, while NLCA make correlation between leakage samples. In fact, NLCA only requires the minimal assumption that processing the same internal state results in similar leakage behavior. If the leakage behavior is precisely captured by the leakage model assumed in the DPA and CPA, NLCA might not show advantage. However, if the leakage model deviates from the physical observables, the two classical methods are more likely to fail while the NLCA is still robust. More details can be found in Section 4.2.

On the negative side, NLCA requires identifying the bivariate leakage samples for processing states Y and Z respectively, prior to the attack. With a known implementation this is not an issue. As $Z = S(Y)$ is processed after Y with a fixed offset of clock cycles, finding the two critical time samples is equivalent to locating the first sample for L_Y and adding the offset to get the second sample for L_Z. For unknown implementations the location and offsets have to be guessed. This can be easy, e.g. if it is highly likely that the non-linear function is implemented as a table-lookup, resulting in an offset of a few clock cycles. But this might not always be the case.

Comparing NLCA with Collision Attacks The earlier works of side channel collision attacks [16, 15, 1–3, 22] define collisions as the same value of one target state from different inputs. The NLCA extends the definition such that collision occurs on two different targets Y and Z of the same value. The second difference is that the previous works belong to the chosen plaintext attacks since only plaintexts in certain pattern can make sure to cause collisions. The NLCA is not a chosen plaintext attack. It works with traces associated with random plaintext inputs and hence belongs to the known plaintext attacks. It sorts traces into different bins U_x and uses all of them. The last but not the least difference is that previous works rely on successfully *detecting* the collisions from traces before making use of their algebraic property to shrink the space of key hypotheses. The NLCA works in a CPA manner that it tests different subkey hypotheses and ensures that only the correct hypothesis *generates* collisions – not just a few collisions, but all the resulting input pairs x, x' cause collisions. In other words, previous works exploit leakage similarity of collisions prior to the use of its algebraic property, while the order reverses for NLCA. The benefit is to avoid the false acceptance of collision detection and hence to reduce the risk of misuse of algebraic property in earlier proposals.

Comparing NLCA with the Linear Correlation Collision Attack. The NLCA and LCCA have one common feature that they do not require a leakage model. This is because both are computing the correlation amongst leakage samples rather than comparing leakage samples to model values. Their complexity is also at the same level. For LCCA, there are totally 15 independent subkey differences amongst the 16 bytes in AES. It means that there is a remaining 8

bit key entropy even after disclosing all subkey differences. Therefore, the total complexity for recovering a full AES key using LCCA is 15×2^8 recoveries of subkey relations plus 2^8 full key verification. While on the other side, the NLCA recovers all subkeys independently. Its total complexity is 16×2^8 in subkey recoveries. Yet there are critical differences between the two. Firstly, the LCCA is categorized by [18] as non-standard side channel attack because it hypothesizes on relation between two subkeys rather than a subkey itself. While NLCA follows a more straightforward divide-and-conquer approach. Secondly, the collision exploited in the LCCA reveals the *homogeneity* of leakage behavior under the *same* operations. More specifically, both states Z_a and Z_b are the output of Sbox as seen from Figure 1(a). Hence they are derived from the same routine in the embedded system. For example, both are loaded from program memory into the state registers. The collisions generated from the correct hypothesis results in homogeneous leakage that should have high magnitude of correlation, which is shown in [11]. The NLCA, however, explores the *similarity* of leakage behavior caused by *different* operations. As can be seen from Figure 1(b) that Y' is the output of key xor and Z is the output of Sbox. It means they are processed with different instructions. For instance, Y' is xored or moved to a register and Z is loaded from program memory onto a register. Such operational difference results in leakages of non-linear collisions behaving similarly but not homogeneously. Therefore, it is not surprising that the level of correlation obtained from NLCA is lower than from LCCA. However, especially in the case of software implementations, it can be assumed that locating the second colliding state is easier for NLCA, as both leakages are more likely to occur close to each other.

Some Limitations The non-reliance of leakage model does not come for free. One prerequisite of the non-linear collision attack is the existence of the bivariate leakages: it is satisfied in the situation of software implementation but not in the hardwares. This restricts the applicability of the NLCA. In addition, it is not clear whether the NLCA can be extended such that it can also overcome countermeasures such as masking schemes.

4 Experiments

Three different groups of experiments are described in the following. The first group is the NLCA attack performed on power measurements of an 8-bit microcontroller executing AES-128. It also compares the performance of NLCA and CPA on the real measurements. The second group discusses situations where NLCA has significant advantage over CPA. The experiments are performed on simulated leakage traces for well-chosen leakage models. The third group focuses on the impact of the similar but inhomogeneous leakage behavior caused by exploiting leakages at different stages of a round.

4.1 Experiments on Smart Card Power Measurements

We first run the proposed NLCA using real measurements of the power consumption of an 8-bit AVR microcontroller, i.e. the ATXMEGA 256A3B processor. The microcontroller runs the Rjindael Furious [12]– a popular and efficient software implementation of AES-128 for AVR. A Tektronix digital sampling oscilloscope is used to measure power leakage traces. The sampling rate is set to 200M Samples per second which provides 100 sampling points per clock cycle. The Rjindael Furious implements the SubBytes operation on each byte as an S-box look up table (LUT). It firstly takes 1 clock cycle to move the input Y of Sbox into a particular register for relative addressing the LUT, then uses 3 cycles to load the output Z of Sbox from program memory into another register. It is therefore expected that there is an offset of 3 clock cycles (approximately 300 time points) between processing input state Y and the output state Z of Sbox.

Using Algorithm 1, we test all 256 subkey hypotheses over all time samples. That is, testing at time sample t refers to assuming L_Y occurring at sample t and L_Z occurs at sample $t + 300$. As can be seen from Figure 2, the correct subkey hypothesis (red) stands out remarkably from wrong hypotheses (gray) at the time sample around 6500, which means Y is processed around that time instance and Z around 6800. More importantly, it is observed that only the correct hypothesis results in a distinguishable correlation coefficient. This verifies the validity of the non-linear collision attack. It also indicates that leakages of collisions at different states under different instructions also behave similarly.

Next, the number of traces needed for a successful NLCA is explored. The correlation experiment is repeated on the discovered critical time point, as visualized in Figure 2, using 75 to 1000 traces. The observed trend is depicted in Figure 3. The correct hypothesis (red/dark) always features a higher correlation than the wrong ones (gray). The correlation computed from the correct subkey increases with the number of used traces, and seems not to have reached the limit with 1000 used traces. The counterparts from the wrong hypotheses, however, are bounded from -0.2 to 0.2. It is clear that the distinguishability in NLCA becomes increasingly remarkable with more traces. Note that the performance of NLCA using fewer traces is not covered in the plot. One might be interested in the performance of NLCA when, for example, only 20 or traces are available. However, NLCA requires finding a sequence of pairs (X, X') such that the resulted Y' and Z collide. With limited availability of leakage traces, it is very likely that intermediate states cannot be paired with the colliding counterpart. In other words, too few pairs or even no pairs of $L_{Y'}$ and L_Z can be used for computing correlation, which is easily biased or even undefined.

Next, the performance of NLCA and correlation based DPA (CPA) are compared on the same measurement setup. The attacks use the same set of 500 leakage measurements. The NLCA is tested on the critical time point discovered in Figure 2. The CPA assumes the Hamming weight leakage model of the output Sbox and it is therefore only performed on the most relevant time point for looking up the output state Z of the Sbox. As can be seen from Figure 4(a) and 4(b), both NLCA and CPA work well in this setting, outputting the correct subkey

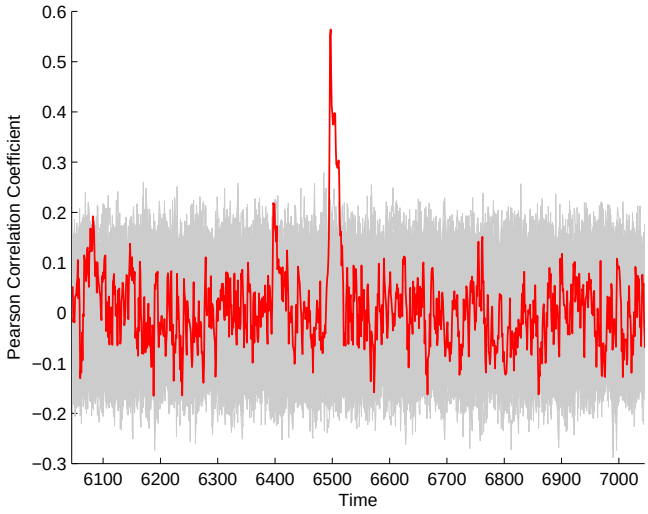


Fig. 2. Pearson correlation coefficient (y-axis) computed from non-linear collision attack over all time samples (x-axis). 1000 traces have been used in the experiments. Gray curves indicate correlation for the wrong subkey hypotheses; red (dark) represents the correct hypothesis.

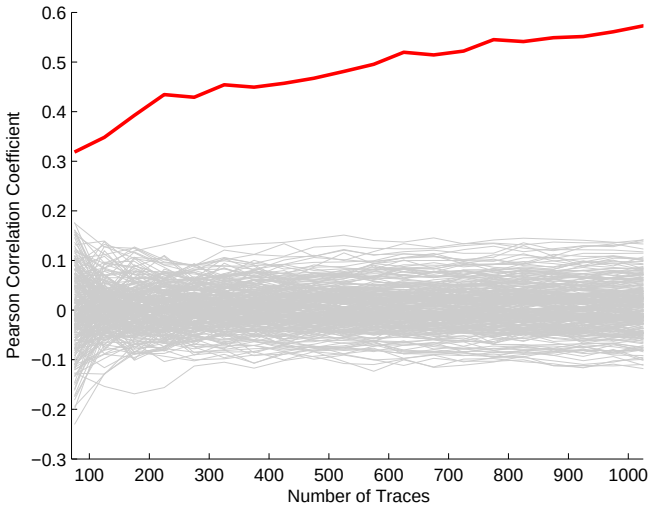
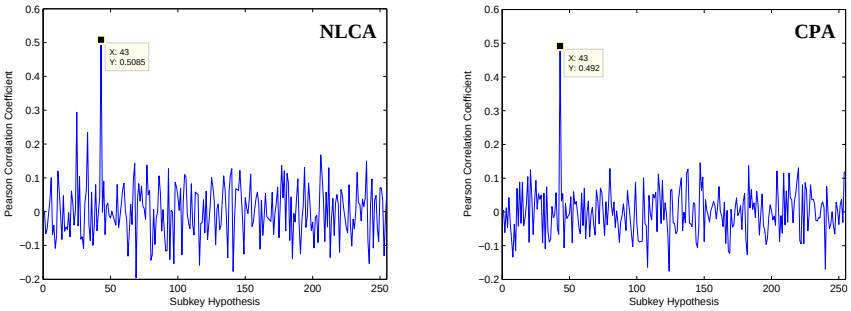


Fig. 3. Pearson correlation coefficient (y-axis) computed for the non-linear collision attack over the number of utilized leakage measurements (x-axis). NLCA is performed only over the most remarkable time sample disclosed in Figure 2. Gray curves represent wrong subkey hypotheses and the red (dark) curve represents the correct hypothesis.

43 with the highest correlation coefficient. It is hard to determine which attack performs better simply from the two plots. The NLCA gives the correlation for the correct subkey a little higher than the CPA. But the level of correlation for wrong hypotheses in NLCA (roughly between -0.2 to 0.2) is also higher than the CPA (roughly between -0.15 to 0.15). Nevertheless, the CPA assumes the Hamming weight leakage model. The experiment only indicates that the behavior of leakage obtained from the target microcontroller is well captured by the leakage model in CPA. In general, if the leakage does not behave according to the assumed leakage model, CPA might fail due to the modeling error. This effect is studied in greater detail in the following simulations.



(a) (b)

Fig. 4. Pearson correlation coefficient (y-axis) for all subkey hypotheses (x-axis) computed from NLCA (a) and from classical CPA (b). The latter assumes Hamming weight model of the Sbox output. The two attacks use the same set of 500 traces, applied to the most related time samples disclosed in Figure 2. The correct subkey 43 gives highest correlation in both scenarios.

4.2 Experiments on Simulations: Immunity to Modeling Errors

In this section, we run experiments to test the robustness of the proposed NLCA under different simulations of the leakage function. We show situations where the NLCA has significant advantage over the CPA and Mutual Information Analysis (MIA).

Adversarial Model. We consider four non-profiling adversaries: the classical CPA, the univariate MIA (UMIA), the multivariate MIA (MMIA), and our NLCA. The univariate target of CPA and UMIA is the output of Sbox. While for the MMIA and NLCA the targets are both the input and the output of Sbox. The CPA and the two mutual information based distinguishers all assume

Hamming weight leakage model [2]. All probability densities for UMIA and MMIA are estimated through the histogram method using 9 bins. The NLCA does not assume any power model.

Leakage Simulation Design. We follow the design proposed in [19] of three situations of simulation—the optimistic, the realistic and the challenging scenario. The optimistic scenario assumes the leakage behaves proportionally to the Hamming weight of the state value. I.e.

$$\lambda^{op}(Z) = \mathrm{HW}(Z) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the additive white Gaussian noise that has variance σ^2. The realistic scenario assumes an unevenly weighted Hamming weight model. That is, the least significant bit (LSB) of the intermediate data has a relative weight of 10 while all the other bits have weight of 1. So the leakage function is expressed as

$$\lambda^{re}(Z) = \mathrm{HW}(Z >> 1) + 10\mathrm{LSB}(Z) + \epsilon$$

The third case, i.e. the challenging scenario, assumes a non-linear leakage function, and it is instantiated as Sbox mapping composition with the Hamming weight function. That is when the state Z is processed, the leakage function evaluated at Z is

$$\lambda^{ch}(Z) = \mathrm{HW}(S(Z)) + \epsilon$$

In other words, processing state Z gives a leakage of the Hamming weight of the Sbox output of Z. It is clear to see that the modeling bias for CPA, UMIA and MMIA become increasingly severe in the three simulation scenarios.

Performance Comparison. We use the first order success rate and the guessing entropy [17] to evaluate the subkey recovery performance of the four distinguishers as shown in Figure 5. All metrics are derived empirically from 1000 independent experiments. In each experiment, the two correlation based distinguishers i.e. CPA and NLCA are fed with 256 simulated traces while the two mutual information based adversaries use 2560 traces because of the demand of pdf estimation.

It can be seen that only NLCA and MMIA survived from all three simulation scenarios: both their first order success rate and guessing entropy converge to 1. The CPA and UMIA are efficient when the Hamming weight model captures the simulated leakage functions very well. However, they become increasingly impacted by the leakage modeling errors. They succeed in the realistic scenario at a much higher SNR and remain as failure in the challenging scenario no matter how SNR varies. Interestingly, in the challenging situation, the guessing entropy of CPA and UMIA grow much higher than 128 – the quantity for a random guess without using side channel leakages– even if provided with strong signal.

[2] As pointed out in [19], the near generic 7LSB power model for AES does not perform well for the MIA and it even fails catastrophically in strong signal setting

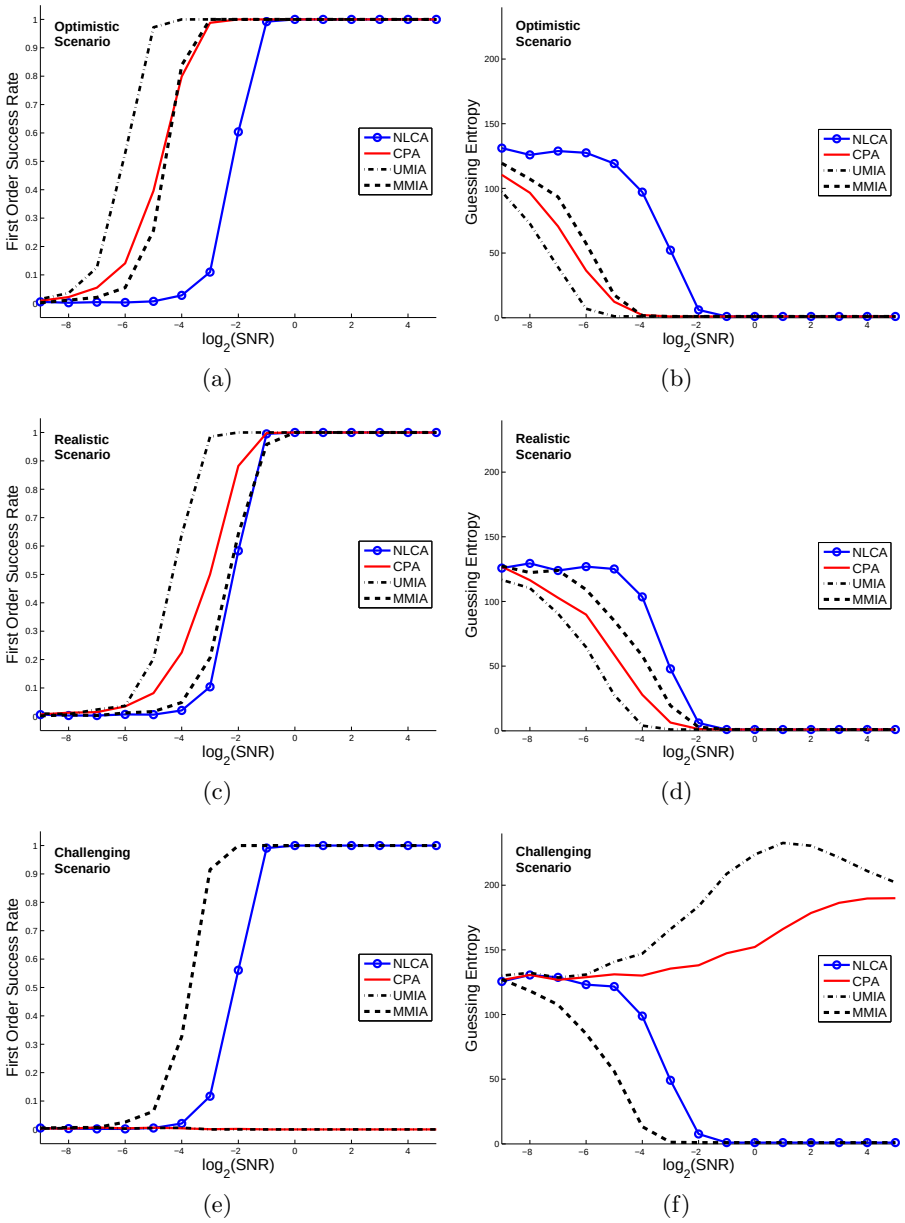


Fig. 5. Performance comparison of NLCA, CPA, UMIA and MMIA using First Order Success Rate (left) and Guessing Entropy (right) under three leakage function assumptions: Optimistic (Upper), Realistic (Middle), and Challenging (Lower). Experiments are simulated at different Signal-To-Noise ratios (x-axis). Leakage modeling error has negligible impact on NLCA, slight but noticeable impact on MMIA and severe impact on CPA and UMIA.

It indicates that the impact of false leakage model can be as catastrophic as misleading the adversary.

A first glance at the behavior of the two remaining distinguishers MMIA and NLCA appears to tell that former has some advantage over the latter. But one should consider that firstly the MMIA requires 10 folds of simulated traces than the NLCA because of the need of pdf estimation. Secondly, the behavior of MMIA at optimistic and challenging situations are much more similar, while at the realistic scenario it actually becomes worse. Such observation shows that the leakage modeling error still have some impact on its performance, just not in the same way as one could expected. On the contrary, the NLCA remains an unchanged pattern in all the three cases. Therefore, NLCA is robust with respect to different leakage functions and is immune to leakage modeling errors.

4.3 Impact of the Inhomogeneity of Leakages

As mentioned in Section 3.3, processing Y with a move instruction and Z with a load instruction results similar but not homogeneous leakage behavior even if the values of the two states collide. Abstractly, it can be viewed as the leakage functions over the state Y domain and state Z domain are different. The impact of the inhomogeneity of the bivariate leakage needs to be investigated. The last group of experiments shows the robustness of non-linear collision attacks against different levels of inhomogeneity in the leakage. We first define the homogeneity coefficient τ as the number of bits that both states Y and Z are leaking in the same manner. It induces the following leakage functions.

$$\lambda_\tau(Y) = \lambda(Y_L \| Y_R) = \lambda(U) + \lambda(Y_R) \tag{3}$$

The Y_R represents, for example, the rightmost τ bits of state Y, which are assumed to be leaking normally (i.e. with the same constant weight). The Y_L is respectively the remaining bits of Y that are assumed to be leaking in a different way. More precisely, in Equation (3), the Y_L is independent[3] of the leakage function, and it is replaced by an independently generated random $8-\tau$ bit value U, which then generates leakage. A corresponding leakage function is defined for state Z such that $\lambda_\tau(Z) = \lambda(V) + \lambda(Z_R)$ with a different random V. It is easy to see that when Y' collides with Z in the NLCA, the part Y'_R is the same as Z_R leading to $\lambda(Y'_R) = \lambda(Z_R)$ while $\lambda(U) \neq \lambda(V)$. In other words, the collisions are detected only from the common τ bits that are leaking in the same way. The remaining bits contribute only as noise. The lower the homogeneous coefficient τ, the more the leakages between the two leaking states will deviate from one another.

In our experiments the leakage function λ is instantiated with the Hamming weight function. In the τ homogeneous setting, this means that the leakage function $\lambda_\tau(Y) = \mathrm{HW}(U) + \mathrm{HW}(Y_R)$ generates Hamming weight of τ bits Y_R as signal, and the remaining random $8-\tau$ bits U give binomially distributed

[3] It can also be considered that Y_L is mapped non-linearly to U before generating leakages. This is similar to the challenging scenario discussed in Section 4.2.

noise. The equivalent is true for Z. A total of 400 independent experiments is performed. Each experiment uses 256 simulated traces generated from the above defined leakage functions. The result in Table 1 shows that for homogeneity coefficient $\tau \geq 3$, the NLCA gives 100% success rate even for a single subkey trial. When $\tau = 1, 2$ which are the fairly low level of homogeneity, the adversary can still achieve success rates more than 40% and more than 95% respectively by making 4 trials. The last line of the table uses the security description Guessing Entropy defined in [17] that quantifies the expected number of subkey guesses until finding the correct subkey. It is not surprising to see that 2 trials can guarantee the adversary finding the correct subkey when $\tau \geq 2$. Even at the lowest homogeneity level, it can still be achieved with 20 trials. To sum up, the NLCA shows very strong robustness against inhomogeneity of leakages for the two states. This result is not restricted to NLCA and apply in the same way to inhomogeneity of leakages in LCCA.

Table 1. The robustness of NLCA against various levels of homogeneity of leakage behavior for the two sensitive states.

	Homo. Coef. $\tau = 0$	Homo. Coef. $\tau = 1$	Homo. Coef. $\tau = 2$	Homo. Coef. $\tau = 3$ to 8
1st order Success Rate	0.3%	23.0%	89.3%	100.0%
4th order Success Rate	1.5%	43.0%	97.8%	100.0%
Guessing Entropy	126.13	19.18	1.35	1.00

5 Conclusion

This work proposes the non-linear collision attack as another variety of collision-based side channel attacks. The attack exploits leakages of collisions of different states and does not rely on accurate leakage modeling. Experimental results show that the leakage behavior for different states are similar enough to be exploited by NLCA especially in the software implementation situations. It also shows that inhomogeneous leakages generated by different operations have only low impact on the performance of the proposed attack.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. #1261399 and Grant No. #1314770. We would like to thank the anonymous reviewers for their helpful comments.

References

1. A. Bogdanov. Improved Side-Channel Collision Attacks on AES. In *Proceedings of the 14th international conference on Selected areas in cryptography*, SAC'07, pages 84–95, Berlin, Heidelberg, 2007. Springer-Verlag.

2. A. Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In *Cryptographic Hardware and Embedded Systems CHES 2008*. Springer Berlin / Heidelberg, 2008.
3. A. Bogdanov and I. Kizhvatov. Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. *Computers, IEEE Transactions on*, PP(99):1, 2011.
4. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 135–152. Springer Berlin / Heidelberg, 2004.
5. S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*. Springer Berlin / Heidelberg, 2003.
6. F. Durvaux, F.-X. Standaert, and N. Veyrat-Charvillon. How to certify the leakage of a chip? Cryptology ePrint Archive, Report 2013/706, 2013. `http://eprint.iacr.org/`.
7. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. *Cryptographic Hardware and Embedded Systems–CHES 2008*, pages 426–442, 2008.
8. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397, London, UK, 1999. Springer-Verlag.
9. L. Mather, E. Oswald, J. Bandenburg, and M. Wojcik. Does My Device Leak Information? An a priori Statistical Power Analysis of Leakage Detection Tests. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 486–505. Springer Berlin Heidelberg, 2013.
10. A. Moradi. Statistical tools flavor side-channel collision attacks. volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer Berlin / Heidelberg, 2012.
11. A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-enhanced power analysis collision attack. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-15031-9_9.
12. B. Poettering. Rijndael furious. Implementation. `http://point-at-infinity.org/avraes/`.
13. M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre. A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices. In K. Paterson, editor, *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer Berlin Heidelberg, 2011.
14. W. Schindler, K. Lemke, and C. Paar. A stochastic model for differential side channel cryptanalysis. In J. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer Berlin Heidelberg, 2005.
15. K. Schramm, G. Leander, P. Felke, and C. Paar. A Collision-Attack on AES. In *Cryptographic Hardware and Embedded Systems - CHES 2004*. Springer Berlin / Heidelberg, 2004.
16. K. Schramm, T. Wollinger, and C. Paar. A new class of collision attacks and its application to des. In T. Johansson, editor, *Fast Software Encryption*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer Berlin Heidelberg, 2003.

17. F.-X. Standaert, T. G. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. *Advances in Cryptology — EUROCRYPT 2009*, pages 443–461, 2009.
18. N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert. Security evaluations beyond computing power. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology — EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 126–141. Springer Berlin Heidelberg, 2013.
19. C. Whitnall and E. Oswald. A fair evaluation framework for comparing side-channel distinguishers. *Journal of Cryptographic Engineering*, 1(2):145–160, 2011.
20. C. Whitnall, E. Oswald, and L. Mather. An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis. In E. Prouff, editor, *Smart Card Research and Advanced Applications*, volume 7079 of *Lecture Notes in Computer Science*, pages 234–251. Springer Berlin Heidelberg, 2011.
21. C. Whitnall, E. Oswald, and F.-X. Standaert. The Myth of Generic DPA and the Magic of Learning. In J. Benaloh, editor, *Topics in Cryptology CT-RSA 2014*, volume 8366 of *Lecture Notes in Computer Science*, pages 183–205. Springer International Publishing, 2014.
22. X. Ye and T. Eisenbarth. Wide collisions in practice. In F. Bao, P. Samarati, and J. Zhou, editors, *Applied Cryptography and Network Security*, volume 7341 of *Lecture Notes in Computer Science*, pages 329–343. Springer Berlin Heidelberg, 2012.