2015-01-27

Side Channel Leakage Analysis - Detection, Exploitation and Quantification

Xin Ye
Worcester Polytechnic Institute

Side Channel Leakage Analysis
– Detection, Exploitation and Quantification

A Dissertation
Submitted to the Faculty
of the

WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy
in
Electrical and Computer Engineering
by

Xin Ye
Dec 2014

APPROVED:

Dr. Thomas Eisenbarth Major Advisor	Dr. Berk Sunar
Dr. Lifeng Lai	Dr. Adam Ding

Dr. Yehia Massoud
Department Head

Abstract

Nearly twenty years ago the discovery of side channel attacks has warned the world that security is more than just a mathematical problem. Serious considerations need to be placed on the implementation and its physical media. Nowadays the ever-growing ubiquitous computing calls for in-pace development of security solutions. Although the physical security has attracted increasing public attention, side channel security remains as a problem that is far from being completely solved.

An important problem is how much expertise is required by a side channel adversary. The essential interest is to explore whether detailed knowledge about implementation and leakage model are indispensable for a successful side channel attack. If such knowledge is not a prerequisite, attacks can be mounted by even inexperienced adversaries. Hence the threat from physical observables may be underestimated. Another urgent problem is how to secure a cryptographic system in the exposure of unavoidable leakage. Although many countermeasures have been developed, their effectiveness pends empirical verification and the side channel security needs to be evaluated systematically.

The research in this dissertation focuses on two topics, leakage-model independent side channel analysis and security evaluation, which are described from three perspectives: leakage detection, exploitation and quantification. To free side channel analysis from the complicated procedure of leakage modeling, an observation-to-observation comparison approach is proposed. Several attacks presented in this

work follow this approach. They exhibit efficient leakage detection and exploitation under various leakage models and implementations. More importantly, this achievement no longer relies on or even requires precise leakage modeling.

For the security evaluation, a weak maximum likelihood approach is proposed. It provides a quantification of the loss of full key security due to the presence of side channel leakage. A constructive algorithm is developed following this approach. The algorithm can be used by security lab to measure the leakage resilience. It can also be used by a side channel adversary to determine whether limited side channel information suffices the full key recovery at affordable expense.

To my parents and my wife.

Acknowledgment

This dissertation is completed partially at the Department of Mathematics at Florida Atlantic University and partially at the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute. The research is supported by National Science Foundation with grant 1261399.

First of all, I would like to express my deepest gratitude to my advisor Prof. Thomas Eisenbarth. The interesting cryptography class he taught in 2010 has introduced me into this research area. He has been organizing lots of seminars on cryptography and security related topics and creating an open-minded academic atmosphere. I am sincerely thankful for his continuous guidance and support in the past five years.

I would also like to thank my committee Profs. Berk Sunar, Lifeng Lai and Adam Ding. They have been making careful review, suggestions and critiques on my research and dissertation. Special thanks are also given to Dr. William Martin from WPI, Drs Lianfeng Qian, Lee Klingler and Rainer Steinwandt from FAU. Thank you for providing academic intelligent support and sharing.

My gratitude shall also be given to my peers Yarkin, Gorka, Aria, Michael, Cong, Wei, Yin and Chenguang. Thanks for being my trustworthy friends in the last two years.

Last but not the least, I want to truthfully thank my parents and my dear wife Yu Zhang for encouraging me and being supportive all the time.

Contents

List of Figures

List of Tables

List of Acronyms

AES Advanced Encryption Standard.

CPA Correlation Power Analysis.

DPA Differential Power Analysis.

FEMS Full Entropy Masking Schemes.

HOCPA Higher Order Correlation Power Analysis.

HOMIA Higher Order Mutual Information Analysis.

HOSCA Higher Order Side Channel Analysis/Attack.

KS Kolmogrov-Smirnov.

KSF Key Space Finding Algorithm.

LB Leakage Bundling.

LCCA Linear Collision Correlation Attack.

LDA Linear Discriminant Analysis.

LDDA Leakage Distribution Decomposition Attack.

LEMS Low Entropy Masking Schemes.

LGCA Leaking Group Collision Attack.

LSCA Leaking Set Collision Attack.

MIA Mutual Information Analysis.

ML Maximum Likelihood.

NLCA Non-Linear Collision Attack.

O2M Observation-to-Model.

O2O Observation-to-Observation.

PCA Principal Component Analysis.

RSM Rotating Sboxes Masking.

SCA Side Channel Analysis/Attack.

SPA Simple Power Analysis.

VGS Key Ranking Algorithm.

Chapter 1

Introduction

1.1 Motivation

Recent years have seen the ubiquitous computing touching almost every corner of the world. For instance, RFIDs have been widely employed in public transportation and logistic systems; Wireless sensor networks allow convenient health care and environment monitoring; Smart technologies even ”bring” the world into our cellphones, automobiles and homes.

Together with the convenience and efficiency are the privacy and security issues brought by those fast developing technologies. While different security requirement and goals are defined from national or international standardization as well as company wide security policies, technical solutions are mostly sought from cryptography. The original objective of cryptography is to provide a secure communication channel between different parties. Modern cryptographic primitives also offer services such as data confidentiality, integrity, authenticity and non-repudiation. The basic idea is to make use of some mathematically hard problems such as factorization and dis-

crete logarithm so that the potential advantage a passive or active adversary might obtain from the communication channel is limited to an negligible level. Various cryptographic algorithms and protocols have been implemented in e.g. banking, telecommunication, health care system, copyright protection and etc. They have been and will be continuously protecting our privacy, securing our financial assets and creating numerous business opportunities around the world.

The wide application of cryptography is supported by physical infrastructures on different scales such as super computers, the cloud and distributed computing systems as well as the embedded systems. Implementation of cryptographic primitives on those platforms is supposed to achieve their defined security objective. However, the discovery of side channel analysis (SCA) in late 1990s alerts the world that the computation on physical platforms can also become a secret teller that impedes any achievement of security goals in the real world. Basically, the SCA considers the execution of algorithm leaks sensitive information regarding internal system states from physical media. The adversary is allowed to make use of such leakage to challenge the security of the system. It reminds us that the physical security is no longer a negligible issue.

The past twenty years have seen numerous investigation on the threats from physical observables. A variety of SCAs and countermeasures have been proposed every year. While the studies facilitate us a more thorough understanding of the nature of SCAs and some possible ways to mitigate or live with them, the threat has been far away from being completely removed. Many problems remain unsolved for both the academia and the industry. They include but are not restrict to:

- How should one define the security goal in the presence of a potential side

channel adversary? Securing against a particular SCA is easier. But enumerating all possible SCA and ensuring security against all of them is just unmanageable. If the latter is desired, the only way is to show there is either equivalent or fully ordered relationship between different SCAs such that securing one will automatically secure the remaining. However, this put forward the difficulty towards the relation between different SCAs even if some of them might not even being proposed yet. The unclear security objective results in a limited standardization on the common security goal of SCA resistance.

- Have the current SCAs reached their maximum potential? Efficient SCAs require precise estimation of leakage models. It indicates that advancement in leakage modeling techniques makes the SCA more powerful than they are now. An alternative question is how much does SCA really need to rely on leakage modeling. Is there any data analytic techniques that can capture any data dependency while still provide meaningful side channel hypothesis discrimination? If the answer is yes, then the technical difficulty for mounting a SCA can be lower since an adversary who lack the experience in leakage model estimation can still perform the attack. Attack efficiency is another restricting factor. Existence of more efficient attack will definitely encourage the harass from adversaries.

- How should one evaluate the effectiveness of some countermeasures? How should one quantify the impact of the remaining leakage? Taking information theoretical approach is one possible solution by analyzing the empirical remaining key entropy. It is actually the approach that adopted by most current related works. However, comparing to the entropies, security evaluator is

more interested in the change in leakage exploitability and adversaries' ability in key guessing. Unfortunately, a systematic study regarding those aspect is not available.

Those unanswered questions provide us sufficient reasons for further researching in this area.

1.2 Our Contribution

This dissertation conduct SCA studies from three major aspects: leakage detection, exploitation and quantification. We are particularly interested in possible side channel distinguishers that do not rely on particular leakage behavior. Equivalently speaking, we aim at returning to capturing the key related data dependency nature of SCA without specifying a fixed dependency relation. Exploration in this respect not just provide an alternative way of mounting SCA, but it tries to release the SCA potentials and may contribute to the black box side channel attack and testing. A second central interest is in evaluating the practical impact on full key security from limited side channel observations. It tries to answer the questions like how much winning probability an adversary can achieve with the limited side channel leakage; How different observations influence her guessing strategy.

The summary of contribution is the following.

- A leakage bundling technique is introduced for leakage detection and exploitation. It improves the computational efficiency of classical SCAs and testing. It calls for an reevaluation on the effectiveness of side channel countermeasures such as hiding and masking.

- A novel observation to observation comparison approach is proposed for leakage exploitation. Comparing to the traditional observation to model comparison, this approach removes the leakage model estimation or assumption without sacrificing the capabilities of capturing generic leakage.

- A novel weak maximum likelihood approach is proposed for leakage quantification. A constructive key space finding algorithm is introduced as its instantiation. It help the security evaluator in determine whether bounded leakage ensures full key security of the system. It also provides a probabilistic winning strategy for the adversary and inform the respective guesswork load.

Most of the above listed contributions can be found in our recent publications [79, 80, 78, 81].

1.3 Outline of the work

We start from fundamentals in abstract algebra, statistics and information theory from Chapter 2 to provide sufficient preliminaries. Next we provide an overview of side channel attacks and countermeasures in Chapter 3. It is followed by the three main Chapters 4, 5 and 6. They discuss leakage detection, exploitation and quantification respectively. We summaries major findings and possible future directions in Chapter 7.

Chapter 2

Preliminaries

This chapter provides some basics in algebra, statistics, and information theory as necessary preliminary knowledge for this dissertation.

2.1 Algebra Preliminaries

In this section we provide the necessary preliminaries in abstract algebra where modern cryptography is built upon. We start from the simplest structure to the ring and finally reaches the more complicated field. In the end of this section we use the popular block cipher *Advanced Encryption Standard* as an example to obtain a first taste of the usage of those algebraic structure.

2.1.1 Group

The algebraic structure is defined within a set of elements and the operation defined among them. The formal notation of a *binary operation* on a set G is a function $* : G \times G \rightarrow G$. More specifically, the binary operation $*$ maps a pair (x, y)

of elements in G to an element $x * y$ that is still in the set G. We say the set G is closed under the operation $*$ if such operation can be defined on any pair $(x, y) \in G \times G$. Now we can introduce the concept of *group*, which is the simplest algebraic structure.

Definition 2.1. A group, denoted as $(G, *)$, is a set G that is closed under the operation $*$ and satisfies the following three properties:

(i) The *associative law* holds. I.e. $x * (y * z) = (x * y) * z$ for any $x, y, z \in G$;

(ii) There is an element $e \in G$ called the *identity* with $e * x = x = x * e$ for all element $x \in G$;

(iii) For any $x \in G$, there is an *inverse* $x' \in G$ with $x * x' = e = x' * x$.

It is easy to see the uniqueness of the identity: if both $e, e' \in G$ are identity as in Def. 2.1, then $e = e * e' = e'$. It follows the uniqueness of the inverse element: if both x', x'' are inverse of x as in Def. 2.1, then

$$x' = x' * e = x' * (x * x'') = (x' * x) * x'' = e * x'' = x''$$

It is also easy to see that the inverse of the identity is the identity. We now give some examples of groups.

Example 2.1. Typical examples of groups

- The set of all integers with additive operation, i.e. $(\mathbb{Z}, +)$. In this group, the identity element is 0, and the inverse of x is $-x$.
- The set of all non-zero rational numbers with multiplicative operation, i.e. $(\mathbb{Q}\backslash\{0\}, \times)$. In this group, the identity element is 1, and inverse of x is x^{-1}.

- The set of integers modulo n with additive operation, i.e. $(\mathbb{Z}_n, +_n)$. The notation $\mathbb{Z}_n$ refers integers $\{0, 1, ..., n-1\}$. The operation is also called mod n addition , i.e. $x +_n y := x + y \bmod n$. For notational convenience, the operation $+_n$ is simplified as $+$.

- The set of non-zero integers modulo prime p with multiplicative operation, i.e. $(\mathbb{Z}_p \backslash \{0\}, \times_p)$. The notation $\mathbb{Z}_p \backslash \{0\}$ refers positive integers $\{1, 2, ..., p-1\}$, which can also be denoted as $\mathbb{Z}_p^{\times}$. The mod p multiplication $\times_p$ is defined as $x \times_p y := xy \bmod p$.

Conventionally, additive group – the group operation being addition or its variants – is denoted as $(G, +)$ with its identity being $\mathbf{0}$ and the additive inverse of x being $-x$. The multiplicative group is likewise denoted as $(G, *)$ with the identity being $\mathbf{1}$ and the multiplicative inverse of x is denoted as x^{-1}. We should point out that such operational distinction only impacts the arithmetic that the operation defines but does not make a difference from the abstract algebraic perspective since group consider only one type of operation. Without loss of generality and for the ease of notation, we use the notation of additive group $(G, +)$

For finite groups, we can also define group *order*. It is denoted as $|G|$ and refers the number of elements that is in this group. The term order can also be defined for each element $x \in G$ and it is denoted as $\circ(x)$. More specifically, $\circ(x)$ refers the smallest positive integer d such that the summation of d copies of x returns the identity, i.e., $d \cdot x := x + x + ... + x$ (d times) $= \mathbf{0}$. We omit the proof of the existence and suggest the interested reader for the work [62]. Nevertheless should we remember the important fact that the order of any group element is a divisor of the order of the group, i.e. $\circ(x) \mid |G|$, for all $x \in G$. In the extreme case where

$\circ(x) = |G|$, we say the group G is a cyclic group and is generated by the element x.

To get a first taste of the group structure, we introduce the subgroup.

Definition 2.2. A subset H of a group $(G, +)$ is a subgroup if

(i) It contains the identity, i.e. $\mathbf{0} \in H$;

(ii) It is closed under the group operation. I.e. if $x, y \in H$ then $x + y \in H$;

(iii) It is closed under inversion. I.e. For any $x \in H$, then $-x \in H$.

It is easy to see that a subgroup $(H, +)$ of the group $(G, +)$ is itself a group. Subgroups induce a special partition of the group into the *cosets*.

Definition 2.3. If H is a subgroup of the group $(G, +)$, then $a + H := \{a + h \mid \forall h \in H\}$ is called a coset of H.

We should see cosets are of the same cardinality and they are either disjoint or completely overlap.

Theorem 2.1. Let H be a subgroup of a group $(G, +)$, and let $a, b \in G$.

(i) $|a + H| = |H|$ for any $a \in G$.

(ii) $a + H = b + H$ if and only if $a - b \in H$.

(iii) $(a + H) \cap (b + H) \neq \emptyset$ implies $a + H = b + H$.

The above three property yields an important property of the finite group as summarized in the Lagrange's Theorem.

Theorem 2.2. (Lagrange's Theorem). If H is a subgroup of a finite group $(G, +)$, then $|H|$ is a divisor of $|G|$.

What it indicates is a finite partition of the group G into equal sized and pairwise disjoint cosets: $G = (a_1 + H) \cup (a_2 + H) \cup ... \cup (a_t + H)$. The maximum number of pairwise disjoint cosets is called the *index* of subgroup H and it satisfies $t = \frac{|G|}{|H|}$.

We now take a look at a particular example of group – the set of all n-bit strings $(\mathbb{F}_2^n, +)$ with bitwise additive operation. Although the notation $\mathbb{F}_2^n$ can have different interpretation, their algebraic essence is the same, which is a binary field extension and we will detail later. Nevertheless, one of its representation is the set of all n-bit strings $\{a_1 a_2 ... a_n \mid a_i \in \mathbb{F}_2, \forall i\}$. We should see if $\vec{a}, \vec{b} \in \mathbb{F}_2^n$, then $\vec{a} + \vec{b} = \vec{c}$ with $c_i = a_i + b_i \bmod 2$. The bitwise binary addition is also called bitwise xor and can be denoted shortly as $\vec{a} \oplus \vec{b} = \vec{c}$. The identity is the zero-string $\vec{0}$ and the inverse of any $\vec{a}$ is, not surprisingly, itself as each component is in $\mathbb{F}_2$.

The *binary linear code* is essentially a subgroup of the n-bit strings $\mathbb{F}_2^n$. The elements contained in this subgroup are called *codewords*. Moreover, the number of different bits between two codewords is called the *Hamming distance* or in short the distance between the two codewords. A binary linear code is denoted as $\mathcal{C} := [n, k, d]$ indicating that the length of each codeword is n, the number of codes is 2^k and the minimum distance of all pairs of codewords is d. Since the order of $\mathbb{F}_2^n$ is 2^n and the order of the subgroup $[n, k, d]$ is 2^k, the index of the subgroup or namely the number of disjoint cosets is 2^{n-k}. Further, it further implies that the lowest Hamming weight of non-zero codewords is exactly d, namely,

$$d = \min\{HW(c \in \mathcal{C} \backslash \{\vec{0}\})\}$$

The reason is that the all-zero string $\vec{0}$ is the identity of the group and must be included in the linear code as the identity in the subgroup. Those parameters

have important meaning in the context of coding theory. Interested readers are recommended to follow [37] for better understanding.

2.1.2 Polynomial Ring and Finite Field Extension

Rings are an algebraic structure that is more complicated than groups. This is reflected in the fact that a ring has two binary operations. To distinguish them, we name one addition and the other multiplication. Here we only consider the case where both of the operations are commutative. In formal, a binary operation $*$ is called commutative on a set S if for any $a, b \in S$, $a * b = b * a$. We now give the formal definition of a commutative ring.

Definition 2.4. A commutative ring $(R, +, \times)$ is a set of elements where addition and multiplication are defined and the following holds:

(i) R is an additive group;

(ii) Both addition and multiplication are commutative, i.e. $a + b = b + a$ and $ab = ba$ for all $a, b \in R$;

(iii) Multiplication is associative, i.e. $a(bc) = (ab)c$ for all $a, b, c \in R$;

(iv) Existence of multiplicative identity (also called the unit): $\exists \mathbf{1} \in R$ such that $\mathbf{1}a = a$ for all $a \in R$;

(v) R satisfies distributivity, i.e. $a(b + c) = ab + ac$ for all $a, b, c \in R$.

It is easy to prove the uniqueness of the unit. Moreover, if there exist $a, b \in R$ such that $ab = \mathbf{1}$, we say a, b are multiplicative inverse to each other. It is easy to see that the multiplicative inverse of the unit is itself. The definition of the ring

does not require the existence of multiplicative inverse for other elements in the ring. However, if all non-zero elements are multiplicatively invertible and multiplicative identity differs from additive identity, we say the ring actually qualifies a *field*. We now give some examples.

Example 2.2. Typical examples of rings and fields

- $(\mathbb{Z}, +, \times)$, $(\mathbb{Q}, +, \times)$, $(\mathbb{R}, +, \times)$ and $(\mathbb{C}, +, \times)$ are commutative rings with usual addition and multiplication.
- $(\mathbb{Z}_n, +_n, \times_n)$, i.e. the set of integers modulo n with mod n addition and mod n multiplication. Moreover, if n is a prime, then all non-zero elements are invertible and hence it becomes a field. It is usually denoted as $\mathbb{F}_n$.
- The set of polynomials $k[X] := \{\sum_{i=0}^{d} a_i X^i \mid d \geq 0, a_i \in k\}$ with polynomial addition and multiplication where k refers a field. This is clearly not a field as e.g. the monomial X is not invertible.

In addition to subrings, there is a special structure called *ideal*.

Definition 2.5. An ideal in a commutative ring R is a subset I of R that satisfies

(i) $\mathbf{0} \in I$;

(ii) I is closed under ring addition;

(iii) $a \in I$ and $r \in R$ imply $ra \in I$.

Conventionally we use $(a) := aR = \{ar \mid r \in R\}$ to denote the ideal generated by element a. The definition of ideal not necessarily require to include the unit. In

fact, if an invertible element u is inside the ideal I, then $I = R$ simply because $u \in I$ implies $1 = uu^{-1} \in I$ and obviously $(1) = I$.

We now consider an important field that is commonly used, which is called field extension. We pay a particular attention to the case of finite field whose cardinality can only be powers of a prime. The construction makes use of the polynomial modulo reduction with an irreducible polynomial. In formal, let $k[X]$ be a polynomial ring with k being a field, a polynomial $f(X) \in k[X]$ is called irreducible if there is no $g(X), h(X) \in k[X]$ such that $f(X) = g(X)h(X)$ and $\deg g(X), \deg h(X) < \deg f(X)$. For example, $X^2 + X + 1$ is an irreducible polynomial in $\mathbb{F}_2[X]$. Theories in abstract algebra shows that there exist irreducible polynomial of arbitrary degrees, which we omitted here. The classical field extension technique is summarized in the following theorem.

Theorem 2.3. Let k be a finite field, $p(X) \in k[X]$ and the notation[1] $k[X]/(p(X))$ represent the set of polynomials after modulo reduction by $p(X)$, i.e. $k[X]/(p(X)) := \{a(X) \bmod p(X)\}$. Then $k[X]/(p(X))$ is a field if and only if $p(X)$ is an irreducible polynomial over $k[X]$. Moreover, if this is the case, then $|k[X]/(p(X))| = |k|^d$ where $d = \deg p(X)$

The construction through polynomial modulo reduction actually results in a linear vector space over the original field k with a basis being $1, \beta, \beta^2, ..., \beta^{d-1}$ where β is a root of the irreducible polynomial in the extended field. More straightforwardly as one's intuition indicates, whenever $p(X)$ is irreducible over k, we get

$$k[X]/(p(X)) \cong \{\sum_{i=0}^{d-1} a_i X^i \mid a_i \in k\}$$

[1]This is in formal referred as the quotient ring, where one can consider each polynomial $f(X)$ is mapped to its equivalent class $f(X) + (p(X))$

2.1.3 Advanced Encryption Standard

The Advanced Encryption Standard (AES) is a US NIST standard block cipher and is listed in the Federal Information Processing Standard (FIPS) publication 197 [57]. It is the most popular block cipher in the contemporary world. For instance it is included in protocols such as IPsec, TLS, SSH, the Wi-Fi encryption standard IEEE 802.11i and so on. In this section we outline the design of AES and use its mathematical structure as an example of the cryptography use of the ring and field structure introduced in Section 2.1.2.

AES has block size of 128 bits and offers three different key length ranging from 128, 192 to 256 bits for encryption and decryption. The 128 bit block string, namely 16 bytes, is indexed from s_0 to s_{15}. Another common organization of the 16 bytes is to treat them as a 4 x 4 matrix and each entry is indexed $s_{i,j}$ where $0 \leq i, j \leq 3$. Moreover, the 8 bits in each byte value is indexed as $b_7 b_6 ... b_0$ with the leftmost being the most significant bit (MSB) and rightmost being the least significant bit (LSB).

The design of AES follows a Substitution Permutation Network (SPN). The AES encryption firstly xors the plaintext with the secret key then passes the state to 10, 12, or 14 (depending on the key length) round functions. The final output is returned as the ciphertexts. Each round function except in the last round consists of four subroutines called **AddRoundKey**, **SubBytes**, **ShiftRows** and **MixColumns**. The last round omits the **MixColumns** operation with the purpose of making the entire decryption routine equivalent to the encryption procedure. The key used in each round is derived as an expansion from the initial key. We now list the algebraic nature of each subroutine.

- The **AddRoundKey** operation $s' = s \oplus k$ is the bitwise addition in $\mathbb{F}_2$.

- The **SubBytes** operation is a byte-wise one-to-one mapping. It treats the 8 bit strings in a byte as a finite field extension $\mathbb{F}_2[Y]/(m(Y))$ where the irreducible

$$m(Y) = Y^8 + Y^4 + Y^3 + Y + 1$$

  The transformation consists two steps. Firstly, it maps any non-zero element to its multiplicative inverse in this field $\mathbb{F}_2[Y]/(m(Y))$, and maps the element zero to itself. The second step is an affine transformation over $\mathbb{F}_2$ which is defined as

$$b'_i = b_i + b_{[i+4]_8} + b_{[i+5]_8} + b_{[i+6]_8} + b_{[i+7]_8} + c_i$$

  where $[\cdot]_8$ is a short notation for the mod 8 operation, the index i refers the i-th bit of a byte and c is the constant bit string 01100011.

- The **ShiftRows** operation cyclically shifts the byte values in each row of the 4 x 4 matrix with different offsets. In particular, entries in row i is shifted cyclically to the left by i position such that $s_{i,j}$ is moved to the position $(i, [j-i]_4)$.

- The **MixColumns** is more complicated. From the high level, it is a word-wise transformation mapping 4 bytes in one column of the 4x4 matrix to another 4 bytes values. Its algebraic essence is the polynomial multiplication in the quotient ring

$$\mathbb{F}_2^8[X]/(X^4+1) := \{a_3X^3 + a_2X^2 + a_1X + a_0 \mid a_i \in \mathbb{F}_2^8\}$$

where if $a(X), b(X) \in \mathbb{F}_2^8[X]/(X^4+1)$, then

$$a(X) \otimes b(X) = \sum_{m=0}^{6} \sum_{i+j=m} (a_i \bullet b_j) X^m \bmod X^4 + 1 \tag{2.1}$$

where all additions among coefficients are in $\mathbb{F}_2^8$ and are hence bitwise xor, and the multiplications $a_i \bullet b_j$ in $\mathbb{F}_2^8$ are modulo reduction by $m(Y)$.

In fact, AES chooses X^4+1 for the quotient ring modulo to make the reduction as simple as $X^i \bmod X^4 + 1 = X^{[i]_4}$. It simplifies (2.1) as

$$a(X) \otimes b(X) = d(X) = d_3 X^3 + d_2 X^2 + d_1 X + d_0$$

where

$$\begin{aligned}
d_0 &= (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3) \\
d_1 &= (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3) \\
d_2 &= (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3) \\
d_3 &= (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)
\end{aligned}$$

which is equivalent to

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.2}$$

The input of the **MixColumns** plays the role as the polynomial $b(X)$ and it is multiplied by a specific choice of $a(X) = 03X^3 + 01X^2 + 01X^1 + 02$ where all coefficients are the hexadecimal representation of the 8-bit string in $\mathbb{F}_2^8$. In other words, the **MixColumns** is exactly $b'(X) = a(X) \otimes b(X)$ with the $a(X)$ specified above and can be computed as the equation (2.2). Remember that since the modulus $X^4 + 1$ is not irreducible, the polynomial quotient ring is not a field and hence not all polynomials have multiplicative inverse. However, the above specified $a(X)$ is chosen in the way such that it does have an inverse which is $a^{-1}(X) = 0bX^3 + 0dX^2 + 09X^1 + 0e$. Consequently, the inversion of **MixColumns** is just $b(X) = a^{-1}(X) \otimes b'(X)$.

2.2 Statistics Preliminaries

Side channel attacks can be interpreted as applying statistical methods to discriminate key hypotheses. This section provides the fundamental concepts in random variables, probabilities and their distribution as well as relationship between variables and distributions.

2.2.1 Probabilities of Events

Probabilities are firstly used to describe the frequency of the occurrence of certain event in repeated experiments. For example, if within N repeated independent experiments, event A occurs $0 \leq N_A \leq N$ times, then the event A is said to have empirical probability $Pr(A) = N_A/N$. Clearly the quantity of probability is a real number between 0 and 1. If multiple events are considered within a single experi-

ment, one might be interested in how the occurrence of one event impact another. This motivates the definition of conditional probability and posterior probability. In formal, the probability of occurrence of A given the condition of the occurrence of B is defined as

$$Pr(A \mid B) = Pr(AB)/Pr(B)$$

where $Pr(AB)$ stands for the probability that both event A and B occur. In particular, the two events are independent to each other if and only if $Pr(A \mid B) = Pr(A)$, indicating the condition B does not affects the probability of A. An equivalent expression for independence relation is $Pr(AB) = Pr(A)Pr(B)$ The posterior probability $Pr(B \mid A)$ describes the probability of event B *after* knowing that A occurs. Expressively, it is defined as

$$Pr(B \mid A) = \frac{Pr(A|B)Pr(B)}{Pr(A)}$$

2.2.2 Random Variables and Probability Distributions

The set of all possible results from a single experiment is referred as the sample space. Random variables are defined on the sample space. Assigning a random variable with a particular experimental result creates an event and one can therefore define probability for this event. The assignment of probabilities for all possible results provides a probability distribution of the random variable. We use X to denote the random variable and Ω be the sample space. If Ω is continuous, e.g. $\mathbb{R}$ then the defined random variable X is continuous and a probability density function (*pdf*) is used to describe the distribution. The integral of the *pdf* over Ω is 1. If Ω is discrete, e.g. $\mathbb{Z}$, then X is said to be discrete and a probability mass function (*pmf*)

describes its distribution. The summation of the *pmf* over Ω is also 1.

Using parametric methods, distributions can be depicted with parameters such as mean, median, mass, standard deviation and etc, which can be estimated empirically. For example, repeating the experiments for n times and one obtains results denoted as $x_1, ..., x_n$, then the mean value is obtained from (2.3) and the standard deviation is evaluated from (2.4).

$$\tilde{\mu}_X = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.3}$$

$$\tilde{\sigma}_X = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \tilde{\mu}_X)^2} \tag{2.4}$$

They serve as an unbiased estimation of the population mean μ and the population standard deviation σ. Another popular parameter estimation is the maximum likelihood estimator. It has the same expression of sample mean as in equation (2.3). The formulation for sample variance is slightly changed to

$$\tilde{\sigma}_X = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \tilde{\mu}_X)^2}$$

Details of estimation of parameters can be found in the textbook [16].

Now we present two commonly seen probability distributions. The first one is the *uniform distribution* with its *pmf* and *pdf* expressed as

$$p(X = x) = \frac{1}{\|\Omega\|}$$

Intuitively from its name, the probability mass/density at any instance of the sample space is a constant value. No sample instance is more probable or less probable than

any other instance. This is actually the ideal situation where modern cryptography intends to achieve. For example, a random number generator is supposed to generate pseudo random numbers that appear just like real random numbers to win the real-or-random security games. A perfect side channel secrecy requires that probability distribution of all key candidates remain uniform both before and after the leakage observation.

The probably most widely used probability distributions is the *Gaussian distribution* (also called the normal distribution). A random variable X that follows a Gaussian distribution is denoted as $X \sim \mathcal{N}(\mu, \sigma^2)$. It is a probability distribution for continuous variable that is uniquely determined by the population mean μ and the population standard deviation σ. The *pdf* is expressed as,

$$p(X = x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.5}$$

Statistical theory shows that if the observed samples are from Gaussian distribution, then $\mathbb{E}(\tilde{\mu}) = \mu$ and $\mathrm{Var}(\tilde{\mu}) = \sigma^2/n$. In fact, the sample mean $\bar{x}$ from n samples follows the Gaussian distribution $\mathcal{N}(\mu, \sigma^2/n)$.

In many communication theory literatures it is shown that the noise in communication channel exhibits Gaussian distribution. Many times it is referred as additive white Gaussian noise (AWGN). Lots of earlier literatures on SCA assume the distribution of noises or leakage in side channel is also AWGN. Later in this dissertation we will use the notation $\mathcal{N}(L; \mu, \sigma^2)$ to indicate the leakage L follows a normal distribution with mean μ and standard deviation σ.

2.2.3 Relation between Random Variables or Distributions

The most commonly used statistic for describing the relation between two random variables is the Pearson product-moment correlation coefficient or in short correlation coefficient. The correlation coefficient between random variables X and Y is formally defined as the ratio between the covariance to the product of their standard deviation. That is,

$$\rho(X,Y) = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}\{(X-\mu_X)(Y-\mu_Y)\}}{\sigma_X \sigma_Y}$$

If the proability distributions for the two random variables are unknown, one can estimate it from their instantiations $x_1, ..., x_n$ and $y_1, ..., y_n$ by applying

$$\tilde{\rho}(X,Y) = \frac{\sum (x_i - \tilde{\mu}_X)(y_i - \tilde{\mu}_Y)}{\sqrt{\sum (x_i - \tilde{\sigma}_X)^2} \cdot \sqrt{\sum (y_i - \tilde{\sigma}_Y)^2}} \tag{2.6}$$

The correlation coefficient is a real number between -1 and 1. The closer the absolute value is to 1, the more linear relationship is observed from the two random variables. However, if the correlation coefficient is close to 0, it does not indicate the two variables are not related. It only tells that the linear correlation is not significant.

Another common way to tell the relation between two random variables is by computing the distance between their probability distributions. The Kolmogorov-Smirnov (KS) distance is one of the candidate. The KS test first evaluates the empirical cumulative distribution functions (*cdf*) F_X (and same for F_Y resp.) for the variable X (and Y resp.) from N samples $x_1, ..., x_N$. The evaluation of *cdf* at

a point a_t is from

$$F_X(a_t) = \frac{1}{N}\sum_{i=1}^{N} \chi_{x_i \leq a_t} \tag{2.7}$$

The two *cdf*s are further compared in

$$D_{KS}(X\|Y) = \max_t |F_X(a_t) - F_Y(a_t)| \tag{2.8}$$

Higher KS distance indicates a greater difference between the two distributions. We will see many applications of the correlation coefficients and KS distance throughout this dissertation.

2.3 Information Theoretic Basics

Information theory has been widely applied in many fields such as data processing and communications. With respect to the side channel security, information theory plays a big role in leakage resilience (c.f. Section 3.2) and leakage quantification (c.f. Chapter 6). This section introduces the most basic concepts such as entropy, mutual information and guesswork and we only consider discrete random variables here. For more detailed context and proofs it is recommended to read early chapters in [15].

2.3.1 Entropy and Conditional Entropy

The term *entropy* is a quantitative description of the uncertainty of a random variable with respect to its underlying distribution. We follow its classical definition.

Definition 2.6. A discrete random variable X with its probability mass function

(pmf) $Pr(X)$ has entropy

$$H(X) = -\sum_{x \in \mathcal{X}} Pr(x) \log Pr(x) \tag{2.9}$$

By convention, the log in Def. 2.6 is of base 2 and the resulting quantity is measured as the number of bits. The remaining context also apply this convention. The expression can also include the case where *pmf* of X vanishes. In particular, if $Pr(x) = 0$ then $Pr(x) \log Pr(x) := 0$. From this definition, it is easy to see that the entropy of a random variable has the range $0 \leq H(X) \leq \log \|\mathcal{X}\|$. The left equality holds if and only if there exists $x \in \mathcal{X}$ such that $Pr(X = x) = 1$ and $Pr(X \neq x) = 0$, namely, the *pmf* is trivial and there is no uncertainty about this variable. The second equality holds if and only if $Pr(X = x) = Pr(X = x')$ for all $x, x' \in \mathcal{X}$, i.e., the random variable X has uniform distribution over its space and therefore has maximum uncertainty. The entropy is also a synonym of *self-information* which will be explained later.

Considering the uncertainty of two or more variables, one can define their *joint entropy*.

Definition 2.7. A sequence of discrete random variables $X_1, X_2, ..., X_n$ has joint distribution $Pr(X_1, X_2, ..., X_n)$, their joint entropy is defined as

$$H(X_1, X_2, ..., X_n) = -\sum_{(x_1,...,x_n) \in \mathcal{X}_1 \times ... \times \mathcal{X}_n} Pr(x_1, ..., x_n) \log Pr(x_1, ..., x_n) \tag{2.10}$$

Before discussing the properties of the joint entropy, we need a quantitative description of the remaining uncertainty of a random variable X_2 given the knowledge of another random variable X_1. This is called the *conditional entropy* and is denoted

as $H(X_2 \mid X_1)$.

Definition 2.8. Two discrete random variables X_1 and X_2 have joint distribution $Pr(X_1, X_2)$, the conditional entropy of X_2 given knowledge of X_1 is defined as

$$H(Y \mid X) = \sum_{x_1 \in \mathcal{X}_1} Pr(X_1 = x_1) H(X_2 \mid X_1 = x_1) \tag{2.11}$$

Without providing proofs, we summarize two important properties of conditional entropy and joint entropy as follows.

1. The conditional entropy is no more than the entropy of the same variable, namely, $H(X_2 \mid X_1) \leq H(X_2)$ for any X_1, X_2. The equality holds if and only if the two variables are independent.

2. The joint entropy can be expressed using the *Chain Rule* as the summation of a sequence of entropy and conditional entropies. In formal,

$$H(X_1, X_2, ..., X_n) = H(X_1) + H(X_2 \mid X_1) + ... + H(X_n \mid X_1, ..., X_{n-1}) \tag{2.12}$$

2.3.2 Mutual Information

Mutual information is used to quantify the information that one random variable contains about another random variable. In other words, it describes the reduced uncertainty of one random variable due to the knowledge of another.

Definition 2.9. Two random variables X_1 and X_2 have joint distribution $Pr(X_1, X_2)$,

the mutual information is defined as

$$I(X_1; X_2) = \sum_{(x_1,x_2)\in\mathcal{X}_1\times\mathcal{X}_2} Pr(x_1, x_2) \log \frac{Pr(x_1, x_2)}{Pr(x_1)Pr(x_2)} \tag{2.13}$$

From this definition we should first see the symmetric property of mutual information, i.e.

$$I(X_1; X_2) = I(X_2; X_1) \tag{2.14}$$

With some mathematical derivation, we should also see the link in the tuple of the entropy, conditional entropy and the mutual information.

$$\begin{aligned} I(X_1; X_2) &= H(X_1) - H(X_1 \mid X_2) && (2.15)\\ I(X_1; X_2) &= H(X_1) + H(X_2) - H(X_1, X_2) && (2.16)\\ I(X_1; X_2) &= H(X_2) - H(X_2 \mid X_1) && (2.17) \end{aligned}$$

More specifically, the mutual information between two random variable is the entropy loss of either one given the knowledge of the other. If we simply consider the mutual information of X to itself, equation (2.13) gives $I(X; X) = H(X)$. Therefore, the entropy of a random variable is sometimes referred to as the self information.

Last but not the least should we mention the quantitative impact on the entropies and mutual information given that the two random variables are independent. Since independence implies $Pr(X, Y) = Pr(X)Pr(Y)$, it is easy to see that the conditional entropy $H(X \mid Y) = H(X)$ and $I(X; Y) = 0$. In other words, there is no entropy reduction of one random variable due to the knowledge of any other random variable that is independent from it. Equivalently, there is 0 mutual information between

two independent random variables.

Chapter 3

Overview of Side Channel Analysis

This chapter presents an overview of the side channel security. We start from the change of security model due to the discovery of side channel attack (SCA) in late 1990s. We revisited the evolution of SCA variants in the last two decades and its common countermeasures. We end this chapter by summarizing several major direction of the contemporary research in SCA.

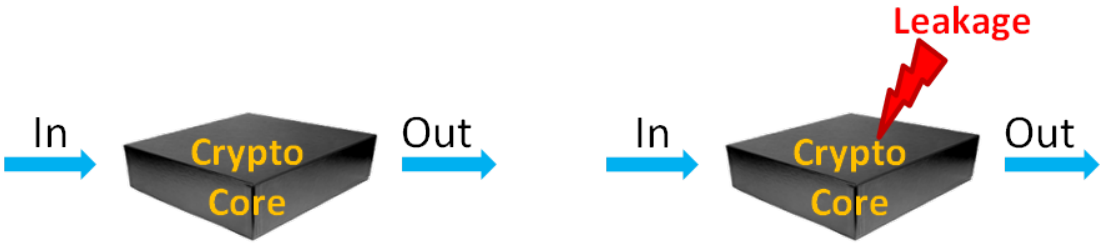
3.1 Side Channel Attack

We have seen numerous security notions such as known / chosen plaintexts / ciphertexts security, which are commonly used in cryptography textbooks e.g. [30, 49]. These security models view the crypto algorithms as black boxes where the knowledge of the adversary is limited upto the input and/or the output of these black boxes. For example, the security of an encryption scheme requires plaintext security such that pure knowledge of the ciphertexts does not enable decrypting and recovering the plaintext without the knowledge of the key. Another example is,

cryptographically secured hash function requires pre-image resistance 2nd pre-image resistance. It implies that the a polynomial adversary who is given the hash checksum cannot find the exact input of the hash function or even another input that results in the same hash output. This black box model was not challenged until the discovery of side channel attack (SCAs) in [32]. The SCA shows the fact that the execution of crypto primitives on physical devices unavoidably leaks sensitive information from side channels such as EM, power consumption, timing channels. Such information leakage can be detected and exploited by the adversary vanish the security goal established in the main (input-output) channel. As a consequence, the security community calls for a remodeling of security that considers also the threat from side channel information leakage as shown in Figure 3.1(a) and 3.1(b). In this revised model, the knowledge of an adversary is not limited to the algorithmic inputs and outputs, she can also observe signals from the side channels. It requires the security goals that were previously defined in black box scenario also being achieved even if the adversary obtains such additional information. By the channel where the adversary gets further benefits from, the SCA can be categorized as Timing Attacks (e.g. [32]), Power Analysis Attack (e.g. [31]), EM Attacks [1] or even multi-channel attacks [2]. In this work, we consider mostly the power analysis attacks. We first present the adversarial model as follows.

3.1.1 Adversarial Model

The SCA is viewed as the most powerful attack that can be mounted by an adversary. This is because the adversary is assumed to have physical access to the physical device where the crypto algorithm is implemented and executed. She is sometimes

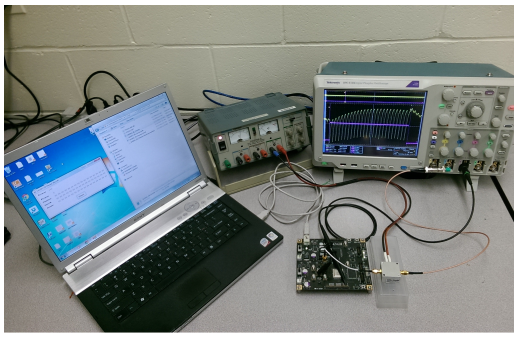


(a) Security Model without the SCA (b) Security Model with the SCA

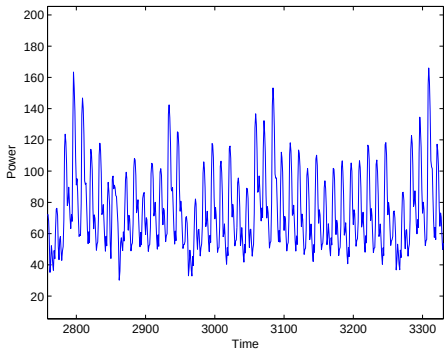
Figure 3.1: In a black box model (a), only the algorithmic input and output are exposed to adversaries. In the model with SCA consideration (b), the side channel information, i.e. the observed leakage provides adversaries extra knowledge about the internal system states. The protection from the underlying crypto core risks being compromised.

even assumed to be able to choose the algorithmic inputs and outputs. In the presence of such powerful adversarial knowledge model, the goal of SCA is unique: secret key recovery. For power based SCA, the setup in a research lab is shown in Figure 3.2(a). The target implementation of crypto algorithm in software/hardware is connected to a SCA workstation that controls the algorithmic inputs and outputs. The crypto device is also connected to a sampling equipment such as a digital oscilloscope, which monitors the side channel, i.e.here the power consumption changes, and returns the saved side channel measurement (also referred to as power traces) to the workstation for further analysis as shown in Figure 3.2(b). More detailed experiment setup can be found in [23].

For example, if the crypto primitive is an encryption scheme $\mathrm{Enc}_{sk}(\cdot)$, the main channel returns $W = \mathrm{Enc}_{sk}(X)$ based on the plaintext input X and the side channel returns $L = \phi(f_{sk}(X))$. Here f, called a target function, represents a part of the crypto algorithm $\mathrm{Enc}_{sk}(\cdot)$ that generate sensitive internal system state $Y = f_{sk}(X)$. As SCA usually proceed in a divide-and-conquer manner, the target function is



(a) Power based SCA Setup



(b) Observation from the Power Channel

Figure 3.2: To setup a typical power based SCA (a), one connects the crypto device, an oscilloscope and a laptop together. An adversary/ lab tester queries the crypto core from the laptop. The oscilloscope records the signal of power consumption on the crypto device and returns the captured power traces (b) back to the laptop for further analysis.

usually related with only partial key k from the whole key sk and can therefore also be denoted as $f_k(X)$. The other function $\phi(\cdot)$ represents the actual power model that characterizes the power consumption caused by the sensitive internal state Y. Unless specified in simulations, the true power model is assumed unknown to adversaries or evaluators in security labs. Nevertheless, one can estimate a degraded power model $\tilde{\phi}(\cdot)$ as an approximation to the actual underlying power function $\phi(\cdot)$. The estimation may come either from the knowledge of the implementation or a profiling stage (c.f. 3.1.5).

The side channel observation can be viewed as a time series, namely, $\mathbf{L} = [\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_N]$ with N time samples. If the adversary is allowed to makes a maximum of q queries (also referred as q-limited adversary) to the target device with inputs $\mathbf{X}^q = [X_1, ..., X_q]$, and she observes the respective power leakage time series

$\mathbf{L}^q = [\mathbf{L}_1^q, \mathbf{L}_2^q, ..., \mathbf{L}_N^q]$ which is further expressed as

$$\mathbf{L}^q = \begin{bmatrix} L_{1,1} & L_{1,2} & ... & L_{1,N} \\ L_{2,1} & L_{2,2} & ... & L_{2,N} \\ ... & ... & ... & ... \\ L_{q,1} & L_{q,2} & ... & L_{q,N} \end{bmatrix} \tag{3.1}$$

with the first sub-script indicating the query-index and the second for the time index.

To sum up, the knowledge of a q-limited adversary include the algorithmic inputs $\mathbf{X}^q$, the power observations $\mathbf{L}^q$ as well as an estimated or pre-assumed power model $\tilde{\phi}(\cdot)$.

3.1.2 Simple Power Analysis

Simple Power Analysis (SPA) is the earliest SCA proposed by Kocher *et al.* in [32]. It often just makes use of only *one* power trace. In other words, the extra knowledge obtained by the adversary is $\mathbf{L}^1 = [L_{1,1}, L_{1,2}, ..., L_{1,N}]$. By comparing the relative height of the consumed power or the relative time length of the operation, it attempts to assign a one-to-one correspondence between the leakage measurement and the internal sensitive value and thereby recovers the key. For example, if RSA uses the square-and-multiply algorithm for modulo exponentiation, the adversary might try to identify if both the squaring and the multiplication occur (corresponding to an 1 on the exponent) or if only a squaring occurs (corresponding to an 0 on the exponent). If all bits of the exponent are recovered in an decryption oracle, the decrypting key is revealed. In general, SPA often requires detailed knowledge about

the implementation of the crypto primitive and it often requires very clear signal or high signal-to-noise ratio (SNR) in practice. Although SPA as an attack can be prevented by decreasing SNR, it can still be very helpful in terms of providing an education on distinguishing the part of crypto algorithm from the measurements.

3.1.3 Differential Power Analysis

The original Differential Power Analysis (DPA) has significantly impacted later researches in physical security. It was proposed by Kocher *et al.* in [31]. Unlike SPA, it does not require the detailed knowledge of the device under attack. As a trade-off it reveals the secret key by using a relatively large volume of measurements. From the data analytic point of view, the DPA is performed across different measurements at fixed time sample each iteration. This is fundamentally different from the SPA, which is only performed over the time-axis. The DPA generally follows a four-step procedure.

1. Choose a target internal state $Y = f_k(X)$ and a power model $\tilde{\phi}(Y)$.
2. Take side channel measurement and obtain knowledge $\mathbf{X}^q, \mathbf{L}^q$.
3. Make subkey hypothesis g and evaluate the hypothetical power values $\mathbf{H}^q = \tilde{\phi}(f_g(X^q))$ in the power model.
4. Compare the observed power traces $\mathbf{L}^q$ with the hypothetical values $\mathbf{H}^q$ and try to distinguisher the correct subkey from all hypotheses.

In particular, the original DPA uses one bit power model such as taking $\tilde{\phi}(Y)$ as the least/most significant bit of the value Y. From this power model it obtains

two populations of the hypothetical power values, namely 0 and 1. The DPA distinguishes the correct key by looking at the difference between the mean values of side channel observations in the population 'Zero' and that in the population 'One'. The argument is that on one side the correct key guess predicts the internal leaky state Y correctly for every input X and hence the adversary obtains the single bit correctly every time. While on the other side, a wrong key hypothesis predicts the state Y wrongly which further results in a wrong partition of the Zero population and the One population.

3.1.4 Correlation Power Analysis

Correlation Power Analysis (CPA) is originally presented in the work [10] as a variant of DPA. The used power model is usually taking as Hamming weight HW or Hamming distance HD depending on details of the implementation. As always, the Hamming weight of a value Y is the summation of its binary representation. And the Hamming distance of two values Y_1, Y_2 is the Hamming weight of their xor $HD(Y_1, Y_2) = HW(Y_1 \oplus Y_2)$. This assumption views that each bit of the sensitive value contributes independently and equally to the total power consumption. As a consequence, each subkey hypothesis yields $n+1$ partitions of the entire population, namely the populations that correspond to power model value being 0, 1,..., n where n is the bit size of internal state, e.g. 8 for a byte. In the distinguishing step, the Pearson correlation coefficient (c.f. Section 2.2) is used to compare the observation with hypothetical power values. The validity argument of CPA is similar to that for the DPA in that only the correct hypothesis results in $n+1$ correct partitions that further yields a significant correlation value.

3.1.5 Template Attack

Template attack (TA) proposed in [11] belongs to the profiling SCA that differs dramatically from the previous SCAs. It does not assume a power model $\tilde{\phi}(\cdot)$ prior to obtaining side channel measurement. Instead, it allows a profiling stage where the adversary is assumed to know the key and is given full control of the device. During this stage she makes as many measurements as she can which enables an estimation of the underlying power model $\tilde{\phi}(\cdot)$. Following the profiling stage is the attacking stage, where all privileges are deprived and the adversary is only given knowledge $\mathbf{X}^q, \mathbf{L}^q$.

In the original proposal, the estimated power model (also called the template) considers the leakage samples follow a Gaussian distribution where the two parameters are fully determined by the sensitive internal state Y. Depending on the dimensionality of the leakage (e.g. number of time points it contains) being considered, the Gaussian distribution can be either univariate or multivariate and is in general denoted as $\mathbf{L} \sim \mathcal{N}(\vec{\mu}_y, \mathbf{\Sigma}_y)$. Or equivalently,

$$P(\mathbf{L}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{L} - \vec{\mu}_y)^T \Sigma_y^{-1} (\mathbf{L} - \vec{\mu}_y)\right)}{\sqrt{(2\pi)^d \det(\Sigma_y)}}$$

where $\vec{\mu}_y$ refers to the mean leakage vector given that the internal state being y and the Σ_y is the covariance matrix for profiled leakage vectors in this population. The exponent d indicates the time dimensionality of the leakage. In a simplified case, each template only contains the mean leakage vector and is therefore referred as the reduced template. In other words, each time sample of the leakage vector is considered as independent from other time points and the variances do not vary for

different values y or different time point. The resulting formulation becomes

$$P(\mathbf{L}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{L}-\vec{\mu}_y)^T(\mathbf{L}-\vec{\mu}_y)\right)}{\sqrt{(2\pi)^d}}$$

In the attacking stage, each newly observed leakage sample is substituted into templates $\mathcal{N}(\vec{\mu}_y, \mathbf{\Sigma}_y)$ for all possible y. The evaluation of probability prioritizes different subkey guesses. The template with highest probability density evaluation is matched to this leakage sample and the respective subkey guess is determined as the key. Furthermore, the reduced template matching is in fact equivalent to finding the shortest Euclidean distance $\parallel \mathbf{L} - \vec{\mu}_y \parallel$ between the observed leakage vector $\mathbf{L}$ and the template mean vector $\vec{\mu}_y$.

3.1.6 Mutual Information Analysis

The Mutual Information Analysis / Attack (MIA) proposed in [26] is motivated from the pursuit of a generic SCA that does not rely on profiling leakage models such as in Template approach 3.1.5 and stochastic modeling approach [63]. It follows the information theoretic intuition. It computes the mutual information (c.f. 2.3) $I(L; \tilde{\phi}(Y))$ between the physically observed L and the leakage model $\tilde{\phi}(Y)$. Here the leakage model is not necessarily the one that approximate the unknown leakage function. The functionality is more to provide a partition on the queries.

The initial purpose of MIA is to build a generic distinguisher and hence it involves the procedures of the key hypothesis discrimination. The procedure is, for each subkey hypothesis g, the mutual information $I(L; \tilde{\phi}(f_g(X)))$ is computed. The subkey guess that gives the highest empirical mutual information is selected as the

correct key. It is important that the used leakage model cannot be the identity model given an injective target function $f()$. That is to say, the most generic situation – the term $\tilde{\phi}(Y)$ is the same as Y – cannot be applied. The reason is a one to one correspondence will just permute the summand within the computation of the mutual information. For example, AES Sbox is a very popular target function. It is however a one-to-one mapping from the knowntext byte to the output Sbox byte. If an identity model is used for computing MIA, constant mutual information will be returned regardless of the hypothesis g and diminish the distinguishability of subkey hypothesis testing. Practically, a less-than fully generic model, namely the 7 LSB model is normally used.

MIA has another popular use case: it serves as a metric for quantifying the leakage. The purpose has changed from revealing the key to evaluating implementation. In other words, the central interest has changed from comparing MI from different subkey hypothesis at the leaking time point to comparing MI computed from the same true key but over different time points. A roughly constant behavior of the computed mutual information over all time points indicates the remaining leakage is hard to detect or exploit. Since subkey hypothesis distinguishing is not required for the quantification, it is safe to choose the leakage model as the identity model.

Many other works studied MIA from various perspectives. The work [74] discusses the usability, pros and cons for using MIA compared with traditional SCAs. Authors in [55] provided a theoretical analysis of this information theoretical tool. It is also integrated in leakage detection testing methods as shown in [13, 12].

3.2 Side Channel Countermeasures

Different countermeasures have been proposed and developed to counteract the threat from side channel attacks during the last 15 years. This section lists three main categories. One starts from an implementation level and aims at *hiding* the leakage by making the side channel observable independent from the processed data. Another features secret sharing nature and *masks* the sensitive values with random values locally in the device. The other one discusses how to achieve the side channel leakage *resiliency* from a protocol level.

3.2.1 Hiding

Hiding is a categorical name of side channel countermeasures that pursues an independent leakage observation from processing different internal states. It is mainly achieved by either increasing the noise or decreasing the signal, i.e. reducing the signal-to-noise ratio (SNR). The purpose is to reduce the information retrievability from the adversary.

With respect to the power based SCA, common hiding practices should hide leakages both in the time dimension and in the amplitude dimension. Popular approaches for hiding leakage in the time dimension include randomly inserting dummy instructions or operations, randomizing order of execution, random delaying, and etc. They increases the difficulty for power trace alignment as the hiding technique makes a particular internal state being processed at different time samples in different queries. On the other side, hiding leakage from the amplitude dimension can be achieved through making constant power consumption such as dual-rail precharge logic.

One should see that the hiding countermeasure does not completely eliminate the threat from side channel. Instead, it increases the difficulty of mounting such attacks.

3.2.2 Masking

Masking schemes (c.f. [14, 29, 24]) were introduced as a generic countermeasure to prevent first order SCA. A d-th order masking scheme protects every internal algorithmic state Y with d randomly generated mask values $M_1, M_2, ..., M_d$ from the mask group $\mathcal{M}$. More specifically, the sensitive algorithmic internal values were processed indirectly in a masked representation

$$Y_M = Y * M_1 * M_2 * ... * M_d \tag{3.2}$$

in the embedded computing environment where $*$ refers as the group operation such as $\oplus$ in the Boolean masking scheme [14], $\times$ in multiplicative masking scheme [29]. Since the device can only manipulate the masked representations, the execution does not directly leak information about the sensitive algorithmic internal value Y.

From a secret sharing perspective, the secrets hidden in sensitive algorithmic internal values Y are split into $d+1$ shares, namely, $M_1, M_2, ..., M_d$ and Y_M. This corresponds to, with our notations introduced earlier in this chapter, $d+1$ time samples $L_{t_0}, ..., L_{t_d}$ where L_{t_0} refers the time sample when the masked output Y_M is processed and the remaining L_{t_i}s refer the time point when the mask M_is are processed. In particular, a software implementation of d-th order masking scheme creates $d+1$ distinct time points to eliminate univariate SCA. While for a hardware implementation, although the time shares are "compressed" into one clock cycle,

the behavior of the leakage is changed dramatically to prevent being explored by first-order SCAs. Nevertheless, a common nature they share is that a perfect secret sharing scheme ensures the secret being independent from any d shares from the $d+1$ tuple. That is to say, even if an adversary can obtain non-decaying information of any d shares from the side channel measurement, she still gains zero information about the algorithmic sensitive states Y and hence the secret key k.

The attacks that overcome the masking countermeasure are called higher order SCA indicating they either explore leakage in a multivariate (higher time order) manner or in the nature of higher order statistical moments. Such attacks are much more complicated than the classical attacks described in this chapter. We will have a detailed discussion in Chapter 4.

Implementing masking scheme in practice is fairly costly because it produces significant overhead of computation. A practical simplification is the low entropy masking schemes. It uses only a few carefully chosen mask values with the purpose of preventing attacks such as DPA and CPA. However, the reduced cardinality of masks does not satisfy the requirement of a perfect secret sharing any more. This leaves the possibility of even univariate attacks. We will address this issue later in Section 5.3.

3.2.3 Leakage Resilience

The discovery of SCA has called for redefining the security model. The milestone work is the physically observable cryptography from Micali *et al.* in [42]. It redefines the security goal and the targeting strongest adversary. It proposes five axioms in the settings where the ones that have deepest influences are "only computation leaks"

and "leakage is local". The work inspires the establishment of the research area of *leakage resilient cryptography*. Proposals includes leakage resilient signatures, leakage resilient pseudo random number generation, leakage resilient symmetric encryption, leakage resilient mode of operation and etc. Important theoretical works include [21, 67, 22, 3, 50]. They admit the information leakage and they attempt to incorporate physical threat into the traditional black-box security model and seek formal security proof. Triggered by the need of the proofs, they make assumptions such as bounded leakage, bounded retrieval, non-adaptive chosen leakage function etc.

Leakage resilient designs of a key-ed cryptography primitive share a common feature of key updating. The idea is simple: since leakage occurs locally and is bounded, one should frequently updating the key to prevent the information leakage cumulates and the key entropy drops too much. As a result, leakage resilient designs should be viewed as a high level or protocol level countermeasure of SCA, which is not an alternative solution for the countermeasures such as hiding and masking. In fact the countermeasures in implementation level and secret sharing perspective affects the ability of retrieval of local leakage.

Although most works focus on the information theoretical proof, the validity of the leakage resiliency stands on a precise estimation of retrievable leakage bound. Therefore there are two central questions for practitioners: given a q-limited adversary, how to estimate the leaked information and how does such leakage impact quantitatively adversary's ability in guessing the key. We will address these questions in Chapter 6.

Chapter 4

Leakage Detection

Side channel leakage detection refers to the process of locating possible time points from side channel observation which offers adversary some advantages in distinguishing secret key. It is a topic that is interested to both adversaries and security labs. For the former, the process is itself part of the key recovery attack. For security labs, the detection is to provide a yes or no answer to determine whether the target implementation or device is leaking information and at what confidence level. In this case, the tester can even complete the detection task with the knowledge of the key.

In this chapter we first present two challenges for side channel leakage detection. One regards generic leakage models and the other is about the computational complexity especially for the multivariate leakage scenario. Their possible solutions are discussed later in this chapter.

4.1 Current Challenges

For side channel adversaries, detecting leakage is a prerequisite for mounting a successful SCA. Classical attacks introduced in Section 3.1 integrate the leakage detection as part of the leakage exploitation or the attack. Some other proposed attacks [7, 60] run the leakage detection prior to distinguishing the secret keys.

From the adversarial model in Section 3.1.1, one can see that the observed leakage $\mathbf{L}_j$ depends on the leakage function $\phi(\cdot)$. A first challenge is, if without any knowledge of the leakage model, can an adversary still detect the leakage successfully? We should see that, if in different queries the same internal states are processed by the same instruction, homogeneous leakage observations should be expected. That is to say, leakages that correspond to processing the same values can be viewed as being sampled from the same distribution. It is independent from the knowledge about the function $\phi(\cdot)$ or its approximation. In fact, it is essential for making a successful SCA without relying on the leakage model. More details are presented in Section 4.2.

A second challenge regards the computational complexity of detecting leakage. Classical SCAs and testing methods complete the detection in a point-wise manner. For example, given some side channel time series $\mathbf{L}^q$ in equation (3.1) that contains N time samples, the complexity for detecting leakages or mounting SCA is at the level of $O(N)$. Remember, this is only for the detection of univariate leakage for unprotected implementation. In fact, multivariate leakage detection is a much more complicated task. For example, given a software implementation is protected by a 1st order masking scheme, the leakage is bivariate, i.e. it consists of two time samples. Locating the two samples correctly at the same time requires checking all the

$\binom{N}{2} = \frac{1}{2}(N^2 - N)$ combinations of time samples. In other words, the computational complexity is $O(N^2)$. In general, the complexity of detecting multivariate leakage is of combinatoric nature. That is, a d-th order masking scheme in a software implementation produces $d + 1$ variate leakage. Trying all the $d + 1$ combinations from a total of N time samples results in the detection complexity $\binom{N}{d+1} = \frac{N!}{(d+1)!(N-d-1)!}$ and hence $O(N^{d+1})$ since $N >> d$. We will later discuss a more efficient leakage detection procedure in Section 4.3 to resolve this problem.

4.2 Wide Collision Detection

Collision attack such as [64, 5] is a different SCA from the classical attacks introduced in Section 3.1. It does not include a subkey hypothesis testing phase. Instead, it tries to use the side channel information together with the ciphers' algebraic property to eliminate impossible key candidates until the correct one is found. The side channel information is used to detect the event of a collision from different queries. Successful collision detection provides a set of correct algebraic relations that further eliminates wrong key candidates.However, errors in collision detection are detrimental. False detection almost surely yields the correct key being eliminated.

The false positive issue motivates the proposal of wide collision attack that provides reliable collision detection. In this section we first review the two concepts of collision attack and wide collision attack. We focus on the topic of the wide collision detection and propose methods that offer high detection rate.

4.2.1 Collision Attack

We have seen in the adversarial model in Section 3.1.1, the target function can be chosen at adversary's will. In general, we say an **internal collision** in some cryptographic primitive occurs if the target function $f_k(\cdot)$ produces the same output value y for two different inputs x_1, x_2, i.e.,

$$f_k(x_1) = y = f_k(x_2) \tag{4.1}$$

For example, any non-injective mapping such as the DES Sboxes will cause internal collisions. Internal collisions in AES were defined by [64] and generalized by [5]. Collisions occur at the output of the **MixColumns** transformation in each round function of AES. This is because the **AddRoundKey**, **SubBytes** are one-to-one mappings hence cannot produce collisions at the same position. **ShiftRows** only affects the order of bytes but does not alter state values. As we have seen in Section 2.1.3 the **MixColumns** is a word-oriented operation, mapping from $\mathbb{F}_2^{32}$ to itself. The formulation of equation (2.2) further reveals each output byte of the column is a function of all the four input bytes. This results in a non-injective mapping and hence produces collisions. For example, given two plaintexts $X^a = \left(x_{ij}^a\right)$ and $X^b = \left(x_{ij}^b\right)$ with $i, j \in \{0, 1, 2, 3\}$ in query a and query b, an internal collision at byte 0 (or the position $(0, 0)$ in matrix representation) after round 1 **MixColumns** can represented by

$$02 \cdot y_{00}^a \oplus 03 \cdot y_{10}^a \oplus 01 \cdot y_{20}^a \oplus 01 \cdot y_{30}^a = 02 \cdot y_{00}^b \oplus 03 \cdot y_{10}^b \oplus 01 \cdot y_{20}^b \oplus 01 \cdot y_{30}^b \tag{4.2}$$

where $Y^a = (y^a_{ij})$ and $Y^b = (y^b_{ij})$ refer to the internal states before the **MixColumns** operation for the two queries [1]. Since each $y^a_{i0} = S\left(k_{ii} \oplus x^a_{ii}\right)$ and $y^b_{i0} = S\left(k_{ii} \oplus x^b_{ii}\right)$, the equation (4.2) can be viewed as a function

$$F(k_{00}, ..., k_{33}) = 0 \tag{4.3}$$

that is algebraically related to the four bytes of key at the diagonal positions (i, i). The plaintext knowledge $x^a_{00}, ..., x^a_{33}, x^b_{00}, ..., x^b_{33}$ are the parameters for the function (4.3). It is clear that four independent collisions determine a unique solution this function, namely the four-byte key. Hence 16 correct collision detections suffice the full key recovery. That is to say, applying collision attack is equivalent to the problem solving of a set of linear equations. However, this is in the ideal case assuming that the side channel leakage reveals non-degraded information about the internal state. In other words, it only happens in the case of error free collision detection –that the decision is always correct regarding the equality of the values on two sides of equation (4.2). In fact, the side channel leakage in the real world practices is usually more sophisticated than the ideal case. One can only learn degraded information about the internal state. As a consequence, it risks in making the set of linear equations being "corrupted" and hence eliminating the correct key from the key space.

4.2.2 Wide Collisions

Pursuing higher collision detection success rate is the proposal of wide collision attack. It is firstly defined in [7] as an chosen plaintexts SCA. More specifically,

[1] Each side of equation (4.2) is the finite field multiplication explained in Section 2.1.3.

plaintexts in all queries are chosen to satisfy the condition that the off-diagonal bytes are pairwisely equal i.e. $x_{ij}^a = x_{ij}^b$ for all $i \neq j$. The basic idea is that if an internal collision occurs for after first round **MixColumns**, more collision will be generated in the second and third rounds. Consequently, this gives rise to one byte collision in Round 2 **SubBytes** and four additional byte collisions in Round 3 **SubBytes**, resulting in a total of five byte collisions. This phenomenon is referred to as *wide collision*.

For example, if two queries a and b collide at byte 0 after Round 1 **MixColumns** $z_{00}^a = z_{00}^b$ (equivalently represented as in equation (4.2)), the two states will continue to be the same after Round 2 **SubBytes**:

$$S\left(k'_{00} \oplus z'^a_{00}\right) = S\left(k'_{00} \oplus z'^b_{00}\right)$$

due to one-to-one correspondence, where k' refers the first round key. After **ShiftRows**, all the four byte states in Column 0 are pairwisely equal and hence yield four additional internal collisions after Round 2 **MixColumns** $z''^a_{i0} = z''^b_{i0}$ for all $i \in \{0, 1, 2, 3\}$. The states will remain pairwisely identical after Round 3 SubBytes

$$S\left(k''_{i0} \oplus z''^a_{i0}\right) = S\left(k''_{i0} \oplus z''^b_{i0}\right).$$

To sum up, the chosen plaintexts enable expanding one internal collision (if any) after first round **MixColumns** to

- one byte collision after second round **SubBytes**;
- one column collision after second round **MixColumns**;

- four byte collisions after third round **SubBytes**.

Procedures: The wide collision attack is described as a three stage algorithm in [7]: an online stage where side channel leakage is measured for the chosen plaintexts, a collision detection stage which returns several pairs of plaintexts which most likely give wide collisions, and finally a key recovery stage. It has the same key recovery complexity as in earlier collision attacks: every 4 correctly detected wide collisions reveal four key bytes and a total of 16 wide collisions allows full key recovery.

In the most important collision detection phase, the collected measurements are partitioned into different bins, which are formed in the way that certain bytes of the intermediate state of the cipher collide to the same value. The partition follows the basic idea that when the same value is processed by certain operations (e.g. **MixColumns**, **SubBytes** in AES) of the cryptographic primitive, the pattern of the power consumption of these operations should be highly similar. Consequently, all possible 256 values of a given byte give rise to 256 bins, hence 256 different patterns for each colliding byte position. In a wide collision scenario, traces of each bin provide at least 5 internal byte collisions, spanning from **MixColumns** in round one up to **SubBytes** in round three of an AES execution. As a result, the side channel leakage for the two queries is homogeneous over this wide span in the time domain. Hence, detecting wide collisions should be much easier than detecting simple collisions.

Conventionally, leaking points or critical time points refer to a subset of samples in the side channel measurement (or its transformation), which represents the features of its pattern. Locating these points usually requires knowledge of the im-

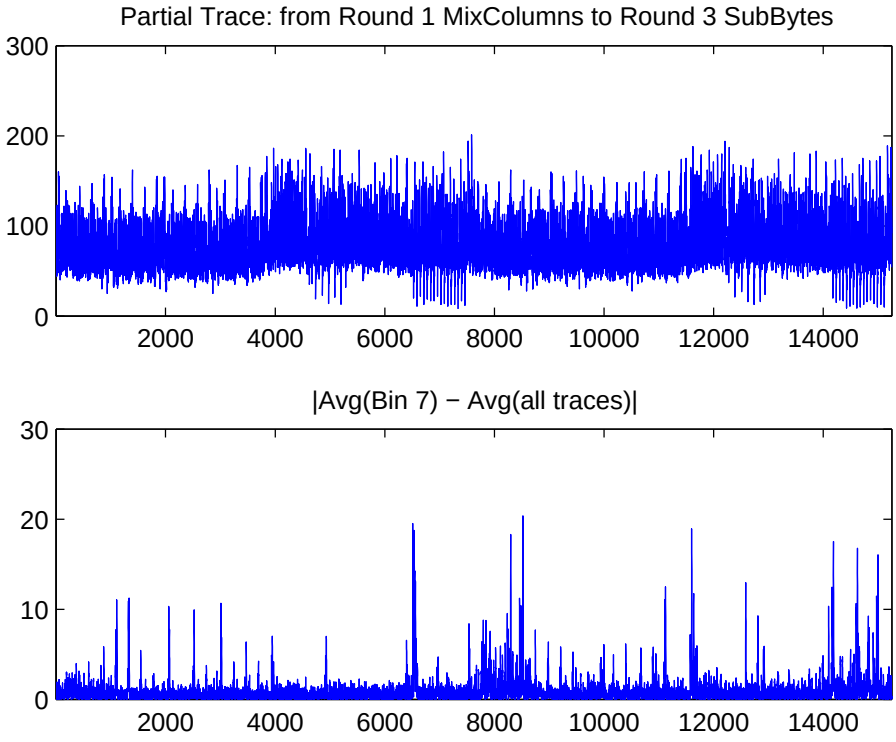


Figure 4.1: The pattern of a single power trace (upper) indicates possible cryptographic operations that are executed at different time. The peaks of the differential trace (lower) reveal the possible location where strong data dependency can be exploitable. X-axis represents time samples; Y-axis is the digital representation of single or differential power signal.

plementation and leakage properties of the platform or profiling. As an example, the two plots in Figure 4.2.2 show a single trace (the upper plot) over the region from round one **MixColumns** to round three **SubBytes** and a differential trace (the lower plot) calculated as the absolute value of difference between an average of traces of bin 7 and the average of all traces obtained. It can be seen from this figure that peaks, which indicate the location of promising leaking points, are spread out all over this region.

Another observation of leakages in the wide collision attack is that the positions of peaks are not invariant with respect to all wide collision bins. Some bins share one or more positions of leakages, while no pair of two different bins follows an identical pattern. Figure 4.2 gives an intuitive idea of the distribution of leakages for some bins.

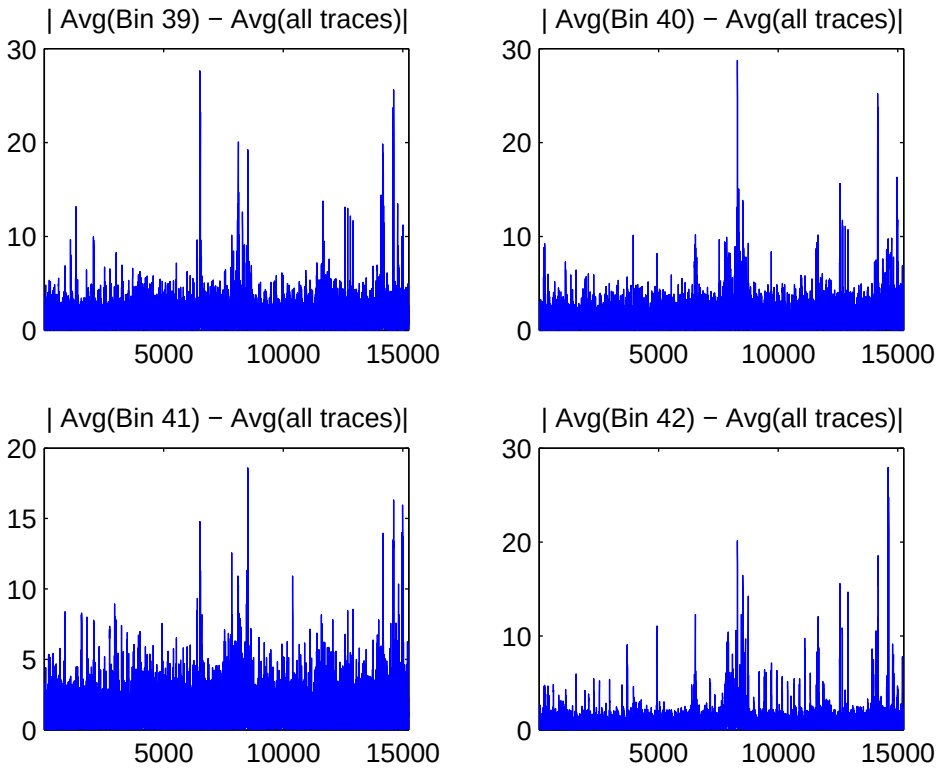


Figure 4.2: The leakages for different bin numbers (or internal values) occur at different time points. This observation makes it possible to distinguish different bin numbers using the power signal at these critical time points.

Hence, picking only a single point from all important points for the purpose of collision detection is rather risky because only few bins leak at this point. In other words, if the closeness of two traces at one fixed point is the only criterion for

wide collision detection, the detection can only succeed in rare cases because power traces of other bins that do not leak locally at this point are dominantly influenced by noise. Such traces make it difficult for a correct collision detection, since, while randomly scattered, they are far more numerous than colliding traces.

We distinguish two different kinds of multiple-point-based approaches: One builds on a template-based detection, the other does not require templates. When generating templates is not possible, we propose an outlier method which assumes that a pair of traces forms a collision in the case they are close to each other and simultaneously far away from the average of all traces. As a comparison, we show that wide collisions can easily be detected using templates. The challenge in this case is to discover the characteristics of each pattern and to correctly recognize each individual power trace from all the patterns with high probability.

Furthermore, we introduce the concepts of Inner-Bin-Variation and Inter-Bins-Variation as two parameters determining the effect of collision detection. We propose methods with the application of Principal Component Analysis (PCA) and iterative PCA so that the idea of maximizing Inter-Bins-Variation is realized.

4.2.3 Outlier Method

Generally, the outlier method assumes that two traces in the outlier region – with distance sufficiently far away from the average of all traces – are more likely to form a collision pair if additionally they are sufficiently close to each other. It includes one distance function $dist\,(\mathbf{L}_a, \mathbf{L}_b)$ that gives a distance metric between two trace representatives $\mathbf{L}_a$ and $\mathbf{L}_b$. It also includes two distance parameters R and r, where R is the outlier lower bound ratio determining if one trace is inside the outlier region,

and r is the mutual distance upper bound ratio determining if two traces are close by enough. Both R and r should be a number between 0 and 1. The procedure of the outlier method as follows:

Step 1: Compute the average $\bar{\mathbf{L}}$ of the q trace representations collected in the online stage using $\bar{\mathbf{L}} = \frac{\sum_i \mathbf{L}_i}{q}$ where $\mathbf{L}_i$ represents the side channel observation in i-th query at this point.

Step 2: Compute the distances vector $\mathbf{d} = (d_1, ..., d_q)$ where each entry $d_i = dist\left(\bar{\mathbf{L}}, \mathbf{L}_i\right)$ is the distance between the trace representation $\mathbf{L}_i$ and the average $\bar{\mathbf{L}}$ computed in Step 1.

Step 3: Find the set O_R of outliers by

$$O_R := \{\mathbf{L}_i \mid d_i \geq R \cdot \max(\mathbf{d})\}$$

It is a collection of trace representations with distance of no less than $R \cdot \max(\mathbf{d})$ from the average trace $\bar{\mathbf{x}}$. Figure 4.3 gives an example of the location of the outlier region.

Step 4: Find the list of pairwise distance

$$O_d := \left\{d^{(i,j)} = dist(\mathbf{L}_i, \mathbf{L}_j) \,|\mathbf{L}_i, \mathbf{L}_j \in O_A, i \neq j\right\}$$

Note that if there are n outliers in set O_R, then the set O_d contains distances of $\frac{n(n-1)}{2}$ pairs of traces.

Step 5: Find the set O_C by

$$O_r = \left\{ (\mathbf{L}_i, \mathbf{L}_j) \mid d^{i,j} \leq r \cdot \max(\mathbf{d}) \ , \ d^{(i,j)} \in O_d \right\}$$

This is a filtration from the set O_R of those pairs with mutual distance greater than $r \cdot \max(\mathbf{d})$. The set O_C is the output of the outlier method containing pairs of traces that are promising candidates for collisions.

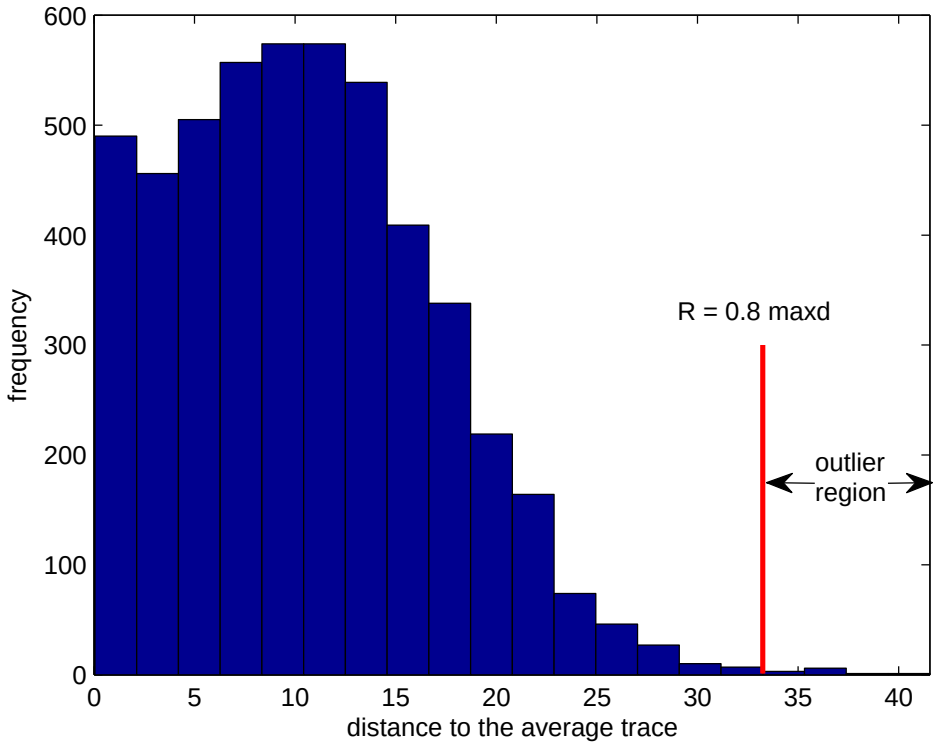


Figure 4.3: Each power trace over the critical time points is treated as elements in a finite dimensional vector space. With a distance metric such as the Euclidean distance, the radius or the maximum possible distance $\max(\mathbf{d})$ is computed. Given an outlier lower bound ratio R, an element at a distance more than $R \cdot \max(\mathbf{d})$ is called an outlier.

Please note that in general the distance function $dist(\cdot)$ is a combination of all components. In practice, one could use the Euclidean distance for the $dist$ function, viewing the trace representatives as elements in a finite dimensional vector space and assuming that components are independent from one another and each component contributes equally to the resulting distance.

Pros and Cons

The existence of leaking points is a necessary prerequisite for the wide collision detection. The points depend on the target device and implementation and should be chosen wisely by the attacker. Usually, either prior knowledge about the implementation or a profiling phase is needed. For detecting significant leakage points, SPA or DPA methods can be applied.

Choice of the parameters R and r is subjective. In practice, decreasing R while increasing r will eventually result in more non-colliding pairs that are detected as collisions. On the other side, decreasing r and increasing R will eventually increase to the set of detected collisions that does not contain enough pairs for key recovery. The Euclidean distance is a convenient metric, since it is a straightforward combination of the influence of each leaking point. However, it is often weaker than a template attack on the same points. Using Euclidean distance makes two additional non-justified assumptions: that the points leak independently of one another and that they contribute equally to the output distance. For improved detection results, it can be replaced by some function $g(\cdot)$ such that the influence of different points can be more accurately reflected.

4.2.4 Template Based Collision Detection

Inter-Bins Variation and Inner-Bin Variation

One way of selecting significant leakage points for collision detection is described in [8] and [9]. Both publications describe the *maxmin* function for selecting the most informative point of power traces and computing characteristics from traces at this point. Specific speaking, they first find for each fixed time point the lowest signal difference between all pairs of traces. They then find the time point that gives rise to the biggest one among all the lowest signal differences and consider that point to be the best choice. The logic of this method is that the lowest signal difference determines the level of difficulty of trace separation at each time point. The larger such lowest signal difference, the easier the separation of traces. However, this method makes use of a single point of the power trace, while ignoring all other remaining points. It yields for the attack a strong reliance on the single point that has been selected, which still suffers the risk of being influenced by the signal noise for each individual collision detection.

In contrast to single point selection, we propose an improved method. First, one should only include the leaking part of the power traces, i.e. the targeted round. For example, in wide collision attack, we only analyze the region starting from round 1 **MixColumns** to round 3 **SubBytes**. In this situation, two parameters – *inter-bins variation* (ITV) and *inner-bin variation* (INV) – determine the ease and probability of correct collision detection.

ITV describes the variation of the characteristics of the averaged traces $\mathbf{m}_i$ of

each bin value. It is computed as a Euclidean distance as

$$\mathrm{ITV} = \sqrt{\sum_i \left(\mathbf{m}_i - \bar{\mathbf{m}}\right)^2}$$

where $\bar{\mathbf{m}} = \sum \mathbf{m}_i/256$ is the mean trace of all the bin average traces. An increased ITV indicates an easier separation of the bins and more accurate pattern matching of each individual trace. Notice that the computation of ITV can also be applied to any representation of the traces. We refer the notion of maximizing ITV as computing the maximal ITV amongst all representations of the traces. Maximizing the ITV is therefore desired for the successful collision detection.

INV describes the variation of the characteristics of each individual trace $\mathbf{x}_j^{(i)}$ from that of the averaged trace $\mathbf{m}_i = \mathrm{avg}_j\left(\mathbf{x}_j^{(i)}\right)$ of a particular bin B_i. That is

$$\mathrm{INV}\,(i) = \sqrt{\sum_j \left(\mathbf{x}_j^{(i)} - \mathbf{m}_i\right)^2}$$

INV can similarly be applied to any representation of the traces and we refer minimizing INV as computing the minimal of INV amongst all representations of the traces. Note that INV is a tuple of 256 entries, corresponding to the inner-bin variation of 256 bins, and minimizing one entry does not imply small values for the rest of entries. Hence, although minimizing INV is desired, it might not be practically feasible.

One should note that the existence of the bin average traces does not necessarily guarantee the feasibility of its computation. In fact, only if one can build up templates for the 256 bins, one can also obtain a raw representation of the average traces of bins. Since each representation gives a computational result for the ITV,

the adversary would profit from a representation that maximizes the ITV.

In our experience, we realize the magnification of ITV through finding the representation of the average traces in the principal subspace, which are detailed in 4.2.4 and 4.2.4.

Templating using Principal Component Analysis

In cases where creating templates is possible, the attacker can build a template for each bin of collision in the time domain, as detailed in [38] and described in Section 2.3. Point selection can be automated by using *principal component analysis* (PCA) [4]. Template-based collision detection can achieve good detection rates, as shown in Section 4.2. PCA is a three step algorithm:

1. Finding the mean vector $\bar{\mathbf{L}}$ and the centered data matrix $\mathbf{L}' = \left(\mathbf{L}'_1, ..., \mathbf{L}'_q\right)^T$ of all the raw data record $\mathbf{L} = (\mathbf{L}_1, ..., \mathbf{L}_q)^T$, where $\mathbf{L}'_i = \mathbf{L}_i - \bar{\mathbf{L}}$;

2. Computing the covariance matrix $\mathbf{S} = \frac{1}{q}(\mathbf{L}')^T(\mathbf{L}')$ and its d eigenvectors $(\mathbf{v}_1, ..., \mathbf{v}_d)$ corresponding to the largest d eigenvalues $(\lambda_1, ..., \lambda_d)$ of $\mathbf{S}$;

3. Projecting $\mathbf{L}$ into the subspace spanned by the d eigenvectors (also called components) $\mathbf{L} \cdot (\mathbf{v}_1, ..., \mathbf{v}_d)$.

PCA performs an orthogonal projection into a subspace called principal subspace. The projection maximizes the variance of the data. Hence, a point selection with minimal information loss becomes possible.

Constructing templates in principal subspace is only one additional step to the build-up of the templates in the time domain. That is, the raw traces need to be projected into the principal subspace before the construction of templates. For

this step, the principal components could be obtained in two ways: from all the raw traces $\mathbf{L}$, or from the bin average matrix $\mathbf{M} = (\mathbf{m}_0, ..., \mathbf{m}_{255})^{\mathbf{T}}$, consisting bin average traces $\mathbf{m}_i$ of bin B_i where $\mathbf{m}_i = \text{avg}\{\mathbf{L}_j \mid \mathbf{L}_j \in B_i\}$. The consequence of applying PCA for the first choice is the maximization of the variance amongst individual traces and for the second approach amongst bin average traces. It is clear that the second method —computing principal components from bin averages— is desired because it achieves the goal of maximizing the ITV. After getting the projected traces in the principal subspace, the regular template building — the computation of bin averages and bin covariance — is performed as discussed in Section 2.3.

Finally, in the template matching phase, each analyzed trace is firstly projected onto principal components, then matched to the closest template through the evaluation of the probability densities. Every two different analyzed traces that are matched to the same template form a collision pair.

Collision Detection using Iterative PCA

A further improvement can be achieved by repeatedly conducting PCA. This gives rise to an iterative algorithm using projection. That is, in the template building phase, if two bins are too close or overlapping after the first PCA projection, one can repeat PCA projection (for which the computation of components only involves the average traces of that two bins) to further separate those two bins. The algorithm is given as follows:

Step 1: Use $\mathbf{M}^{(1)} = \{\bar{\mathbf{m}}_0, ..., \bar{\mathbf{m}}_{255}\}$ to compute the first set of principal components $\mathbf{V}^{(1)} = (\mathbf{v}_1, ..., \mathbf{v}_r)$ and the projection of each trace $\mathbf{P}^{(1)} = \mathbf{X} \cdot \mathbf{V}^{(1)}$.

Step 2: Partition the 256 bins into $C_\alpha^{(1)}$ and $C_\beta^{(1)}$ where bins from $C_\alpha^{(1)}$ can be clearly

separated from other bins, while bins from $C_\beta^{(1)}$ are still clustered with some other bins. That is, if bin $B_i \in C_\beta^{(1)}$, then there exists bin $B_j \in C_\beta^{(1)}$ such that traces of bin B_i are not separable from traces in B_j.

Step 3: If $C_\beta^{(1)}$ is not empty, compute the set $\mathbf{M}^{(2)}$ which consists of the averages of projected traces of non-separable bins in $C_\beta^{(1)}$

$$\mathbf{M}^{(2)} = \left\{ \bar{\mathbf{m}}_i = \text{avg}_j \left\{ \mathbf{p}_j \mid \mathbf{p}_j \in B_i \right\} \mid B_i \in C_\beta^{(1)} \right\}$$

Then based on $\mathbf{M}^{(2)}$, compute a second set of principal components $\mathbf{V}^{(2)}$ and obtain another set of projected traces $\mathbf{P}^{(2)} = \mathbf{P}^{(1)} \cdot \mathbf{V}^{(2)}$ from the previously projected ones.

Step 4: Repeat steps 2 and 3 until after k iterations $C_\beta^{(k)}$ is empty so that all bins are sufficiently separated.

4.2.5 Experimental Results

All experiments have been performed on a smart card featuring an 8-bit microcontroller based software implementation of AES. The measurements have been performed using a Tektronix digital sampling oscilloscope with an 8 bit A/D converter. The sampling rate of 50MS/s provides about 12 sampling points per clock cycle.

We first evaluate the detection rate of the outlier method. We explore the impact on the detection rate of several parameters: the number of promising leaking points, the choices of the outlier lower bound ratio R and the mutual distance upper bound ratio r, as well as the number of traces being investigated. Detection results are

shown in Tables 4.1 through 4.3. The first column contains the analyzed influencing factor. The second column is the average size of set A, i.e. the average number of the outliers, as described in Section 3.1. Again, if n traces were in the outlier region, $n\,(n-1)\,/2$ pairs are further analyzed by computing pairwise closeness. The third column shows the size of the set C, i.e. the average number of output pairs of promising collisions. The next column counts the number of correctly detected collisions, that is the detected pairs which actually form wide collisions. The last column is the ratio between the third and the fourth column, i.e. the ratio of correctly detected collisions.

For comparison we apply template-based collision detection in three different scenarios, (1) reduced templates in the time domain, (2) full template in time domain and (3) full template in the principal subspaces. Figure 4.4 shows how many traces per bin are necessary for training good templates and further indicates the asymptotic recognition rate for the three cases for our platform.

Results for the Outlier Method

We apply the outlier method as detailed in Section 3.1 to detect wide collision from the power traces. In our experiment, we define the distance function with the norm $\|\cdot\|_1$. That is,

$$dist\,(\mathbf{x},\mathbf{y}) = \|\,\mathbf{x}-\mathbf{y}\,\|_1 = \sum_i |x_i - y_i|$$

gives the distance from trace $\mathbf{x} = (x_1, ..., x_t)$ to $\mathbf{y} = (y_1, ..., y_t)$. In our experiments we use 3000 traces and fix the parameter R to 0.9 and r to 0.3. We locate between 1 to 8 promising leaking points to analyze the influence of the combination of leaking points. Table 4.1 confirms that using multiple points results in a better collision

detection rate comparing to the case of locating only one important point per power trace.

Table 4.1: Collision Detection: Impact from the Dimension of the Feature

# Points	# Outliers	# Detected	# True Collisions	Success Rate
1	19.6	127.7	21.9	23.0%
4	30.6	46.3	33.4	71.1%
6	110.7	126.3	105.4	86.2%
8	81.7	88.1	82.3	93.7%

Next, we explore different choices of the parameter pair (R, r) to analyze the effect on the collision detection rate. As explained in the algorithm in Section 3.1, R is the parameter determining which traces are in the region of "outliers" (the set O_R), sufficiently far away from the center of all traces. The larger R is, the fewer traces are considered as outliers. On the other hand r is the parameter that determines if two outliers are close enough to each other. The smaller r is, the fewer pairs of traces are detected as collisions, namely the smaller the cardinality of the set O_r . Our experiments use 3000 traces (the same as above) and fix 6 locations of promising leaking points. They confirm that the stricter the choice of (R, r), i.e. the larger choice of R and the smaller choice of r, the more accurate the detection is, as shown in Table 4.2.

As a last analysis of the outlier method, we explore the relationship between the successful detection and the number of traces being used in the experiment. In this experiment, 6 leaking points are fixed, the parameter R is set to 0.8 and r is 0.2. It is found that increasing the number of traces yields the increase in the number of outlier traces and the number of pairs being detected, meanwhile the detection rate

Table 4.2: Collision Detection: Impact from the Choices of R and r.

(R, r)	# Outliers	# Detected	# True Collisions	Success Rate
$(0.7, 0.2)$	382.1	807	551	68.4%
$(0.8, 0.2)$	110.7	126.3	105.4	86.2%
$(0.9, 0.2)$	19.9	8.3	7.7	89.6%
$(0.9, 0.3)$	19.9	16.1	12.9	81.3%
$(0.9, 0.4)$	19.9	22.9	13.9	60.8%

does not significantly increase, as shown in Table 4.3.

Table 4.3: Collision Detection: Impact from the Number of Traces

# Traces	# Outliers	# Detected	# True Collisions	Success Rate
1000	37.1	13.4	12.1	93.6%
3000	81.7	88.1	82.3	93.7%
5000	118.7	217.1	200.1	93.7%
7000	127.3	277	256.9	94.3%

Results for Template-based Detection

In the preceding outlier method, it is assumed that one can locate several leaking points and these points are independent of each other and contribute equally to the computation of distance function. The same assumptions hold if a reduced template attack is mounted in the time domain. But if a full template attack is applied, these assumptions do not need to be fulfilled. If the templates are built in the time domain, one only needs to locate good leaking points. While templates built in principal subspaces, as described in Section 3.3, even locating leaking points is no longer necessary. This is because the attacker can make use of all the region of power

traces that corresponds to wide collisions operations. Our experiment compares the method using reduced template and the full template to see how the dependency amongst leaking points helps with assigning an analyzed trace to its collision bin. We also compare the recognition rate between templates in the time domain and in the principal subspace through which we can verify that magnifying ITV enhances the recognition rate.

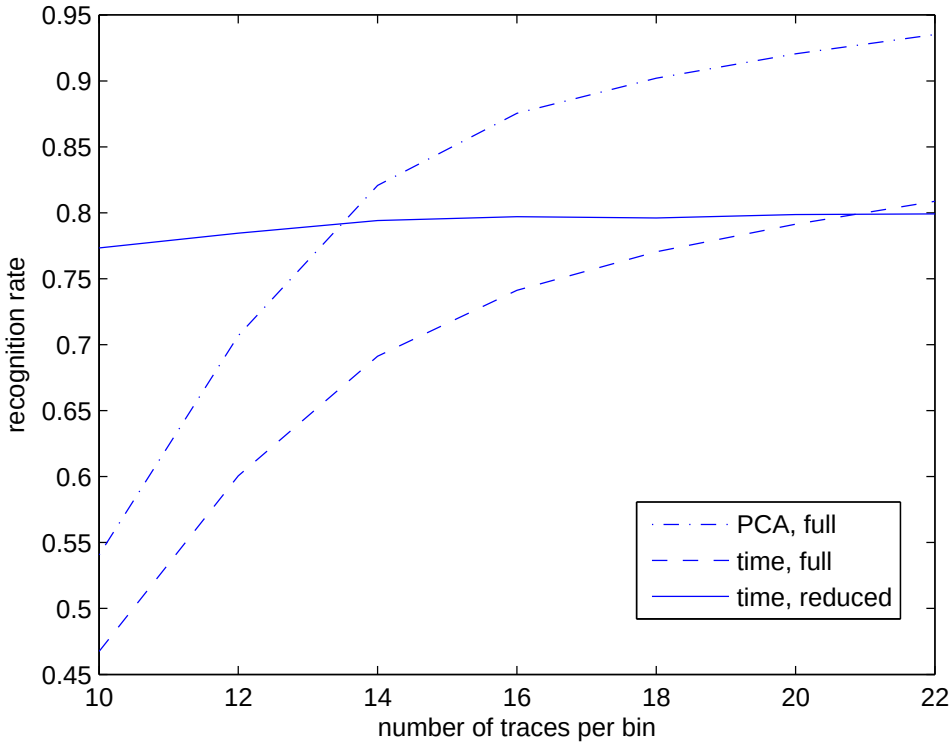


Figure 4.4: When templates are built over the selected time samples, the success rate of collision detection maintains around 77% (Y-axis). The number of training traces can be as low as 10. Including more time samples does not improve the success rate at all. When templates are built in the principal subspace with PCA, the success rate can reach as high as 95%, given more than 20 training traces.

Our experiments use 8 leaking points in the time domain. The counterpart in

PCA is 8 principal components that correspond to the 8 largest eigenvalues computed as specified in Section 3.4. We use 2560 to 5632 traces to build up templates so that each template makes use of 10 to 22 traces. We test 1000 traces to match to the templates and interpret #(*correct* recognition)/1000 as the recognition rate. From Figure 4.4 we can draw the following conclusions:

1. Using reduced templates in the time domain gives very stable recognition rate for a wide range of the number of used training traces for the templates. The asymptotic rate is at approx. 0.8. This is lower than the result for the two full models when sufficient training traces are provided. Therefore, the assumption of independence of leaking points cannot provide a strong collision distinguisher.

2. Full templates with PCA gain better recognition results comparing to the full templates in the time domain. Full templates in principal subspace gain close to 0.95 recognition rate given more than 20 training traces per bin. While full model in the time domain achieves only around 0.8 to 0.85. This confirms the contribution of PCA for maximizing ITV in terms of recognition.

3. Full models have stricter requirements on the number of training traces. In particular, if the attacker chooses n leaking points in the time domain or n principal components in the principal subspace, then the number of training traces per bin cannot be less than n, otherwise a singular covariance matrix C is an unavoidable result and this makes the computation of probability density infeasible. Even when the least number of training traces is satisfied, the computation of the covariance matrix can still be remarkably impacted by

the noise in the side channel. That is why the recognition rate for both of the full templates is low when fewer than 14 traces per bin are used.

4.3 Faster Leakage Detection

As mentioned earlier in this chapter, the efficiency of locating critical time samples is another challenging issue for leakage detection. In this section we first review earlier works on univariate and multivariate leakage detection. We then propose a novel technique called *leakage bundling* which enables efficiently locating univariate and multivariate leakages. We finish this section with experiments that validate the complexity reduction.

4.3.1 Related Works

On Univariate Leakage Detection

Given an unprotected implementation, the side channel leakage is of univariate nature as seen from the adversary knowledge model from Section 3.1.1. Processing a sensitive internal state Y results in a leakage L_j at a specific time point j. The leakage detection is to locate the critical time sample $t_y \in [1 : N]$ which provides maximum distinguishability of subkey hypotheses.

The first type of detection methods is provided by classical SCAs described in Section 3.1 where the leakage detection process is integrated with the leakage exploitation process. More specifically, the distinguisher is applied to leakage samples over each time point from 1 to N. For example, if the CPA is used, the goal is to find the time point j and the most promising subkey candidate g that gives higher

correlation coefficient $\rho(\tilde{\phi}(Y_g), L_j)$ between the prediction $\tilde{\phi}(f_g(X))$ and the observation L_j than any other subkey hypothesis. If mutual information is used as a side channel distinguisher, the goal is to find the time point j and the most promising subkey guess g that gives the highest amount of mutual information $I(\tilde{\phi}(Y); L_j)$ between the prediction $\tilde{\phi}(f_g(X))$ and the univariate side channel observation L_j. The pointwise operation implies a total of $|\mathcal{K}| \times N$ times hypothesis testing with each key hypothesis $g \in \mathcal{K}$ and each time point.

Another common technique inherits the idea of the *least square* optimization. With respect to the SCA scenario using the adversarial model in 3.1.1, it is to locate the time sample that maximizes the variance of side channel observation at all time points.

$$t = \text{argmax}_j \left\{ \sum_{i=1}^{q} \left(L_{i,j} - \mathbb{E}_i(L_{i,j})\right)^2 \right\} \tag{4.4}$$

The resulting time point shows maximum variability of all collected side channel traces. However it neglects the key distinguishing objective of SCA. One step further is to emphasize the knowledge of the input so that one first computes a group mean signal $\bar{L}_j^{[x]}$ for input-based query partition $G_x = \{i \in [1:q] | X_i = x\}$ at time sample j as

$$\bar{L}_j^{[x]} = \frac{1}{|G_x|} \sum_{i \in G_x} L_{i,j}.$$

After that the maximization using the least square can be taken with respect to the group mean signals, namely,

$$t = \text{argmax}_j \left\{ \sum_{x \in \mathcal{X}} \left(\bar{L}_j^{[x]} - \mathbb{E}_j(\bar{L}_j^{[x]})\right)^2 \right\} \tag{4.5}$$

The located time sample exhibits maximum separability of the leakage patterns

amongst all input partition groups.

A third common approach (c.f. [28]) is the *Welch T test*. The central interest is to determine whether there is univariate leakage amongst the side channel measurement. From statistical perspective, the Welch T test is interested in the problem of whether there is significant difference between the means $\mu_{\mathcal{A}}$ and $\mu_{\mathcal{B}}$ for two populations $\mathcal{A}$ and $\mathcal{B}$ based on a total of $N_{\mathcal{A}}$ samples $L_{\mathcal{A}}$ from population $\mathcal{A}$ and $N_{\mathcal{B}}$ samples $L_{\mathcal{A}}$ from $\mathcal{B}$. The test statistic is computed from

$$t = \frac{\tilde{\mu}_{\mathcal{A}} - \tilde{\mu}_{\mathcal{B}}}{\sqrt{\tilde{\sigma}^2_{\mathcal{A}}/N_{\mathcal{A}} + \tilde{\sigma}^2_{\mathcal{B}}/N_{\mathcal{B}}}} \tag{4.6}$$

where $\tilde{\mu}$ and $\tilde{\sigma}$ represent the empirical means and standard deviation as in equation 2.3 and 2.4. The tester checks whether $|t| > C$ for a pre-assigned parameter C for the purpose of significance test and confidence estimation. The samples from the two populations $\mathcal{A}$ and $\mathcal{B}$ are derived more or less like the original single bit DPA 3.1.3. One important difference is that there is no subkey hypothesis testing. The tester uses the knowledge of the key to compute the interested internal states Y and maps them to each bit. A natural partition of bit being 0 and 1 forms two populations, namely

$$\begin{aligned} \mathcal{A} &= \{L_{i,j} : i \in [1:q], \phi\left(f_k(X_i)\right) = 0\} \\ \mathcal{B} &= \{L_{i,j} : i \in [1:q], \phi\left(f_k(X_i)\right) = 1\} \end{aligned}$$

where the leakage model function ϕ is to take one bit of the internal values $Y = f_k(X)$. The T-test in the SCA setting also include Fixed-to-Random or Random-to-Random tests for population partitioning. More details can be found in original

proposal [28].

On Multivariate Leakage Detection

Masking is a popular countermeasure. As discussed in Section 3.1.1, it splits the secret internal state into multiple shares. Unless all shares are detected and recovered, the secret information remain a mystery. For software implementations of masking schemes, the shares are processed at different clock cycles. Detecting side channel leakage in this scenario is then equivalent to finding all time points $(t_0, t_1, ..., t_d) \in [1 : N]$ from the side channel measurements where the $d + 1$ shares are processed (t_0 refers to the time sample on which Y_M is processed).

Most of previous literatures assume that critical time points are known. Under this assumption, exploiting the multivariate leakage is equivalent to find a distinguisher that converts the information contained in the $d+1$ shares to key dependency and therefore reveals the key. The distinguisher here is also called higher order SCA (HOSCA). There are two main streams. One seeks for a combining function that convert the multivariate leakage into a univariate leakage and then apply classical univariate SCA to attack. Examples of combining functions are absolute difference of leakages at two distinct time samples [41] and the centered product of two points [56]. Recent paper [17] confirms the optimality of the choice of centered product combining function. The others [25] seek the information analysis from the distribution of leakages even in a multivariate setting.

In particular, the centered product combining function makes the following pre-

processing on the traces.

$$C(L_{t_0}, ..., L_{t_d}) = \prod_{i=0}^{d} (L_{t_i} - \mathbb{E}(L_{t_i})) \tag{4.7}$$

This combined leakage is then correlated with the predicted leakage model $\tilde{\phi}(f_g(X))$ for each subkey hypothesis g just like regular univariate CPA.

The information analysis approach seeks the multivariate mutual information analysis (MMIA). Works in [25] treat the d shares of leakages as a whole and intend to check how much information it infers the leakage model by evaluating

$$I(\tilde{\phi}(f_g(X)); (L_{t_0}, ..., L_{t_d})) \tag{4.8}$$

as a generic distinguisher. In other words, they compute the mutual information between the predictable power model and the actual multivariate leakage. Similar to the univariate MIA distinguisher, the power model cannot be the identity model if the target function f_g induces an one-to-one correspondence between the input and the algorithmic internal values.

Other proposals indicate to use MMIA as a leakage detection test. Since there is no subkey discrimination process, the power model can be an identity model. The objective becomes to quantify the information leakage from the empirical leakage distribution. The mutual information can either be between the measurement samples and either the algorithmic inputs or the internal values derived from the key. In fact, if the internal values and inputs are in one-to-one correspondence, the two computation does not differ at all. Another interesting work is [60]. It aims at finding promising tuples of time points where multivariate leakage might occur.

It considers the mutual interaction (as seen in equation (4.9)) amongst different shares as well as the algorithmic inputs.

$$I(L_{t_0}; ...; L_{t_d}; X) = I(L_{t_0}; ...; L_{t_d}) - I(L_{t_0}; ...; L_{t_d} \mid X) \tag{4.9}$$

One can recursively apply this equation until all terms is expressed as regular mutual information and conditional mutual information. The computation is not key dependent, and therefore only need to be executed once. After filtering time tuples that do not result in a significant negative values, classical multivariate SCA can be applied to distinguish the subkey. The main argument is the mutual interaction amongst shares become remarkable when conditioned at each particular input X. I.e. $I(L_{t_0}; ...; L_{t_d}) < I(L_{t_0}; ...; L_{t_d} \mid X)$ Consequently the interaction among the shares and the input should be a negative value.

4.3.2 Bundling Leakage Observation

Methods in all the above related works process the data pointwisely from the entire observed power traces. Detecting univariate leakage needs to traverse all N time points, while for multivariate leakage it needs to trial all $\binom{N}{d+1}$ combinations. In this section we introduce a novel technique called *leakage bundling*. Instead of processing each time sample independently, it treats a set of samples as a whole and process it with the leakage detection or exploitation method at adversary's or tester's choice. The expected benefit is to reduce the number of times of subkey hypothesis testing from the overall observations. Meanwhile, the expected disadvantage is the cumulation of the noises decreases the SNR and more traces are required to overcome the noise.

We denote $\Omega_0 = [1 : N]$ as the whole set of time samples and S be a subset of Ω_0. Bundling leakages for time samples in a set S means to compute the summation of each power trace over the time samples inside the set S.

$$L_S := \sum_{j \in S} L_j \tag{4.10}$$

It serves as a representation of leakage over the set of time points. Integrating the leakage bundling (LB) with SCA or tests is to determine whether leakage is more likely to occur in the set S or its complement S^c. We show how the technique can be applied for classical SCAs, Welch T test, and even HOSCAs.

Integrate LB with univariate SCA

Given an unprotected implementation, we can mount an fast SCA (given distinguisher $\mathcal{D}$) using the following iterative process. During the i-th iteration,

- **Step 1.** Obtain the half-and-half partition of Ω_{i-1}, namely $S_i = [\min(\Omega_{i-1}) : \max(\Omega_{i-1})/2]$ and $S_i^c = \Omega_{i-1} \backslash S_i$. Evaluate leakage bundles L_{S_i} and $L_{S_i^c}$ for the set S_i and its complement S_i^c from equation (4.10)

- **Step 2.** Make subkey hypothesis testing on the two leakage bundles separately using the distinguisher $\mathcal{D}$ at adversary's choice. That is, for each subkey guess g, compute the scores

$$R^g_{S_i} = \mathcal{D}(\tilde{\phi}(Y_g), L_{S_i}) \text{ and } R^g_{S_i^c} = \mathcal{D}(\tilde{\phi}(Y_g), L_{S_i^c});$$

- **Step 3.** Obtain the more favored scores R_{S_i} and $R_{S_i^c}$ for the two sets of hypothesis testing. They are computed as in the normal SCA that $R_{S_i} =$

$\max\{R^g_{S_i} | g \in \mathcal{K}$. Assign the next round time sample whole set Ω_i

$$\Omega_i = \begin{cases} S_i & \text{if } R_{S_i} > R_{S_i^c} \\ S_i^c & \text{otherwise} \end{cases}$$

The iteration terminates when the set Ω_i becomes a singleton set, i.e. only contain one time point.

More specifically, in each iteration the adversary partitions the time sample whole set Ω_{i-1} into the first half S_i and the second half S_i^c. She wants to identify which one is the more probable region of the occurrence of the leakage. Using a predetermined distinguisher $\mathcal{D}$, she attempts regular hypothesis testing only on the two leakage bundles L_{S_i} and $L_{S_i^c}$ and obtains the scores $R^g_{S_i}$ and $R^g_{S_i^c}$ for each subkey hypothesis g. Based on the nature of the distinguisher, the most favored scores R_{S_i} and $R_{S_i^c}$ are returned for the two bundles. The subkey candidates g_α and g_β that achieve the scores are also returned. For example, in MIA, it returns the hypothesis that gives the maximum of mutual information; in CPA, the one that gives highest absolute value of correlation coefficient is returned; and in Kolmogorov Smirnov SCA, the guess that yields the minimum of KS distance is returned. Further, the two best scores are compared to determine whether it is S_i or its complement S_i^c is more likely to contain the critical time sample of the leakage.

Clearly, iterating the three step procedure provides a sequence of inclusive sets of decision: $\Omega_0 \supsetneq \Omega_1 \supsetneq$ .... The cardinality of each set Ω_i is reduced to a half of its predecessor Ω_{i-1}. And it quickly becomes a trivial case that contains only one time point. This time sample is returned to the adversary as the critical time instance where the internal Y is leaking information. At the same time, the respective

hypothesis g_α or g_β that wins in the last iteration becomes the final subkey decision.

Complexity: Leakage bundles can be precomputed using a binary tree structure at complexity of $O(\log N)$. Clearly, the total number of iterations is $\log_2 N$ as the size of decision set Ω_i is reduced by a factor of 2 after each session. And as in each iteration there are $2|\mathcal{K}|$ subkey hypothesis tests, the total number of subkey hypothesis tests is $2|\mathcal{K}| \times \log N$. Therefore, applying leakage bundling reduces the total number of hypothesis testing significantly.

However, one should notice that the essence of leakage bundling is to sum up observation sample points. Unavoidably, the noise is added up. As for example, if $L_1, L_2, ..., L_m$ are independent variables that follow Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$, then $\sum_j L_j$ follows distribution $\mathcal{N}(\sum_j \mu_j, \sum_j \sigma_j^2)$. The added up noise not only affects the correct hypothesis testing but also the incorrect ones. Section 4.3.3 explores the impact on the number of traces required from leakage bundling.

Potential Impact on Random Order of Execution: Randomizing execution order is a type of hiding countermeasure. It can significantly decrease the SNR to increase the minimum volume of the side channel measurement for a successful attack. By applying the leakage bundling, the leakage samples are summed up and therefore the critical time sample has more chance to be included in the bundle. The effect of hiding might be mitigated.

Integrate LB with Multivariate SCA

When the implementation is protected with masking schemes, the secret of the sensitive internal state splits into several shares. Therefore, the leakage detection in

this scenario refers the detection of the position of all the shares from all the time samples. Naturally, this results in the problem of the multivariate leakage detection. As mentioned earlier in this chapter, a naive approach for detecting $d+1$ shares of leakage from N time samples is of the so called combinatorial complexity $O(N^{d+1})$. Mounting higher order SCA is therefore a rather expensive process. In this section we try to apply leakage bundling to reduce the overall complexity to $O(N^d)\log(N)$.

The idea is simple. In order to find all the $d+1$ shares, we fix the first d shares and bundle the leakage samples to locate the last share. In formal, assume the first d shares $\mathbf{t}^d := \{t_1, ..., t_d\}$ are found with $t_j \in [1:N], t_i \neq t_j$, the last leakage sample t_0 lies in the remaining set $\Omega \backslash \mathbf{t}^d$.

We do the following modification to the three step procedure. To begin with, the set Ω_0 should exclude the points in $\mathbf{t}^d$. As a consequence, none of the set S_i, S_i^c, or Ω_i contain any known point t_j with $1 \leq j \leq d$. There is no change in step 1: the computation of leakage bundles L_{S_i} and $L_{S_i^c}$ over the two sets are the same as in Equation (4.10). In step 2, the leakage bundles L_{S_i} and $L_{S_i^c}$ together with observations at other known samples L_{t_j}s are used as inputs for the higher order SCA distinguisher to derive the subkey hypothesis scores. For example, if higher order CPA is used, then one computes the correlation coefficients for each subkey guess g as

$$R_{S_i}^g = \rho(\tilde{\phi}(Y_g); C(L_{t_1}, ..., L_{t_d}, L_{S_i})) \tag{4.11}$$

$$R_{S_i^c}^g = \rho(\tilde{\phi}(Y_g); C(L_{t_1}, ..., L_{t_d}, L_{S_i^c})) \tag{4.12}$$

where the combining function $C(L_{t_1}, ..., L_{t_d}, L_S)$ is computed from Equation (4.7).

If MMIA is used, then one computes the mutual information as

$$R^g_{S_i} = I(\tilde{\phi}(Y_g); (L_{t_1}, ..., L_{t_d}, L_{S_i})) \quad (4.13)$$

$$R^g_{S^c_i} = I(\tilde{\phi}(Y_g); (L_{t_1}, ..., L_{t_d}, L_{S^c_i})) \quad (4.14)$$

as defined in Equation (4.8). Finally there is nothing to be changed for step 3: the decision set is derived in the same manner as in the univariate case. And the iteration of such modified three step procedure will shrink the range of the location of the last leaking time point t_0. The final decision also reveals the decision of subkey.

Complexity: Computing Leakage bundles that exclude the known d shares is still trivial in with a binary tree structure at complexity of $O(\log N)$. The total number of iterations changes slightly from $\log_2 N$ to $\log_2(N-d)$ but this change is negligible since usually d is far less than the window size N. Remember the time points for the first d shares are assumed to be known. In fact, this assumption is realized at the cost of exhaustive searching the d shares from all possible N-choose-d combinations and hence still suffices the complexity of $O(N^d)$. Therefore, applying leakage bundling reduces the complexity benefit from $O(N^{d+1})$ to $O(N^d \log N)$ in terms of the total number of subkey hypothesis testings. For example, to tackle a first order masking protection, the adversary breaks down the complexity of the problem of locating a bivariate leakage from a quadratic level to an almost linear level. This might re-alert the security evaluation of a masking scheme.

Integrate LB with Univariate Leakage Tests

We have seen that the central objective of the side channel testing is to determine the existence of leakage in the side channel time series. In addition to the yes or no answer, a significance level or a confidence interval is also expected. This is because the quantities of type I and type II errors are highly interested. Therefore, high volume of side channel measurements is often available. The two main stream leakage detectors are the Welch T test [28] and the mutual information based test [13] as compared in [39].

In this section, we show how to apply leakage bundling for univariate leakage tests with the following procedure.

- **Step 1.** Evaluate leakage bundles L_{S_i} and $L_{S_i^c}$ the same as before.
- **Step 2.** Evaluate the test scores for T_{S_i} and $T_{S_i^c}$ with the testing operator $\mathcal{T}$

$$T^g_{S_i} = \mathcal{T}(\tilde{\phi}(Y_k), L_{S_i}) \text{ and } T^g_{S_i^c} = \mathcal{T}(\tilde{\phi}(Y_k), L_{S_i^c});$$

- **Step 3.** Assign the set that corresponds to the more favored score as the next round whole set Ω_i

$$\Omega_i = \begin{cases} S_i & \text{if } T_{S_i} > T_{S_i^c} \\ S_i^c & \text{otherwise} \end{cases}$$

Comparing to the three step procedure for LB-SCA, the main difference is on the second step. Remember there is no *subkey* hypothesis testing but only *leakage* testing. If Welch T test is used, then Step 2 should return the t-scores for the two populations for each of the two sets of leakage bundles. That is, one needs to

compute T_{S_i} and $T_{S_i^c}$, which can be evaluated from equation (4.6) where the two populations are defined as

$$\begin{aligned} \mathcal{A} &= \{L_{i,S} : \tilde{\phi}(f_k(X_i)) = 0\} \\ \mathcal{B} &= \{L_{i,S} : \tilde{\phi}(f_k(X_i)) = 1\} \end{aligned}$$

If MI based test is used, then Step 2 should return the mutual information $\mathtt{MI}_{S_i} = I(L_{S_i}; \tilde{\phi}(Y))$ (as well as $\mathtt{MI}_{S_i^c} = I(L_{S_i^c}; \tilde{\phi}(Y))$) between the leakage bundles L_{S_i}, $L_{S_i^c}$ and the model $\tilde{\phi}(Y)$.

Multivariate Leakage Testing: Testing leakages in the multivariate scenario is difficult. Probably the only known method that enables significance level or confidence interval estimations is through multivariate mutual information estimation. Yet another interesting approach is to locate time sample combinations that gives no advantage for side channel adversaries. Typically the work [60] belongs to this category. In order to ease the second order attack, it computes the *mutual interactions* $I(L_{t_1}; L_{t_2}; X)$ between the two time samples L_{t_1}, L_{t_2} and the algorithmic inputs X for eliminating impossible combination of bi-variate leakage samples. The computation can be seen in Equation (4.9). The main idea is that for fixed $X = x$, the leakage shares of the mask M and the masked state y_M are related by differing a constant $y = f_k(x)$ and hence the conditional mutual information $I(L_1; L_2 \mid X = x)$ significant non-zero. If the input X is not fixed (i.e. without using the knowledge each specific x), however, the mask M and the masked state Y_M are differing by a variable Y. As a consequence, the time sample L_1 and L_2 for processing the two shares are independent, i.e. $I(L_1; L_2)$ approximates to 0. In short, a successful

locating both of the shares features in a negative mutual interaction $I(L_{t_1}; L_{t_2}; X)$. This gives some rationals to eliminate impossible combinations.

This process may not be integrated with the leakage bundling, since the procedure is used to reduce the possible combinations rather than directly finding the actual time position of the shares. In other words, it is not necessarily true that the time samples L_1 and L_2 for the two shares gives smallest interaction quantity.

4.3.3 Experiments

In this section we run experiments to test the performance of leakage bundling when it is applied to leakage exploitation and detection. Targeting the univariate leakage scenario, we use the real measurement of the RjindaelFurious software AES [53] running on an 8 bit AVR microcontroller. Targeting the multivariate leakage, we use the measurements from DPA contest V4 [69], which is a software implementation of AES with the Rotate Sboxes Masking countermeasure. We run the SCA code written in Python 3.3 on a 64-bit Ubuntu server with 2.50 GHz Intel Xeon CPU and 64 GiB memory.

Univariate Leakage Scenario

First experiments are performed for the unprotected AES implementation. Figure 4.5(a) shows the univariate leakage exploitation with regular CPA and the leakage bundling CPA (LB-CPA) using 10,000 traces. The window size N is fixed to 10,000 time samples. Both attacks start with loading the measurements into the memory as a 10000×10000 matrix L, each row of which corresponds with one measurement while each column of which corresponds with one time moment. To

distinguish one correct subkey from $|\mathcal{K}| = 256$ subkey candidates, regular CPA runs $|\mathcal{K}| \times N$ correlation coefficient computations and each computation is assumed to cost t_c time. For LB-CPA, the precomputation of leakage bundle tree costs t_p and the LB-CPA itself costs $2|\mathcal{K}| \times \log N \times t_c$. Therefore, the overall improvement rate r equals

$$\begin{aligned} r =& \frac{|\mathcal{K}| \times N \times t_c}{2|\mathcal{K}| \times \log N \times t_c + t_p} \\ =& \frac{|\mathcal{K}| \times N}{2|\mathcal{K}| \times \log N + \frac{t_p}{t_c}} \end{aligned} \tag{4.15}$$

In our practical experiments, the timing costs excluding the data loading procedure for both attacks are 1192.31 seconds and 11.39 seconds respectively. Each correlation coefficient computation costs $t_c = 0.00046$ seconds while leakage bundling precomputation costs $t_p = 5.14$ seconds. The practical results comply with the equation (4.15) very nicely when plugged into with the above value and verify the effectiveness of leakage bundling.

We also observe that the regular CPA provides three main regions where the leakage occurs. Leakage bundling returns the last region. Moreover, the leakage bundling does not necessarily return the position where the regular CPA has the highest value of Pearson correlation coefficient. A zoom in Figure 4.5(b) confirms this observation. It may be caused by the partition of time samples in the middle of some clock cycles and the three step procedure determines as the next region the partition whose leakage bundle contains gives higher correlation although it does not contain the heaviest leaking point.

Next, we compare the success rates of regular CPA and LB-CPA when increasing

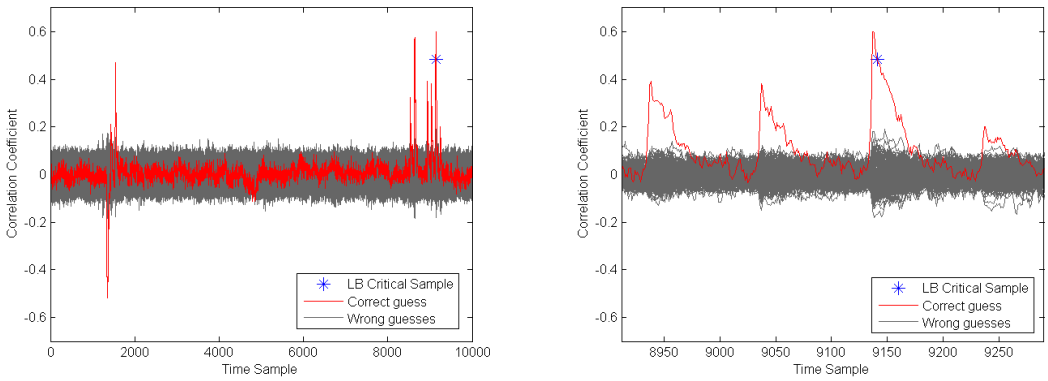


Figure 4.5: Univariate leakage detection using CPA and Leakage Bundling CPA, both applied to 10,000 traces with a window size of 10,000 samples per trace. Plot (b) is a zoomed version of (a).

the number of traces used to perform the attacks. Figure 4.6 shows that the success rate of regular CPA is already 1 with only 100 traces. In contrast, the success rate of LB-CPA is relatively very low as no more than 0.2 with 100 traces. As the number of traces increases to 1000, its success rate grows close to 1. This is because that while bundling leakage decreases the SNR, increasing number of traces will raise it up.

We apply the Welch t-test to determine the existence of the univariate leakage using 10,000 measurements. An illustration is shown in Figure 4.7 with the same window size of $N = 10,000$ samples. The curve represents a pointwise testing procedure and the * marks the time sample location returned from the leakage bundling. Similar to the exploitation case, the LB t-test successfully locates the time sample that has extremely high t-score.

In terms of complexity, the regular one consumes 26.5 seconds while the LB t-test costs 6.5 seconds, of which leakage bundling precomputation consumes $t_p = 6.39$

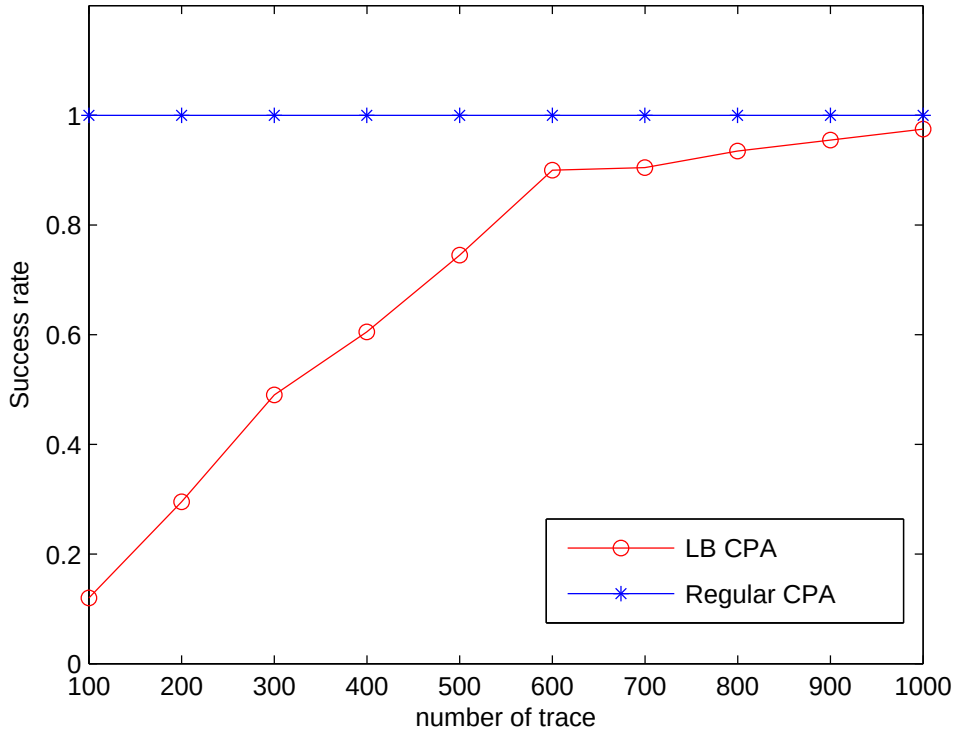


Figure 4.6: With the unprotected implementation, the regular CPA reaches almost 100% success rate (Y-axis) with the number of traces (X-axis) being as low as 100. The leakage bundling CPA reaches comparable success rate given much more traces. The efficiency in the time complexity is at the cost from larger number of measurements.

seconds and t test on each time sample costs $t_c = 0.0025$ seconds. Unlike leakage exploitation, the subkey is fixed as the correct one when applying t-test, hence $|\mathcal{K}|$ equals as 1 in this case. The practical results also comply with equation (4.15) very nicely when plugged into with the above value and verify the effectiveness of leakage bundling when applied to t-test.

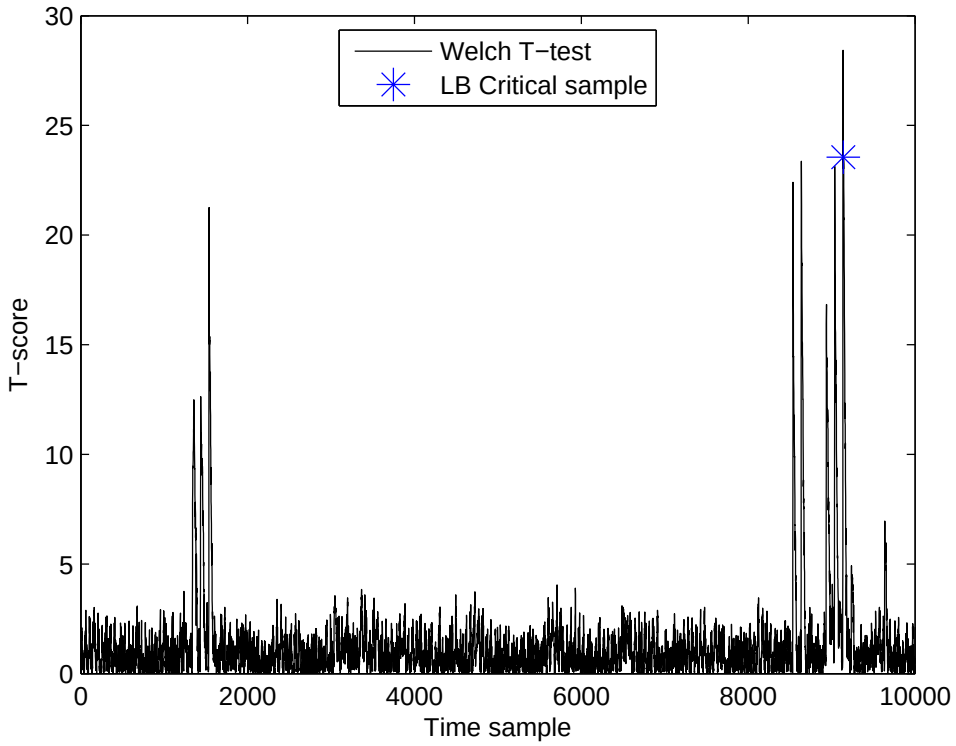


Figure 4.7: Integrating leakage bundling with Welch-t-test successfully captures univariate leakage (* point) for the unprotected implementation. The detected point does not necessarily correspond to the highest score (Y-axis) among the scores of regular Welch-t-test on all time points (X-axis).

Multivariate Leakage Scenario

Next, we combine leakage bundling with a second order CPA and apply it to the traces of the DPA contest V4, which applied a first order low entropy masking schemes. The available $40,000$ measurements are pre-processed using peak detection, effectively compressing the time domain from 435002 to 5661. With peak detection we implicitly assume a strong leakage to occur on the peak point of a clock cycle. Further, we select a window of size $N = 1000$ peaks based on some knowledge of the measurements to perform the analysis.

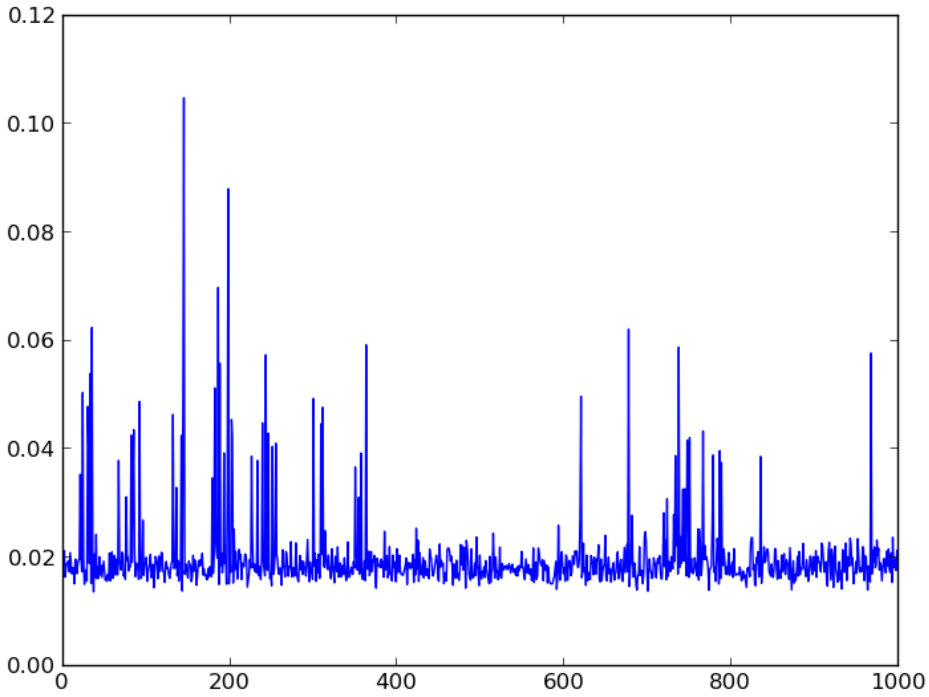


Figure 4.8: Applying leakage bundling in the 2nd order CPA reduces the complexity into $O(N \log N)$. For each data point on the curve, the X-axis fixes one time sample as a possible leakage. A total of $2|\mathcal{K}| \log N$ absolute values of correlation coefficients (Y-axis) are computed and compared to locate the second leakage. It also returns the highest absolute value as the Y-axis. The experiment is performed on 40,000 traces in DPA contest V4 with a window size being 1000 peak points. The hypothesis on the highest score is verified to be the correct key.

Figure 4.8 returns the correct key at more than 0.1 correlation coefficient. It should be mentioned that the 2 dimensional visualization is enabled because LB-2OCPA reduces the task of locating bivariate leakage samples from $N(N-1)/2$ to $2N \log N$. In fact, each data point (t, ρ_t) on the correlation trace is obtained by fixing one of the time sample at t and using binary search for the other point. Therefore the subkey decisions for each (t, ρ_t) may be different. In terms of time complexity,

LB-2OCPA costs 200 minutes to recover one subkey while regular second order CPA needs 8300 minutes.

4.4 Conclusion

In this chapter we have defined two challenging issues on the detection of side channel leakages. One is to reliably detect leakages without a-priori knowledge about the leakage model. The other is to improve the efficiency on locating critical time samples of leaking shares from the physical observables.

To address the first issue, we presented the wide collision detection method which reduces false positives and suffices an combined algebraic side channel attack. One major finding is the outlier method which shows that neighboring power traces within the outlier region exhibits a much higher chance of forming a collision pair. We have also shown that template-based approaches are a great method for detecting collisions provided that profiling is possible. Different ways of building collision-detecting templates are compared. Using PCA for finding independent strong leaking points seems to be better than hand-picking points in the time domain.

On the second issue, we have shown that classical SCAs and testing methods can be mounted with a sub-combinatorial complexity. The achievement of higher time efficiency is at the cost of lower SNR and hence more measurement requirement.

Chapter 5

Leakage Exploitation

We have seen that side channel analysis reveals information leakage from capturing data dependency. Moreover, it provides adversaries a simple divide-and-conquer approach to exploit the leakage and recover the secret key in a crypto system. Developing efficient and generic side channel distinguishers has been highly interesting to the community. Most classical SCAs introduced in Section 3.1 rely on the knowledge of the underlying leakage model. This is because a successful key recovery attack comes from a series of comparison between the observed side channel leakage and the hypothetical ones evaluated from the leakage model.

In this chapter, we first present the challenge and motivation for finding a side channel distinguisher that does not require the knowledge of leakage model. We then propose an novel approach that directly compares observations to observations without the need of leakage model. We provide two instantiations of this idea and show how they are applied in exploiting leakages for both unprotected and protected implementations. The advantage over traditional approach is also discussed.

5.1 Challenges and Motivation

SCA achieves its key recovery objective through exploring the data dependency between side channel observables and the internal state of the system. Such data dependency has usually been exploited in a series of observation-to-model (O2M as shown in Figure 5.1) comparisons with some particular leakage models in classical DPA (c.f. 3.1.3) and CPA (c.f. 3.1.4). Briefly, an adversary makes a sequence of comparisons between the observed leakage ($L_{k,p}$) and the hypothetical leakage value ($M_{g,p}$) under the subkey hypothesis g and leakage model M. The subkey hypothesis that produces the closest match between the observations and the model is selected as the correct key. A critical component in the O2M approach is the leakage model. The model can range from Hamming weight/distance models to more complicated toggle count models depending on the a-priori knowledge about the implementation. It is unavoidable that the leakage model differs from the reality. In an extreme situation, the error from leakage modeling assumption or the lack of detailed a-priori knowledge can aggravate or even prevent successful attacks.

Recent studies [59, 39, 19] call for generic distinguishers that do not rely on a-priori knowledge about the implementation and have minimum assumption on the leakage distribution. Perhaps the most generic distinguisher is the MIA (c.f. 3.1.6). Although it follows the information theoretic intuition, it is well known that the identity model with an injective target function [1] does not enable MIA's discrimination of wrong key candidates. This fact reveals the hidden requirement for a successful SCA – only the correct hypothesis can capture the meaningful data dependence. Recent theoretical study [77] even shows that generic univariate attacks

[1] the identity model together with an injective function implies a one to one correspondence, which assumes no a-priori knowledge of the leakage function

with a leakage model exist only for a very limited selection of target functions. It is indicated that profiling attacks such as template attacks (c.f. 3.1.5) and stochastic modeling attacks [63] are necessary for security evaluation. The profiling attacks require a separate profiling stage only for the purpose of precise estimation of leakage model, which is later used in the attacking phase to be compared with the observation. In short, profiling attack still follows an O2M intuition. Naturally comes the question whether knowing the leakage model is a prerequisite for a successful SCA. Or equivalently, are there any alternative approaches to the O2M comparison that also provides reliable key distinguishability?

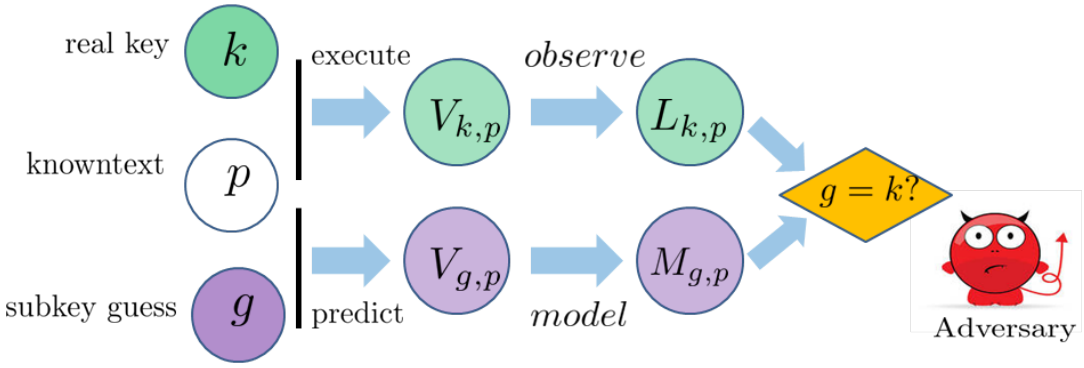


Figure 5.1: Classical SCAs make observation-to-model comparison to reject or accept subkey hypotheses.

In fact, a lesson we have learned from Section 4.2 is that processing the same values results in homogeneous leakage. It enables comparing side channel observations directly to another observation without even mattering the behavior of underlying leakage function. The attacking approach can be generalized as the **observation-to-observation** (O2O) comparison in contrast to the traditional observation-to-model comparison. The O2O approach allows direct comparison between one or

one set of leakage observation to another one or another set. The comparison incorporates subkey hypotheses hence the key distinguishing objective is still achievable. We propose a Non-Linear Collision Attack (NLCA) in Section 5.2 that exploits generic bivariate leakage. We also propose a series of distribution based collision attacks in Section 5.3 that even suffice univariate attacks on implementations with low entropy masking protection. Both of the works follow the O2O comparison and hence do not suffer from leakage modeling error.

5.2 Non-Linear Collision Attack

Although previously discussed collision attacks risk in potential false positive collision detection, they remind us that SCA can be mounted without estimating leakage models. A further step in resolving the false detection issue is the work [45]. It is the first work that not only takes advantage of side channel collisions but also makes the attack proceeding in a hypothesis testing manner. We first review the algorithm they provide.

5.2.1 Related Work: Linear Correlation Collision Attack

In [45], an interesting algorithm has been proposed to attack AES using correlation enhanced linear collision, which is called here the linear correlation collision attack (LCCA). 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, LCCA does not recover each subkey directly, but instead it tests hypothesis of the difference between subkeys as shown in Figure 5.2(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 $\rho(L_a^X, L_b^{X\oplus\delta})$ between the averaged leakage trace L_a^X of the byte-a-plaintext $X_a = X$ and the averaged leakage trace $L_b^{X\oplus\delta}$ 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. $\delta^* = \text{argmax}_\delta\{\rho(L_a^X, L_b^{X\oplus\delta})\}$. 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.

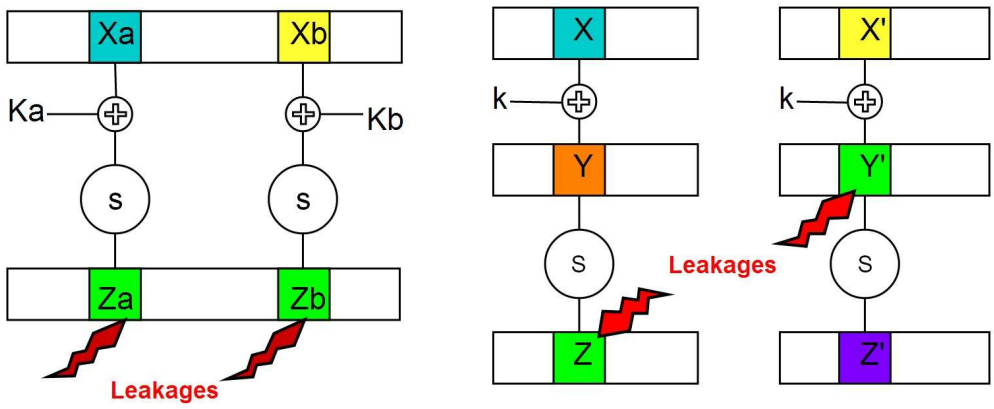
$$
\begin{aligned}
& X_a \oplus X_b = X \oplus X \oplus \delta = \Delta = k_a \oplus k_b \\
\iff\ & X_a \oplus k_a = X_b \oplus k_b \\
\iff\ & S(X_a \oplus k_a) = S(X_b \oplus k_b)
\end{aligned}
$$

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

5.2.2 Existence of Non-Linear Collisions

The LCCA takes advantage of the similar leakage behavior between linear collisions processed by the same operations. Now we show that the concept of exploitable

collisions can be 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 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.



(a) (b)

Figure 5.2: Linear Correlation Collision Attack (LCCA) (a) VS Non-Linear Collision Attack (NLCA) (b).

We introduce the following notations that are additional to the ones defined in Section 3.1.1. 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. 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 $Z = \tau(Y)$ is mapped through an intermediate non-linear function τ from the predecessor state Y. It is clear that the state Z is a functional composition output, represented as $Z = \tau \circ f_k(X)$. 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.

$$f_k(X') = \tau \circ f_k(X) \tag{5.1}$$

Clearly, if X' is chosen as one of the pre-images of $\tau \circ f_k(X)$, then it is a solution to equation (5.1). In other words, $X' \in f_k^{-1}(\tau \circ f_k(X))$ implies that the internal state $Y' = f_k(X')$ is guaranteed to be colliding with the internal state $Z = \tau \circ f_k(X)$. Hence the observed leakage behavior of L_Z and $L_{Y'}$ can be expected to be very similar.

5.2.3 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 τ as the first round[2] SubBytes. More precisely, we only consider τ 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 5.2(b) visualizes the idea of NLCA in this setting. The cross state collision in equation (5.1) becomes

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

$X' \oplus k = S(X \oplus k)$ and clearly it has a unique solution

$$X' = k \oplus S(X \oplus k) \tag{5.2}$$

In other words, if the AES encryption algorithm is executed with plaintexts X and X' computed from equation (5.2), 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 5.1 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. $k^* = \text{argmax}_g \{\rho(\alpha, \beta_g)\}$.

Algorithm 5.1 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^*

Validity

If the hypothesis is correct, i.e. $g = k$, the computed $X'_g = X'_k$ has the same format as in equation (5.2). It follows that

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

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.

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 [43]. The proposed non-linear collision attack can easily be extended to capture such hidden leakages. The adjustment is on line 3 of Algorithm 5.1. 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 [43].

5.2.4 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 makes 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 5.2.5.

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 [65, 64, 5, 6, 8, 79] 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, 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. Consequently, the LCCA is categorized by [73] 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 5.2(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 [45]. The NLCA, however, explores the *similarity* of leakage behavior caused by *different* operations. As can be seen from Figure 5.2(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.

5.2.5 NLCA-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.

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 [53]– 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

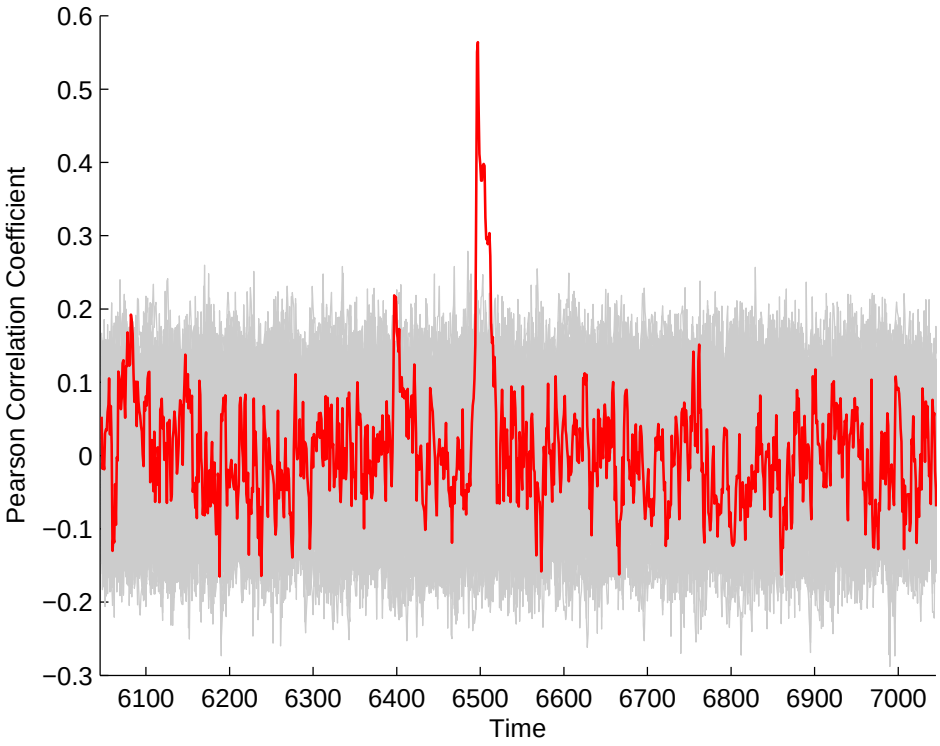


Figure 5.3: Use NLCA to distinguish the correct subkey and critical time sample.

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 5.1, 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 $t + 300$.

Figure 5.3 shows how NLCA distinguishes the correct subkey from wrong ones

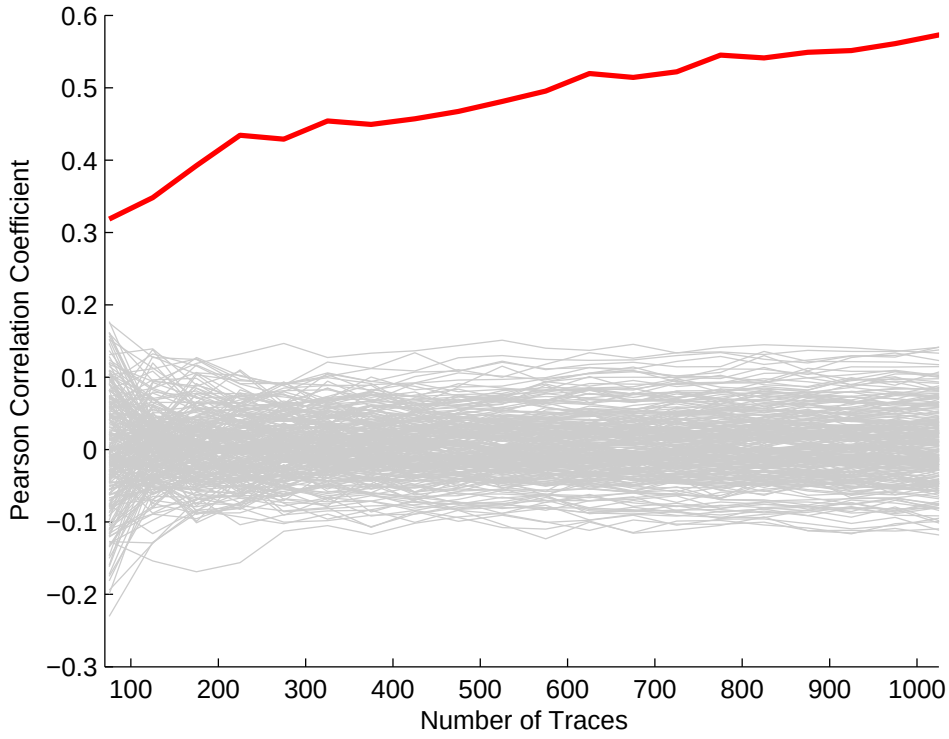


Figure 5.4: The number of observations that suffices NLCA

using 1000 traces. 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. The clear distinguishability of the correct Pearson correlation coefficient 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.

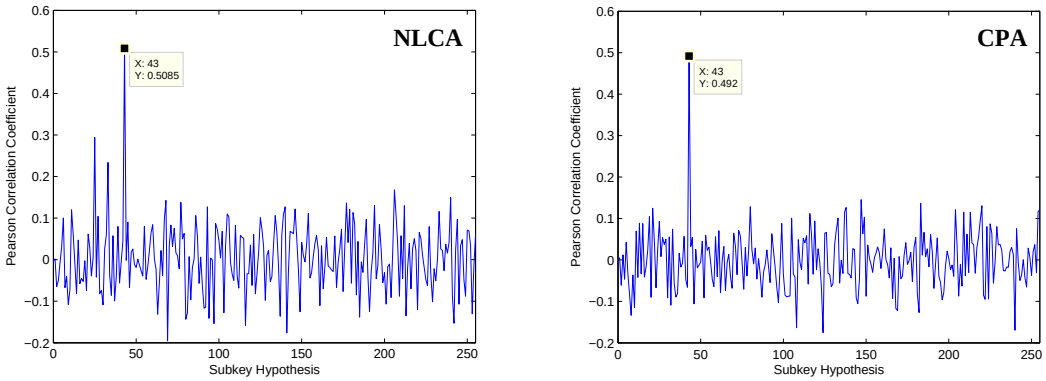
Next, the number of traces needed for a successful NLCA is explored. The correlation experiment is repeated on the discovered critical time point, as visualized in Figure 5.3, using 75 to 1000 traces. The observed trend is depicted in Figure 5.4. The correct hypothesis (red/dark) always features a higher 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 5.3. 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 5.5(a) and 5.5(b), both NLCA and CPA work well in this setting, outputting the correct subkey 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)

Figure 5.5: Performance of NLCA (a) VS classical CPA (b) over microcontroller measurements.

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).

Adversaries. 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 [3]. 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 [75] 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.

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

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

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

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

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

[3] As pointed out in [75], the near generic 7LSB power model for AES does not perform well for the MIA and it even fails catastrophically in strong signal setting

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 [66] to evaluate the subkey recovery performance of the four distinguishers – NLCA, CPA, UMIA and MMIA – as shown in Figure 5.6. They are tested with the three simulation scenarios – Optimistic (Upper), Realistic (Middle), and Challenging (Lower) – as discussed before. 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. 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

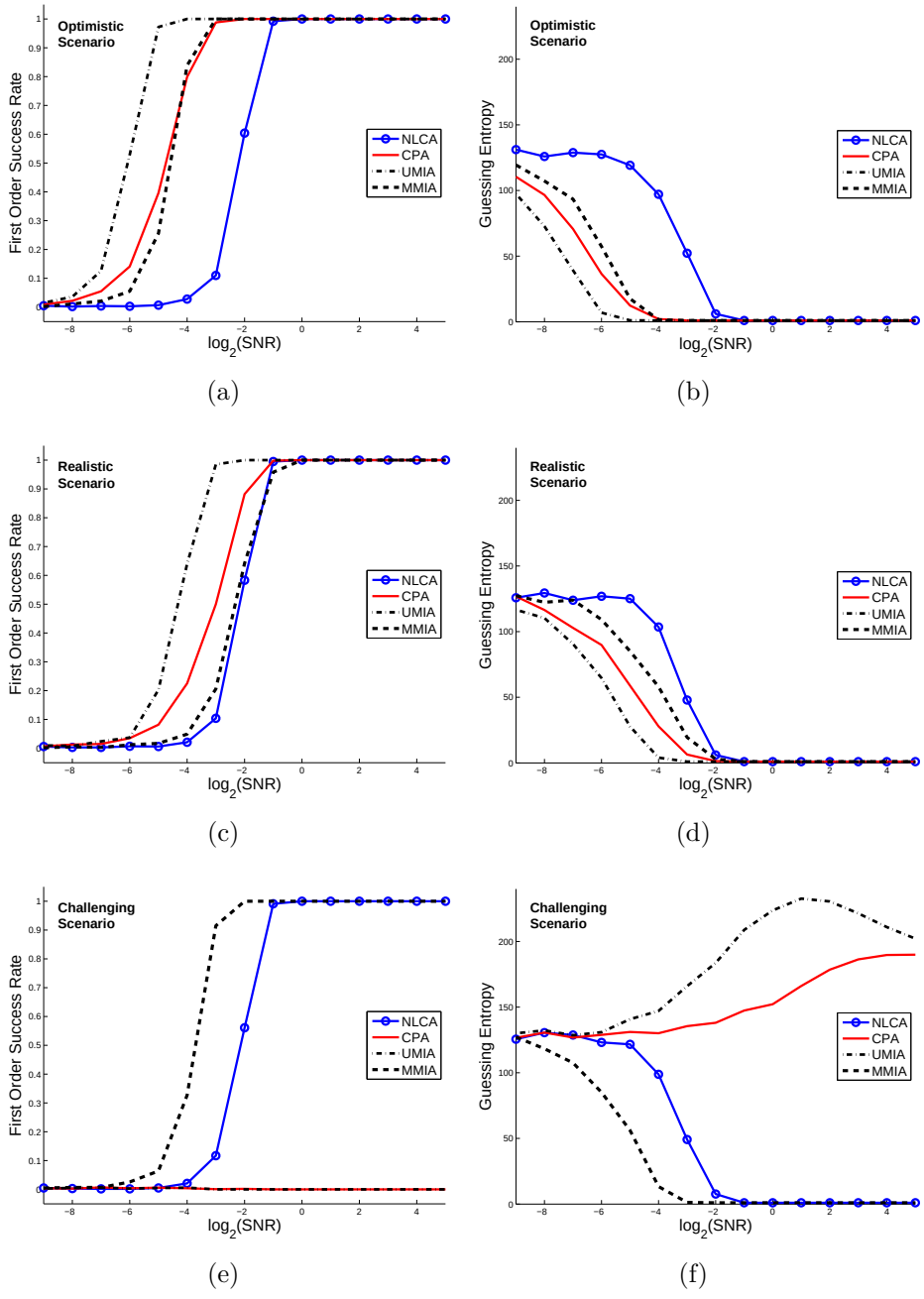


Figure 5.6: Robustness Tests for NLCA, CPA, UMIA and MMIA against Modeling Error.

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.

Impact of the Inhomogeneity of Leakages

As mentioned in Section 5.2.4, 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.

$$\phi_\tau(Y) = \phi(Y_L \| Y_R) = \phi(U) + \phi(Y_R) \tag{5.3}$$

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 (5.3), the Y_L is independent[4] 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 $\phi_\tau(Z) = \phi(V) + \phi(Z_R)$ 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 $\phi(Y'_R) = \phi(Z_R)$ while $\phi(U) \neq \phi(V)$. 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 ϕ is instantiated with the Hamming weight function. In the τ homogeneous setting, this means that the leakage function $\phi_\tau(Y) = \mathrm{HW}(U) + \mathrm{HW}(Y_R)$ generates Hamming weight of τ bits Y_R as signal, and the remaining random $8-\tau$ bits U give binomially distributed noise. The equivalent is true for Z. A total of 400 independent experiments are performed. Each experiment uses 256 simulated traces generated from the above defined leakage functions. The result in Table 5.1 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 [66] 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

[4]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 5.2.5.

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 applies in the same way to inhomogeneity of leakages in LCCA.

Table 5.1: The Robustness of NLCA under Inhomogeneous Leakage Behavior

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

5.3 Vulnerabilities of Low Entropy Masking Schemes

In this section, we show that the O2O comparison approach can also be used to exploit the leakage in the presence of side channel countermeasure. In particular, we study the low entropy masking schemes (LEMS) and propose a series of *univariate* attacks against them.

5.3.1 Low Entropy Masking Schemes

Like other masking schemes, the LEMS try to randomize the observed leakage by applying random values to intermediate states. With the notation introduced in Section 3.1.1, the algorithmic values Y are no longer processed and hence not produce leakage directly. The leaking states become the masked output Y_M. That is, the leakage generation function is changed to

$$L = \phi(Y_M) = \phi(f_k(X) * M) \tag{5.4}$$

However, the LEMS reduces the size of the mask alphabet, resulting in a limited extent of randomization of leaking states. For example, for a LEMS protected implementation of AES, the mask set $\mathcal{M}$ is a strict subset of $\{0,1\}^8$ such that the number of applicable mask values is much smaller than 256. The Rotating SBoxes Masking (RSM) scheme proposed in [47] is a realization of LEMS. It is a Boolean masking scheme that uses 16 mask values uniformly at random to protect AES internal states. In general, we denote the set of masks $\mathcal{M} = \{m_1, m_2, ..., m_s\} \subset \mathbb{F}_2^n$. We say a LEMS has masking entropy of $\log s$ if mask values are chosen uniformly at random from this set. The RSM is therefore said to have 4 bits of mask entropy. Furthermore, authors in [46] proposed a selection criterion of optimal mask values for LEMS. According to this guideline the following 16 byte values (written in hex format) are used as the mask set in the DPA contest V4.

$$\mathcal{M} = \{00,0F,36,39,53,5C,65,6A,95,9A,A3,AC,C6,C9,F0,FF\} \tag{5.5}$$

The 16 chosen values form an $[8, 4, 4]$ linear code (c.f. Section 2.1.1). It is therefore not surprising that they satisfies the *self-complementary* property: $\overline{\mathcal{M}} = \mathcal{M}$. Namely, $m \in \mathcal{M}$ if and only if $\bar{m} \in \mathcal{M}$, where $\bar{m}$ is the bitwise inversion of m.

The benefit of applying LEMS lies in the fact that it saves lots of computation when compared to a full entropy masking scheme (FEMS) where $s = 2^n$. The latter usually suffers from the huge amount of additional computation as a consequence of repeated masking/de-masking for the non-linear operation of a block cipher (e.g. Sbox in AES). One example is the Generalized Look-Up Table countermeasure proposed in [54]. It increases the size of a single Sbox sufficiently to make parallelized implementation of AES on FPGAs infeasible. However, with fewer masks, the total

number of necessary extra-computations can be kept at an acceptable level or even completed from pre-computations (e.g. defining masked sbox as look up tables). In short, LEMS enables more efficient implementation of a masking countermeasure. Unavoidably, applying LEMS causes some loss of protection when compared to FEMS. The natural question is how much security has been sacrificed and whether an attacker can construct an efficient attack to break the LEMS. Experiments in [47] show that RSM can resist univariate attacks including first and second-order DPA and CPA. The work uses MIA as the metric to get a quantification of 0.015 bit of information leakage in the described experimental setup, motivating a claim that such a low amount should be hard to exploit.

LEMS are designed to resist low statistical order DPA/CPA attacks while maintaining small computational overhead. The low level of leakage indicated by the mutual information $I(HW(Y_M); Y)$, as quantified in [47, 46], however, does not exclude the possibility of a univariate attack. In this section we analyze the composition of the leakage distribution under the protection of LEMS. We propose a univariate attack that can correctly decompose the observed one-dimensional distribution of leakage into several sub-distributions.

5.3.2 Leakage Distribution Composition

With the masking countermeasure, one algorithmic internal state Y can produce side channel leakage L through multiple leaking values $Y_{m_1}, ..., Y_{m_s}$. Consequently, the conditional entropy of leakage $H(L \mid Y)$ increases, making the classical attacks harder to succeed. According to equation (5.4) the leakage L depends on the knowntext X and the mask M, which are the main sources of entropy. If the knowntext

is fixed to one value $X = x$ at a time, the leakage entropy is lowered because only mask values are changed and LEMS only contains a small number of masks.

We use $\mathcal{D}^{X=x}_{M\in\mathcal{M}}[L]$ (or simply $\mathcal{D}^{x}_{\mathcal{M}}[L]$) to denote the leakage distribution under the condition that the knowntext X is fixed to x and the mask M is chosen uniformly at random from the mask set $\mathcal{M}$. In this situation, $X = x$ implies only one sensitive value $y = f_k(x)$ is to be protected by the masks, which results in the leaking set $(y)_{\mathcal{M}}$. Processing each leaking value y_{m_i} produces leakage $\phi(y_{m_i})$. The respective leakage observations form a leakage sub-distribution denoted by $\mathcal{D}^{X=x}_{M=m_i}[L]$ (or simply $\mathcal{D}^{x}_{m_i}[L]$) [5]. Since the leaking set $(y)_{\mathcal{M}}$ contains s leaking values, the observed leakage distribution $\mathcal{D}^{x}_{\mathcal{M}}[L]$ is a composition of s sub-distributions, namely,

$$\mathcal{D}^{x}_{\mathcal{M}}[L] = \frac{1}{s}\sum_{i=1}^{s}\mathcal{D}^{x}_{m_i}[L] = \frac{1}{s}\sum_{i=1}^{s}\mathcal{D}[\phi(y_{m_i})] \tag{5.6}$$

This equality actually comes from the law of total probability, i.e.

$$\begin{aligned} p[L = l \mid X = x] &= \sum_{i=1}^{s} p[L = l \mid M = m_i, X = x] \cdot Pr[M = m_i] \\ &= \frac{1}{s}\sum_{i=1}^{s} p[L = l \mid M = m_i] \end{aligned}$$

simply because $\mathcal{D}^{x}_{\mathcal{M}}[L]$ has the same meaning as the pmf/pdf $p[L = l \mid X = x]$ and $\mathcal{D}^{x}_{m_i}[L]$ the same as $p[L = l \mid M = m_i, X = x]$.

It is important to see that in LEMS the distribution $\mathcal{D}^{x}_{\mathcal{M}}[L]$ with fixed input x is different from the overall leakage distribution $\mathcal{D}[L]$ where the knowntext is not fixed. The former is a mixture of only s sub-distributions, while the latter is composed

[5] The notation $\mathcal{D}^{X=x}_{M=m_i}[L]$ is of the same meaning of leakage distribution as $\mathcal{D}[\phi(y_{m_i})]$. Both describe the leakage for processing y_{m_i}. The former emphasizes leakage decomposition and the latter focuses on connecting with estimated sub-distributions.

of all 2^n sub-distributions caused by all 2^n leaking values. In fact, the proposed leakage distribution decomposition attack (LDDA) makes use of this difference to explore the weakness of LEMS. It also indicates that the univariate LDDA cannot be extended to attack FEMS where both $\mathcal{D}^x_{\mathcal{M}}[L]$ and $\mathcal{D}[L]$ are composed of 2^n sub-distributions and hence not distinguishable from each other.

5.3.3 Leakage Distribution Decomposition Attack

Prior to the attack, the adversary needs to estimate the sub-distributions $\mathcal{D}[\phi(v)]$ of leakage for each leaking value v. We discuss this issue in more detail in Section 5.3.3 and 5.3.3. Here the attacker is assumed to have already obtained a precise estimation of sub-distributions. We show how this idea of decomposition in leakage distribution converts to a side channel attack. For each subkey hypothesis g and each prefixed knowntext $X = x$, the adversary follows a three-step procedure.

1. Find the hypothetical leaking set $(\hat{y})_{\mathcal{M}}$;
2. Compute the hypothetical mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}]$;
3. Evaluate the distance $dist(\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}], \mathcal{D}^x_{\mathcal{M}}[L])$ between the mixture and the observed distribution.

More specifically, with the subkey hypothesis g for a subkey k, the adversary computes $\hat{y} = f_g(x)$ and its respective masked states $\hat{y}_{m_i} = \hat{y} * m_i$ for all $m_i \in \mathcal{M}$. Since each hypothetical leaking value y_{m_i} contributes as one component $\hat{\mathcal{D}}[\phi(\hat{y}_{m_i})]$ of the leakage distribution, the adversary rebuilds the hypothetical mixture of all the s sub-distributions as

$$\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}] = \frac{1}{s}\sum_{i=1}^{s}\hat{\mathcal{D}}[\phi(\hat{y}_{m_i})] \tag{5.7}$$

Next, the adversary measures the similarity of the hypothetical mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}]$ and the observed distribution $\mathcal{D}^x_{\mathcal{M}}[L]$. A distance metric $dist(\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}], \mathcal{D}^x_{\mathcal{M}}[L])$ is evaluated for this purpose. In general, a small value of the computed distance metric indicates the two distributions are close to each other. A typical instantiation of the distance metric is the Kolmogorov-Smirnov distance suggested by [74, 76], which is later used in our experiments.

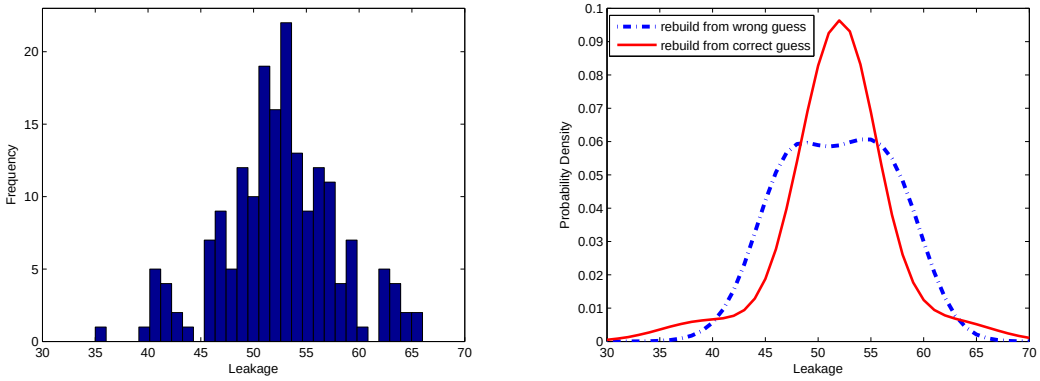
The adversary repeats the three-step procedure for all subkey hypotheses and all prefixed x. Her final decision for the correct subkey k is the hypothesis k^* that results in the lowest averaged distance as in equation (5.8). The attack is successful if $k^* = k$.

$$k^* = \underset{g}{\operatorname{argmin}}\left\{\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} dist(\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}], \mathcal{D}^x_{\mathcal{M}}[L])\right\} \tag{5.8}$$

Please note that the LDDA does not predict each individual *leaking state*. Instead, it analyzes the entire predicted *leaking set*. Figures 5.7(a) and 5.7(b) give an intuitive idea of how the decomposition of the observed distribution works for correct and incorrect subkey guesses.

Validity If the subkey guess g is correct, i.e. $g = k$, then $\hat{y} = f_g(x) = f_k(x) = y$ and the prediction of leaking set is correct $(\hat{y})_{\mathcal{M}} = (y)_{\mathcal{M}}$. Given precise estimations of sub-distributions, the rebuilt mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}]$ from equation (5.7) will be close to the observed distribution $\mathcal{D}^x_{\mathcal{M}}[L]$ because

$$\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}] = \frac{1}{s}\sum_{i=1}^{s}\hat{\mathcal{D}}[\phi(\hat{y}_{m_i})] = \frac{1}{s}\sum_{i=1}^{s}\mathcal{D}[\phi(y_{m_i})] = \mathcal{D}^x_{\mathcal{M}}[L]$$



(a) (b)

Figure 5.7: The observed leakage distribution (a) VS the hypothetical mixtures (b). The rebuilt mixture from the correct guess has a similar shape to the observed distribution while the mixture from the wrong guess is quite different from the observed distribution.

However if the hypothesis is wrong, i.e. $g \neq k$, then $\hat{y} = f_g(x) \neq f_k(x) = y$ for most of the inputs x, hence the hypothetical leaking set $(\hat{y})_{\mathcal{M}}$ has a low probability[6] to be the same as the actual leaking set $(y)_{\mathcal{M}}$. It follows that with high probability the rebuilt mixture $\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}]$ differs significantly from the observed $\mathcal{D}^x_{\mathcal{M}}[L]$ and hence their distance metric output should be large.

LDDA With Profiling

We have mentioned that the adversary should estimate the sub-distributions before mounting the LDDA attack. This has a straightforward solution by combining a profiling phase. More specifically, the profiling adversary is also assumed to have full control of the masks during the profiling stage – she knows each mask that is applied in each invocation. This assumption is frequently used in previous work [48,

[6] An exception is when $(\hat{y})_{\mathcal{M}}$ is a permutation of $(y)_{\mathcal{M}}$ for some particular g and x. Such exception occurs with small probability because the predicted leaking states take the range of entire $\{0,1\}^n$ rather than $\mathcal{M}$

34, 63]. The described attacks require at least bi-variate leakages consisting of the sample for processing the mask and the sample for the masked state. Hence, these approaches are not applicable for univariate attacks. Nevertheless, the profiling capability allows the adversary to build univariate leakage templates for each leaking value $v = f_{k'}(x) * m$ on another device that runs the same crypto algorithm with a different but known key k'. In other words, although the low entropy masking protection mingles different sub-distributions $\mathcal{D}[\phi(v)]$ together to achieve confusion, the assumed profiling adversary can still isolate each from the mixture. The isolated $\mathcal{D}[\phi(v)]$ can then serve as a sub-distribution look up table, enabling the adversary to rebuild the hypothetical mixture (the second step of LDDA) easily.

LDDA Without Profiling

Allowing the adversary having profiling capability is sometimes demanding. We show that sub-distribution estimation is also feasible for adversaries who are not granted with such privilege. This is achieved by assuming a leakage model and estimating the expression of leakage function explicitly. For a clear illustration, we assume the commonly accepted Hamming weight leakage model for a LEMS protected AES. It should be mentioned that advanced techniques of non-profiling leakage modeling such as linear regression model [18] may play a similar role if adjusted properly. With the Hamming weight model, the leakage L is expressed as a linear function of the Hamming weight of the leaking states with additive white Gaussian noise ϵ. I.e.

$$L = aHW\left(Y_M\right) + b + \epsilon$$

where the coefficients a, b are unknown constants, and the noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is mean zero and the noise level σ is also unknown. Since the sub-distributions $\mathcal{D}[\phi(v)]$ for processing leaking value v can now be represented as $\mathcal{N}(L; aHW(v) + b, \sigma^2)$, estimating sub-distributions is simplified to estimating the unknown parameters a, b, σ. Meanwhile, it is easy to see that the overall leakage distribution $\mathcal{D}[L]$ is a weighted composition of nine Gaussian curves. I.e.

$$\mathcal{D}[L] \approx \sum_{h=0}^{8} w_h \mathcal{N}(L; ah + b, \sigma^2) \tag{5.9}$$

where $0 \leq w_h \leq 1$ is the proportion of the normal curve $\mathcal{N}(L; ah + b, \sigma^2)$ and $\sum_{h=0}^{8} w_h = 1$. It follows that the Hamming weight h of leaking values Y_M forms a Binomial distribution and the weight parameters $w_h = \binom{8}{h}/2^8$, provided that the knowntext X is uniformly distributed. It is because xoring and SBoxing are one-to-one mappings. They deliver the uniform distribution from X to the sensitive Y and its masked output Y_M.

Finally, we solve the parameter estimation as an optimization problem. Optimal choices of a, b, σ should minimize the difference between the two sides of equation (5.9), namely, the observed overall leakage distribution and the composition of the parameterized sub-distributions. We set it as the objective function in equation (5.10). Furthermore, the optimization should be associated with the restriction that the statistical characteristics of the two sides should be approximately equal as in (5.11). Examples of the restriction functions are the statistical moments including $\text{Mean}(\mathcal{D}[L]) \approx 4a + b$, $\text{Var}(\mathcal{D}[L]) \approx \sigma^2 + 2a^2$ (derived from analysis of variance) and etc. The optimally parameterized $\mathcal{N}(L; aHW(v) + b, \sigma^2)$ can then serve as a

sub-distribution look up table, enabling the adversary to carry out the LDDA.

$$\texttt{Minimize}\ dist(\mathcal{D}[L], \sum_{h=0}^{8} w_h \mathcal{N}(L; ah+b, \sigma^2)) \tag{5.10}$$

$$\texttt{StatChar}(\mathcal{D}[L]) \approx \texttt{StatChar}(\sum_{h=0}^{8} w_h \mathcal{N}(L; ah+b, \sigma^2)) \tag{5.11}$$

It should be mentioned that the non-profiling LDDA is heavily influenced by the accuracy of leakage modeling. Large bias results in the derived sub-distributions being significantly different from the actual leakage function and hence reduces the efficiency or even disables the LDDA.

5.3.4 Leaking Set/Group Collision Attack

The previously discussed LDDA follows a 'decompose'-then-'rebuild' approach to compare the distributions of leakage. We now propose a second attack named leaking set collision attack (LSCA). It circumvents the 'rebuild' step and allows adversary directly comparing related distributions and therefore gains the benefit of avoiding the sub-distribution estimation.

Existence of Leaking Set Collisions

The approach extends side channel collision attacks [64, 45] by defining collisions between two leaking sets. Two distinct knowntexts $x \neq x'$ are said to induce a *leaking set collision* if the respective leaking sets are the same, i.e.

$$(y)_{\mathcal{M}} = \{y_{m_1}, ..., y_{m_s}\} = \{y'_{m_1}, ..., y'_{m_s}\} = (y')_{\mathcal{M}}$$

For Boolean masking schemes, the existence of leaking set collisions is a consequence of the *self-complementary property* for the choice of the mask values suggested in [46, 47]. It indicates that if m is chosen as a possible mask value, so should its bitwise inverse $\bar{m} = m \oplus 1^n$ as explained in Section 5.3.1 (1^n denotes the all-1 bit string, e.g. `0xff` for a byte). One simple choice is $y' = \bar{y}$. Because for any $m \in \mathcal{M}$,

$$\begin{aligned} (\bar{y})_m = \bar{y} \oplus m &= y \oplus 1^n \oplus m \\ &= y \oplus \bar{m} = y_{\bar{m}} \in (y)_{\mathcal{M}} \end{aligned}$$

This proves $(\bar{y})_{\mathcal{M}} \subset (y)_{\mathcal{M}}$. Similarly the other direction $(\bar{y})_{\mathcal{M}} \supset (y)_{\mathcal{M}}$ also holds and hence $(y)_{\mathcal{M}} = (\bar{y})_{\mathcal{M}}$. On the other hand, this choice $y' = \bar{y}$ identifies a relation between the respective knowntexts x, x' by setting $f_k(x') = \overline{f_k(x)}$. It is equivalent to

$$x' = f_k^{-1}(1^n \oplus f_k(x)) \tag{5.12}$$

It implies that the knowntext pair $\langle x, x' \rangle$ derived from equation (5.12) results in a leaking set collision between $(y)_{\mathcal{M}}$ and $(y')_{\mathcal{M}}$.

Building a Leaking Set Collision Attack

An importance consequence of the leaking set collision is that the respective underlying leakage distributions are identical. In fact, the set collision $(y)_{\mathcal{M}} = (y')_{\mathcal{M}}$ implies the both $\mathcal{D}^x_{\mathcal{M}}[L]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[L]$ have the same composition of sub-distributions.

$$\mathcal{D}^x_{\mathcal{M}}[L] = \frac{1}{s}\sum_{i=1}^{s} \mathcal{D}[\phi(y_{m_i})] = \frac{1}{s}\sum_{i=1}^{s} \mathcal{D}[\phi\left(y'_{m_i}\right)] = \mathcal{D}^{x'}_{\mathcal{M}}[L]$$

Therefore, the empirically observed leakage distributions $\mathcal{D}^{x}_{\mathcal{M}}[L]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[L]$ should be very close to each other. We now show how to convert this into a side channel attack against LEMS protected AES. The Sbox of the first round is chosen as the target function. Hence, the sensitive states y, y' are the s-box outputs and the knowntexts x, x' are the corresponding plaintext byte values[7]. The paired relation in equation (5.12) is then instantiated as in the following pairing equality in (5.13).

$$x' = \texttt{Pairing}(x, k) = k \oplus S^{-1}\left(\text{0xff} \oplus S\left(x \oplus k\right)\right) \tag{5.13}$$

It indicates that the plaintext pair $\langle x, x' \rangle$ which satisfies the paring equality forms a leaking set collision at their respective masked outputs.

The adversary, however, does not know the subkey k and cannot directly plug in the pairing equality to derive a collision. Nevertheless, she can make subkey hypothesis g and check for collisions just like a standard side channel attacker. A detailed attacking procedure is shown in Algorithm 5.2. It firstly sorts all leakages according to their respective plaintext x so that the empirical distributions $\mathcal{D}^{x}_{\mathcal{M}}[L]$ are obtained for all possible x. The adversary then starts testing subkey hypotheses. With each hypothesis g, she computes the hypothetical pairing $x' = \texttt{Pairing}(x, g)$ defined in equation (5.13). The two sets of related leakage distributions $\mathcal{D}^{x}_{\mathcal{M}}[L]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[L]$ are fetched and their similarity is measured using the distance metric $\text{dist}(\cdot, \cdot)$. In practice, the adversary can add up the computed distances derived from all possible collisions (line 8 of the algorithm). The decision strategy is similar to LDDA: the adversary determines as the correct subkey the hypothesis k^* that

[7]The same approach can be applied to arbitrary intermediate states, as long as they are a non-linear function of x and k: For states y that are linear functions of x and k, e.g. the s-box input, the key cancels out so that the knowntext pair become independent from the key, making the conversion into an attack infeasible.

results in the smallest overall distance. The attack is successful if $k^* = k$.

Algorithm 5.2 Leaking Set Collision Attack on RSM-AES

Input: Number of traces q; Knowntexts $x_1, ..., x_q$; leakages $l_1, ..., l_q$
Output: Subkey Decision k^*

Precomputation:
for $x = 0$ **to** 255 **do**
  $\mathcal{D}^x_{\mathcal{M}}[L] = \{l_i \mid x_i = x\}$ ▷ collect leakage whose knowntext is x
end for
Key recovery:
for $g = 0$ **to** 255 **do**
  $\delta_g = 0$
  for $x = 0$ **to** 255 **do**
    $x' = \texttt{Pairing}(x, g)$ ▷ compute hypothetical pairing x'
    $\delta_g = \delta_g + dist(\mathcal{D}^x_{\mathcal{M}}[L], \mathcal{D}^{x'}_{\mathcal{M}}[L])$ ▷ sums the distances from all pairings
  end for
end for
$k^* = \text{argmin}_g\{\delta_g\}$
return k^*

Validity If the key hypothesis is correct, i.e. $g = k$, then the derived pairing $x' = \texttt{Pairing}(x, g) = \texttt{Pairing}(x, k)$ is exactly the same as the true pairing equality in equation (5.13). It follows that a leaking set collision $(y)_{\mathcal{M}} = (y')_{\mathcal{M}}$ is generated. Hence the compared distributions should feature a low distance metric quantity $dist(\mathcal{D}^x_{\mathcal{M}}[L], \mathcal{D}^{x'}_{\mathcal{M}}[L])$. However if the subkey hypothesis is wrong, the computation yields

$$y' = S(x' \oplus k) = S(g \oplus S^{-1}(\text{0xff} \oplus S(x \oplus g)) \oplus k)$$

It is different from $\bar{y} = \text{0xff} \oplus S(x \oplus k)$ for most x. Hence the resulting leaking set $(y)_{\mathcal{M}}$ has low probability to completely overlap $(y')_{\mathcal{M}}$ and the two distributions have high probability to differ significantly.

Complexity It should be mentioned that the roles of x and x' of a hypothetical pairing are symmetric for any hypothesis. That is, if x' is a hypothetical pairing of x satisfying $x' = \texttt{Pairing}(x, g)$, then reversely x is also a pairing of x' satisfying $x = \texttt{Pairing}(x', g)$. Here is a simple proof.

$$\begin{aligned} x'' = \texttt{Pairing}(x', g) &= g \oplus S^{-1}(\text{0xff} \oplus S(x' \oplus g)) \\ &= g \oplus S^{-1}(\text{0xff} \oplus S(S^{-1}(\text{0xff} \oplus S(x \oplus g)))) \\ &= g \oplus (x \oplus g) = x \end{aligned}$$

This symmetry implies there are a total of 128 possible leaking set collisions for all 256 knowntexts x. It suffices to make only 128 distance comparisons for testing one hypothesis. Therefore the total complexity is 256×128 distance computations to recover one key byte.

Comparing LSCA with LDDA One common feature of LDDA and LSCA is that both attacks are achieved by comparing leakage distributions. More precisely, the compared leakage distributions refer to the leakages $\mathcal{D}^x_{\mathcal{M}}[L]$ with some prefixed knowntext x. It results in a lowered leakage entropy which become exploitable by the two attacks.

There are also many differences between the two attacks. Firstly, the LDDA compares empirically observed leakage distribution with the rebuilt hypothetical mixtures, while the LSCA compares two sets of distributions that are both obtained empirically. Therefore, the correct subkey hypothesis in the LDDA measures the closeness of the empirical distribution to its underlying distribution. In the LSCA it measures the closeness between two empirical distributions that are sampled from

the same underlying distribution. Secondly, the LDDA requires sub-distribution estimations to complete the "rebuild" step, while the LSCA avoids this. We have seen that estimating sub-distributions not only adds some complexity or even requires profiling privilege, but is also influenced by the accuracy of leakage modeling. Thus the LSCA does not suffer from the modeling bias. Last but not the least, the LDDA requires the mask set $\mathcal{M}$ to be known but the LSCA only requires the self-complementary property for the masking set $\mathcal{M}$. To sum up, the LDDA shows the explicit composition of leakages and the LSCA makes use of leakage composition implicitly and is more efficient in practice.

Leaking Group Collision Attacks

The DPA contest V4 actually advocates using the linear code or even its cosets as the set of mask. The algebraic structure is assumed to provide leakage resistance up to the fourth order statistical moments. We now present an even more powerful SCA to exploit the univariate leakage. We need two properties about linear code and cosets which are discussed in Section 2.1.1: the sum of two codewords remains as a codeword, the sum of two strings that belong to the same coset remains as a codeword.

First we consider using the linear code as the set of reduced masks $\mathcal{M}$. Given knowntext x, the state to be protected is $y = f_k(x)$ and the resulting leaking values are $y_{\mathcal{M}} = \{y \oplus m_1, ..., y \oplus m_s\}$. Let another knowntext be

$$x' = f_k^{-1}(m \oplus f_k(x)) \tag{5.14}$$

for some $m \in \mathcal{M}$. Clearly, it results in an algorithmic internal value $y' = y \oplus m$

which yields the leaking set being

$$y'_{\mathcal{M}} = \{y' \oplus m_1, ..., y' \oplus m_s\} = \{y \oplus m \oplus m_1, ..., y \oplus m \oplus m_s\} = y_{\mathcal{M}}$$

the same as the leaking set induced from the input x. Here the last equality just come from the linear code property of the masking set $\mathcal{M}$ that $m \oplus m_i \in \mathcal{M}$. It means that for any mask value m from the mask set, the input evaluated at equation (5.14) generates the same leaking set collision. The total s mask values give s inputs x' that produces a permutation on the leaking set $y_{\mathcal{M}}$. For example, all the 16 masks in the set (5.5) induce the same leaking set collisions for any fixed x. We call it a leaking group collision. It creates a partition on the input set $\mathcal{X}$ into $|\mathcal{X}|/|\mathcal{M}|$ groups, each (denoted as $U^x_{\mathcal{M}}$) of which contains $|\mathcal{M}|$ many knowntexts, i.e.

$$U^x_{\mathcal{M}} := \{f_k^{-1}(m \oplus f_k(x)) \ : m \in \mathcal{M}\}$$

It follows that $\forall x, x' \in U^x_{\mathcal{M}}$, the leakage distribution are the same $\mathcal{D}^{x'}_{\mathcal{M}}(L) = \mathcal{D}^{x}_{\mathcal{M}}(L)$ and hence the empirically obtained sample distribution should be close to each other.

Next, we consider using the same linear coset as the set of reduced masks, i.e. $\mathcal{M} = m \oplus \mathcal{C}$, where $\mathcal{C} = \{c_1, ..., c_s\}$ represents the linear code. To induce the same leaking set $y_{\mathcal{M}}$, we choose x' as

$$x' = f_k^{-1}(c \oplus f_k(x)) \tag{5.15}$$

for some $c \in \mathcal{C}$. The chosen x' gives rise to the internal state being $y' = c \oplus f_k(x)$,

which further yields the leaking set as

$$y'_{\mathcal{M}} = c \oplus f_k(x) \oplus m \oplus \mathcal{C} = y \oplus m \oplus \mathcal{C} = y_{\mathcal{M}}$$

being same as the leaking set induced from x. Similar argument provides an partition of the input set into $|\mathcal{X}|/|\mathcal{M}|$ groups, each (denoted as $U^x_{\mathcal{C}}$) of which contains $|\mathcal{M}|$ many knowntexts, i.e.

$$U^x_{\mathcal{C}} := \{f_k^{-1}(c \oplus f_k(x)) \ : c \in \mathcal{C}\}$$

Again, whenever $x, x' \in U^x_{\mathcal{C}}$, the induced leakage distributions are the same respectively. $\mathcal{D}^{x'}_{\mathcal{M}}(L) = \mathcal{D}^{x}_{\mathcal{M}}(L)$ and hence the empirically obtained sample distribution should be close to each other. The attack works in the same way as the LSCA. That is, subkey hypothesis yields knowntext partition. Only the correct subkey provides correct partition and resulting in homogeneous leakage distributions.

Different from the LSCA where only homogeneous distribution comes in pairwise manner, the knowntext partition here brings the benefits that the homogeneous distributions $\mathcal{D}^{x'}_{\mathcal{M}}(L)$ are uniquely determined by the leaking group $U^x_{\mathcal{M}}$ (or $U^x_{\mathcal{C}}$). It implies applying distribution distance testers such as the Kolmogorov-Smirnov distance can provide good performance while using fewer number of observations. For example, in the LEMS defined in equation (5.5), each leaking group $U^x_{\mathcal{M}}$ contains 16 knowntexts $\{x_1, ..., x_{16}\}$. We can firstly randomly mix 8 from the 16 distributions.

Without loss of generality, say

$$\mathcal{D}_A = \sum_{i=1}^{8} \mathcal{D}_{\mathcal{M}}^{x_i}[L] \tag{5.16}$$

$$\mathcal{D}_B = \sum_{i=9}^{16} \mathcal{D}_{\mathcal{M}}^{x_i}[L] \tag{5.17}$$

We then apply $dist(\mathcal{D}_A, \mathcal{D}_B)$ to measure their difference. If the subkey guess is correct, then the distributions to be mixed are homogeneous and the grouping does not change the distribution. If the subkey guess is wrong, then the distributions to be mixed are NOT homogeneous and the random grouping offer great chance such that the two mixture distributions differ significantly. Since mixing distributions provides more sample points for each mixture, the tests can have better performance.

5.3.5 Experiments

In this section, we carry out the LDDA and LSCA described in Section 5.3.3 and Section 5.3.4. Our experiments are performed on the measurements from DPA contest V4 [69]. It is a software implementation of AES-256 protected by the RSM countermeasure (cf. Section 5.3.1) and a total of 100,000 leakage measurements are provided. All attacks are performed on a univariate leakage sample representing the leakage of the first round AES output of SBox. Before showing our result we want to mention as reference that [47] shows 0 success rate for DPA,CPA and VPA based on 150,000 observations of a hardware implementation of RSM. It also reports 0.001 to 0.012 bit of information being leaked from mutual information analysis.

LDDA With Profiling

We firstly implemented LDDA using the template attack approach and we assume full knowledge of the mask application during the profiling stage as detailed in Section 5.3.3. A total of 50,000 measurements are used to build the templates, i.e. the 256 sub-distributions $\mathcal{D}[\phi(v)]$ of leakages for processing each possible leaking state v. The obtained sub-distributions are represented as 256 Gaussian curves $\mathcal{N}(L; \mu_v, \sigma_v^2)$. Upon the completion of sub-distribution estimation, another 2,000 to 16,000 measurements are used to test all 256 subkey hypotheses using the 3-step LDDA. In particular, the rebuilt distribution from each hypothesis is now instantiated as a Gaussian mixture,

$$\hat{\mathcal{D}}^x_{\mathcal{M}}[\hat{L}] = \frac{1}{s} \sum_{v \in (\hat{y})_{\mathcal{M}}} \mathcal{N}(\hat{L}; \mu_v, \sigma_v^2)$$

resulting in a model similar to [35]. The Gaussian mixture is compared with the observed distribution $\mathcal{D}^x_{\mathcal{M}}[L]$ using Kolmogorov-Smirnov (KS) distance metric.

Figure 5.8 shows the profiling LDDA hypothesis testing for the first subkey byte. We can see that LDDA succeed – the correct subkey $k = 108$ always gives the smallest KS distance among the 256 subkey hypotheses– whenever more than 2000 traces are used for testing. It verifies the correctness of the LDDA that only the correct hypothesis yields a correct decomposition of the leakage distribution. The four plots also show that the KS-distance drops when increasing the number of testing traces. In particular, the averaged KS distance for the correct subkey hypothesis drops from 0.273 all the way to 0.097. While for the wrong hypotheses, the average drops from 0.279 to around 0.112. The reason is that the computed distance depends

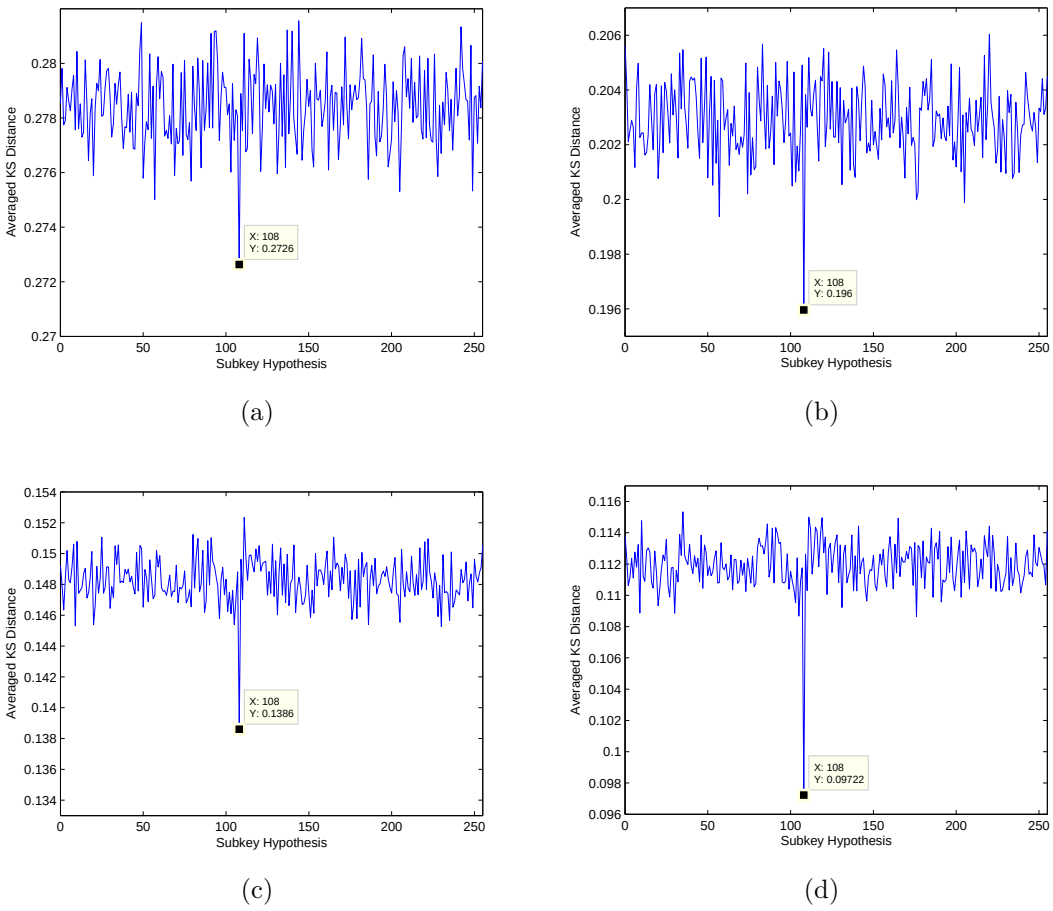


Figure 5.8: Profiling LDDA hypothesis testing: Kolmogrov - Smirnov distance (y-axis) between observed leakage distribution and rebuilt Gaussian mixture from all subkey hypotheses (x-axis) with the profiled sub-distributions. Experiments use 2,000 traces (a); 4,000 traces (b); 8,000 traces (c); and 16,000 traces (d).

on two main factors: *(1)* the correctness of the prediction of the leaking set $Y_{\mathcal{M}}$ (the effect exploited by LDDA); and *(2)* the sampling errors: viewing the observed leakage (the empirical one) as the samples from its underlying distribution (the true one approximated during the profiling). The law of large numbers implies that the empirical distribution of the leakage converges to its underlying distribution when increasing the number of leakages. Therefore, using more testing traces reduces the

influence of the sampling error and hence decreases the overall KS distance metric. As a consequence, the correct hypothesis becomes better distinguishable from the wrong guesses.

LDDA Without Profiling

Our second group of experiments carries out the LDDA without a profiling stage as described in Section 5.3.3. Each experiment estimate different combinations of the required parameters in the presented optimization problem in equations (5.10) and (5.11). The guessing entropy from [66] is used for the evaluation of the attack. That is, the subkey k is said to have guessing entropy t if the KS distance for k is on average the t-th smallest value among all KS distances for all hypotheses. Results are summarized in Table 5.2.

Table 5.2: Performance Evaluation for Non-Profiling LDDA

Number of Traces	20,000	40,000	60,000	80,000	100,000
GE (average case)	19.74	16.65	4.02	2.93	1.31
GE (worst case)	30	33	11	9	5
GE (best case)	9	2	2	1	1

Best case is evaluated with optimal estimation of parameters; Worst case is with non-optimal estimation.

It can be seen from the table that the correct subkey k has very low guessing entropy of 1 or 2 if more than 40,000 traces are used in the optimal estimation cases. Even for the worst estimation case shown in the table, the guessing entropy is still 33. The average estimation cases indicate that the non-profiling LDDA enables a reasonable attack – the guessing entropy is kept at an acceptable level– whenever

more than 60,000 measurements are used.

It can be seen that the non-profiling LDDA needs much more traces to succeed comparing to the profiling LDDA. Notice that the latter serves as the closest approximation to the real leakage function and the non-profiling LDDA here is merely derived from a coarse modeling of the leakage function – a noised linear transformation of the Hamming weight. The performance difference between the two methods indicates that a more precise estimation of sub-distributions yields better attacking performance for the non-profiling LDDA.

Leaking Set Collision Attack

The third group of experiments mounts the LSCA described in Section 5.3.4. Figure 5.9 shows the hypothesis testing of one LSCA attack using 10,000 to 40,000 traces. The correct subkey hypothesis $k = 108$ gives clear lowest KS distance metric when more than 15,000 traces are used. The distinguishability of the correct subkey increases with the number of traces that are used. Similar to the situation of profiling LDDA, we can observe a drop in the magnitude of the KS distance for the same hypothesis when the number of traces increases. The reason is still the reduction of sampling errors by using more traces.

In addition, we use the provided 100,000 traces to run as many independent experiments as possible for evaluating the LSCA attack. Table 5.3 summarizes the attacking performance using guessing entropy and t-th order success rate. It can be seen that the LSCA starts a stable success (GE = 1 and 1st order success rate is 100%) with more than 12288 traces, namely, 48 traces per plaintext byte. It is interesting to see that even with a total of 8192 traces (32 traces per plaintext),

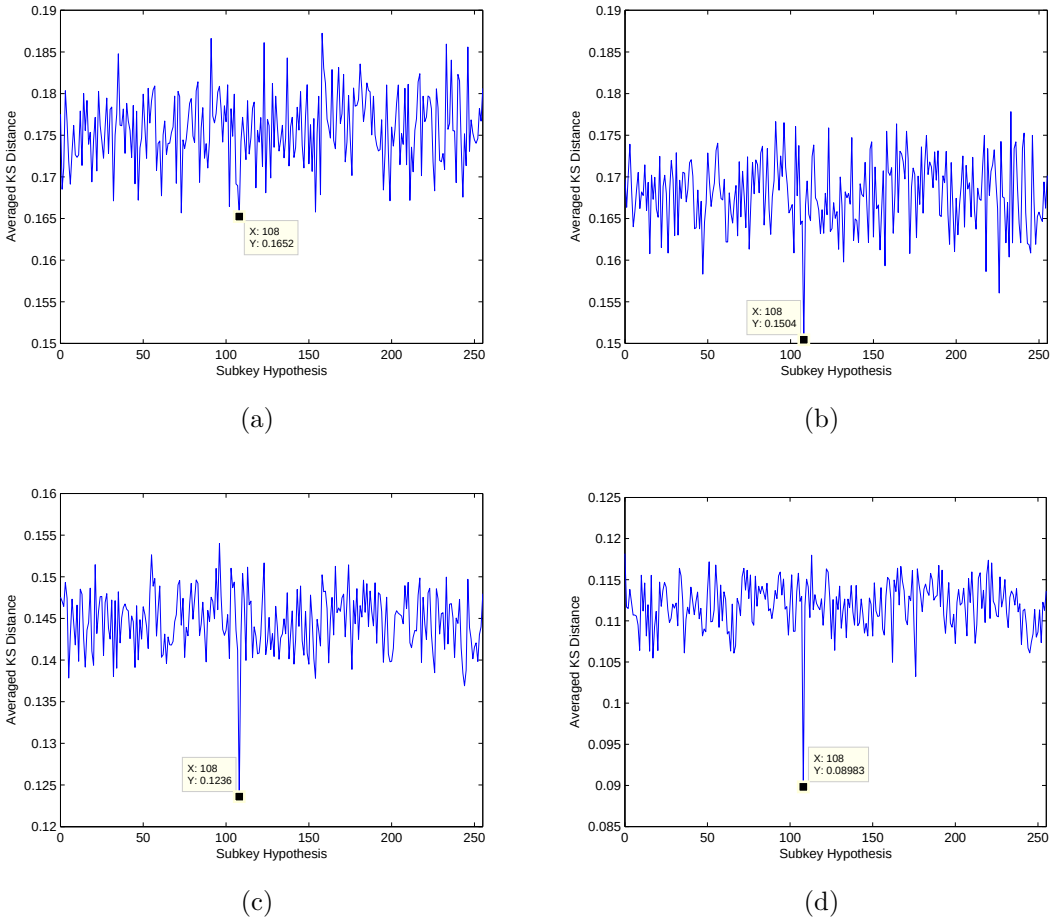


Figure 5.9: LSCA hypothesis testing: Kolmogrov-Smirnov distance (y-axis) between observed leakage distributions for the pairings induced from subkey hypothesis (x-axis). Experiments use 10,000 traces (a); 15,000 traces (b); 20,000 traces (c); and 40,000 traces (d).

making 4 guesses still ensures 2/3 success rate. The overall performance is much better than the non-profiling LDDA. However, the comparison with the profiling LDDA shows that the LSCA loses some success rate and requires more traces. The possible reason is that LSCA expands the sampling error. Since the two observed distributions $\mathcal{D}^{x}_{\mathcal{M}}[L]$ and $\mathcal{D}^{x'}_{\mathcal{M}}[L]$ are two sampling distributions from the same underlying distribution because of the set collision, the distance $dist(\mathcal{D}^{x}_{\mathcal{M}}[L], \mathcal{D}^{x'}_{\mathcal{M}}[L])$

is composed of two components: the distance from $\mathcal{D}^{x}_{\mathcal{M}}[L]$ to the underlying distribution and the distance from the underlying distribution to $\mathcal{D}^{x'}_{\mathcal{M}}[L]$. Although the compositional effect is not necessarily as strong as doubling the distance, it is very likely that the sampling error is expanded in the LSCA. While the profiling LDDA only measures one sampling error: the difference between the observed $\mathcal{D}^{x}_{\mathcal{M}}[L]$ and its underlying distribution. No expansion of sampling error occurs in the profiling LDDA. Nevertheless, with the slight sacrifice of success rate, the LSCA makes the full use of leakage similarity from the generated the leaking set collisions and therefore does not need assuming profiling capability nor the full control of masks.

Table 5.3: Performance Evaluation for LSCA

Number of Traces	4096	8192	12288	16384
Guessing Entropy	34.17	5.33	1.00	1.00
1st order Success Rate	0	33.3%	100.0%	100.0%
4th order Success Rate	33.3%	66.7%	100.0%	100.0%

5.4 Conclusion

In this chapter we have proposed a novel SCA approach, the observation-to-observation comparison. The main advantage is to bypass the complicated procedure of precise leakage modeling and make a successful SCA invariant with respect to modeling errors.

Two instantiations of the O2O approach have been proposed. The first one is the non-linear collision analysis. It is a technique of bivariate leakage exploitation targeting on unprotected crypto implementations. It generates internal collisions for different micro-operations in unprotected implementation and results in exploitable

bi-variate leakages. Experimental results from simulation have shown that the performance of the NLCA is immune to leakage modeling errors. It is also shown that inhomogeneous leakages generated from different operations have only low impact on the performance of the proposed attack.

The second instantiation includes several algebraic collision generation attacks. They belong to univariate leakage exploitation but targeting on protected implementation such as low entropy masking schemes. We have shown that the leakage that is reduced by the LEMS can still be exploited from analyzing the leakage distribution composition. Furthermore, the proposed leaking set and group collision generation attacks succeed even without the complicated leakage distribution estimation.

These attacks have shown that studying a countermeasure with resistance of the first, second or even higher order CPA/DPA is not sufficient to guarantee the resistance to other univariate attacks. Experimental results confirm the limited advantage from the profiling LDDA while the cost of modeling error is heavy as seen from the non-profiling LDDA. It is indicated that the O2O approach can be used to reveal improperly implemented masking countermeasure.

Chapter 6

Leakage Quantification

Earlier chapters have shown methods to detect and exploit information from side channel leakage. From the adversaries' perspective, those efforts are all made towards the ultimate goal of revealing the entire key. Therefore, side channel threat evaluation should be always orienting to the full key security. The basic task for a security evaluation is to address questions like how many observations ensure a successful full key recovery. A more important problem is to quantify the advantage that a probabilistic polynomial time (PPT) adversary can obtain from the maximum exploitable information. In this chapter we focuses on the problem of leakage quantification. We start with motivations and related works. Then we propose a novel constructive method and show how it can be used to determine whether bounded leakage enables full key recoveries.

6.1 Motivations and Related Works

Recently, there has been a growing interest in studying and quantifying the amount of information that can be extracted from a *limited* number of side channel observations. Knowing how much leakage actually suffices for a *full* key recovery is of high practical relevance. This question is closely tied to the computational capabilities of the side channel adversary, since SCA often includes an extensive key search component. A good comparison of algorithms using tradeoffs between side channel information and computation are the submissions to the DPA contest [68], where the success metric is solely based on the number of needed observations, without a clear limitation of computation. Another emerging trend in SCA are new attacks that are made feasible only by tapping into the massive parallel computing power as provided by GPUs, such as [44]. This indicates that computational power of the adversary needs to be considered as part of side channel security metrics. Finally, *leakage resilient cryptography* usually assumes limited leakage of a given key or secret state (c.f. [33, 22, 40, 67]) before it is updated. The schemes provide security if an adversary cannot successfully exploit more than the bounded leakage. In all of these cases, it is of high interest to know how much leakage the adversary can get from the observed measurements. Closely related is the question of the remaining attack complexity—given the limited side channel information—and the resulting search strategy.

So far only little effort has been put into the quantification of the remaining computational complexity when limited leakage is available but insufficient to narrow the key space down to a simply searchable size. While systematic metrics to quantify side channel leakage exist [38, 71, 61, 66], many of them perform rela-

tive comparisons of implementations or attacks [27, 36, 58]. The most promising approach has been presented in [72, 73]. The authors present a full key enumeration algorithm [72] as well as a key ranking algorithm [73] in the case where only limited side channel leakage can be extracted. These algorithms enable estimating the remaining full key recovery complexity even if the experimental verification is infeasible. However, their algorithms assume the correct key to be known. In other words, their results can be used by evaluation labs, but not by the key recovering adversary.

6.1.1 Full Key Ranking Algorithm

The aforementioned metrics have mostly been applied for *subkey* recovery experiments. This changed with the algorithms by Veyrat-Charvillon et al. in [72, 73]. The authors present algorithms to enumerate full keys [72] and to estimate the rank of the correct full key among all full key candidates [73]. With the latter algorithm they manage, for the first time, to approximate the computational complexity of successful side channel adversaries for cases where experimental verification is no longer possible or just too expensive. This means, the work pioneers in actually getting meaningful metrics for the *expected guesswork* of an adversary achieving *full key recovery*. Furthermore, they apply statistical bootstrapping to achieve cost evaluation and approximate a ML approach adversary for full key recovery.

The rank estimation algorithm [73], referred to as VGS algorithm, works as follows: As input it receives probabilities for all subkeys from a single side channel experiment, as well as the knowledge of the correct key (and consequently its probability). After sorting each of these subkey probabilities decreasingly, the different

dimensions are combined to create the key space. Next, volumes where keys have higher (or lower) probabilities than the correct key are removed from the space and their size is added to the lower (or upper) bound for the rank of the correct key. The VGS algorithm stops either after a set time or once the bounds are close enough, i.e. once the key rank has been narrowed down sufficiently. Finally, it outputs (upper and lower bounds for) the key rank of the correct key.

By itself, the key rank only provides the placement of the probability of the correct key. It cannot specify, in each *individual* side channel experiment, how much probability of success one can achieve by guessing full key candidates up to the correct key. Instead, the probability of success is derived by statistical bootstrapping: the side channel experiment is repeated e.g. $n = 100$ times, and the success probability is derived as the percentiles of the key ranks in *different* experiments are turned into success probabilities. The VGS algorithm is used for comparison and as a benchmark for our algorithm that we introduce next.

6.1.2 Security Metrics

For evaluating side channel security on subkey recovery, the framework in [66] proposes the (t-th order) Success Rate (SR) and the Guessing Entropy (GE). The t-th order SR is defined as

$$\mathsf{SR}^{k_i}(t) = \Pr[k_i \in \{g_{i,[1]}, ..., g_{i,[t]}\}]$$

It describes the probability that the correct subkey k_i is compromised in the first t prioritized guesses, namely, $k_i \in \{g_{i,[1]}, ..., g_{i,[t]}\}$. The GE describes the expected

ranking of the correct subkey. It is formulated as

$$\mathrm{GE} := \sum_{t=1}^{2^n} t \cdot \Pr[k_i = g_{i,[t]}]$$

Clearly, GE can be expressed as a function from the t-th order SR. The two metrics can be applied either to subkeys (as done in the above definitions) or to full keys. However, the latter is of little help in side channel analysis, since it sacrifices the divide-and-conquer technique and makes key spaces too large to derive meaningful results.

In addition, Pliam introduces *marginal guesswork*[1] in [52] as a metric to benchmark password recovery attacks, or more generically, *smart* exhaustive key searches. 'Smart' refers to adversaries that have and utilize prior information about the key distribution. Thus, marginal guesswork is well suited to describe adversaries that can assign probabilities to subkey candidates. In fact, it relates the success probability to its minimum computational complexity. More specifically, let $\sigma \in [0, 1]$ be the probability of success the adversary expects to achieve, the *σ-marginal guesswork* is defined to be the minimum number t of guesses to ensure finding the correct subkey k_i with at least σ success rate.

$$w_\sigma(k_i) = \min\{t : \sum_{j=1}^{t} p_{i,[j]} \geq \sigma\}$$

Here $p_{i,[j]} = \Pr[k_i = g_{i,[j]}]$ are the probabilities of the ordered subkey guesses. The candidates are ordered decreasingly in terms of likelihood, the same as before.

$$p_{i,[j]} \geq p_{i,[j+1]} \geq \ldots \tag{6.1}$$

[1] referred to as *work-factor* in [51]

This approach is also known as *Maximum Likelihood* (ML) attack. Based on side channel information, namely the inputs $\mathbf{x}^q$ and leakages $\mathbf{l}^q$, it first assigns posterior probability $p_{i,j} = \mathtt{Pr}[g_{i,j} \mid \mathbf{x}^q, \mathbf{l}^q]$ to subkey candidates $g_{i,j}$. Next, it enumerates them in a descending order $g_{i,[j]}$ according to the posterior likelihood $p_{i,[j]}$. Since $p_{i,j}$ is interpreted by definition as the likelihood of the true subkey k_i being the candidate $g_{i,j}$, the guess $g_{i,j}$ ensures subkey success rate $p_{i,j}$. Therefore the t-th order success rate using ML approach is

$$\mathsf{SR}^{k_i}(t) = \sum_{j=1}^{t} p_{i,[j]} \tag{6.2}$$

It also establishes a connection between the t-th order SR and the σ-marginal guess work as : $w_\sigma(k_i) = \min\{t : \mathsf{SR}^{k_i}(t) \geq \sigma\}$. To sum up, the adversary using *maximum likelihood* approach is expected to have the minimum complexity to find the correct subkey.

6.2 Evaluating Full Key Security

Side channel leakage enables assigning scores or posterior probabilities to subkey candidates. However, to verify the correctness of a guess, different subkey parts must be combined and checked. That is to say, as long as the leakage is not strong enough to reveal each subkey part with a negligible error probability, the remaining full key security is not trivially evaluated and is worthy of investigation. Conceptually, the ML approach can be extended to cover full key recovery attacks so that all the metrics described in Section 6.1.2 can also be applied to evaluate *full* key security. However, the size of the key space is 2^{bn}, e.g. in AES-128 it is 2^{128}, and it makes it infeasible to calculate the posterior probabilities to all full key candidates and then

to enumerate them strictly following the ML principle. In this section, we introduce a weaker but computationally efficient approach to evaluate full key security. We call this approach the weak Maximum Likelihood (wML) approach. We describe its basic idea, followed by a Key Space Finding (KSF) algorithm as its realization and explain how it differs from a true ML approach.

6.2.1 Weak Maximum Likelihood Approach

Since computing and enumerating probabilities for all full key candidates is infeasible, the adversary can, nevertheless, adopt the following straightforward strategy. For each subkey part k_i, the adversary only considers the top e_i subkey candidates. When making full key guesses, she checks the Cartesian product of such selected candidates from all subkey parts. More specifically, the adversary considers the prioritized guesses $\{g_{i,[1]}, ..., g_{i,[e_i]}\}$ for the true subkey part k_i and verifies all possible combinations $\{g_{1,[j_1]} \| ... \| g_{b,[j_b]}$ where $1 \leq j_i \leq e_i, 1 \leq i \leq b\}$ as full key candidates. It is clear that this approach ensures a subkey success rate of $\mathsf{SR}^{k_i}(e_i)$ with e_i guesses for the subkey part k_i. Therefore, a full key success rate of $\prod_{i=1}^{b} \mathsf{SR}^{k_i}(e_i)$ is achieved, implying a full key verification cost of $\prod_{i=1}^{b} e_i$. The vector $\mathbf{e} = (e_1, ..., e_b)$ is called an *effort distributor* or simply a node. The node defines how the adversary distributes her verification complexity (or guesswork) over different subkey parts. It is easy to see from the definition above that an effort distributor not only determines the full key success rate $\texttt{Prob}(\mathbf{e})$ that is achieved through guessing all candidates in the Cartesian product, but also determines the full key verification cost $\texttt{Cost}(\mathbf{e})$, or *guesswork*. They are expressed as

$$\texttt{Prob}\,(\mathbf{e}) \;=\; \prod_{i=1}^{b} \mathsf{SR}^{k_i}(e_i) = \prod_{i=1}^{b} \sum_{j=1}^{e_i} p_{i,[j]} \tag{6.3}$$

$$\texttt{Cost}\,(\mathbf{e}) \;=\; \prod_{i=1}^{b} e_i \tag{6.4}$$

In general, the adversary is interested in finding the minimal necessary guesswork to achieve a σ success rate for a full key recovery attack. The procedure of finding minimal full key recovery guesswork through finding optimal effort distributors is referred to as the *weak Maximum Likelihood* (wML) approach. Intuitively and informally, the observed leakage $\mathbf{l}^q$ reveals different amounts of secret information for different subkey parts. The more information is leaked of a certain key part, the more confidence the adversary gets for prioritized subkey guesses. Therefore, she can include more subkey candidates for the subkey positions where she has less confidence in the correctness of the output hypothesis (cf. e.g. [70]).

Formally, the wML approach can be stated as an optimization problem with the objective function and restriction condition defined as below.

$$\text{Objective:} \quad \textbf{Minimize } \texttt{Cost}(\mathbf{e}) \tag{6.5}$$

$$\text{Restriction Condition: } \texttt{Prob}(\mathbf{e}) \geq \sigma \tag{6.6}$$

We will show how to solve this optimization problem in Section 6.2.3.

There are differences between the wML and the true ML approaches. In ML, all full key candidates are ordered according to their posterior probability. In wML, this is not necessarily the case. In fact, full key candidates that are inside the Cartesian

product of selected subkey guesses are prior to combinations that are not defined by the effort distributor. For example, given an effort distributor $\mathbf{e} = (e_1, ..., e_b)$, the full key candidate $\mathbf{g}_x = g_{1,[e_1]} \| g_{2,[e_2]} \| g_{3,[e_3]} \| ... \| g_{b,[e_b]}$ is inside the Cartesian product, while the candidate $\mathbf{g}_y = g_{1,[e_1-1]} \| g_{2,[e_2+1]} \| g_{3,[e_3]} \| ... \| g_{b,[e_b]}$ is not. The former is to be considered by the wML approach while the latter is not. Therefore, wML sets priority of the former over the latter. However, it is not always the case that $\mathbf{g}_x$ is more probable than $\mathbf{g}_y$. This means using wML will unavoidably cause some ordering violation. The impact of such violation is discussed in Section 6.3.4 and it turns out that the penalty is rather low, which confirms the usability of wML approach.

6.2.2 The Search Domain and its Calculus Model

An optimization problem in the continuous domain can usually be turned into a searching problem. Tools from differential calculus such as the gradient vector can help providing efficient search directions. Here we adjust it to our search space which is a discretized domain and build the model for the problem of searching optimal effort distributors. All concepts introduced here will be used in the KSF algorithm in Section 6.2.3. For a clear illustration we use AES-128 as an example. It can be easily applied in other block cipher scenarios.

Structure of the Search Domain

We first define the search space. Each effort distributor $\mathbf{e}$ is treated as a node in the b-dimensional discrete space. For AES-128, the key has 16 subkey parts (bytes) and each effort entry—the number of guesses for each subkey part—can be any integer

between 1 and 256 inclusively. Therefore, the entire search space is 16 dimensional with each dimension taking integers in $[1 : 256]$, namely $\mathcal{E} = [1 : 256]^{16}$. The optimization problem is now equivalent to finding the optimum node $\mathbf{e}^* \in \mathcal{E}$ that minimizes the full cost or guesswork while achieving the required full key success probability. To better understand the structure of the search space and enable an efficient search, we introduce the following concepts.

Definition 1: a node $\mathbf{e}' = (e'_1, ..., e'_b)$ is called the j-th *decremental neighbor* of the node $\mathbf{e} = (e_1, ..., e_b)$ if $e'_j = e_j - 1$ and $e'_i = e_i$ for all $i \neq j$. It is also denoted as

$$\mathbf{e}_j^- = (e_1, ..., e_j - 1, ..., e_b).$$

Similarly, the j-th *incremental neighbor* of node $\mathbf{e}$ is denoted as

$$\mathbf{e}_j^+ = (e_1, ..., e_j + 1, ..., e_b).$$

Definition 2: a node $\mathbf{e} \in \mathcal{E}$ is said to be σ-*feasible* if it satisfies the restriction condition (6.6). The set of all σ-feasible nodes is denoted as

$$\mathcal{E}_\sigma := \{\mathbf{e} \mid \texttt{Prob}\,(\mathbf{e}) \geq \sigma\}.$$

Definition 3: a σ-feasible node $\mathbf{e} \in \mathcal{E}_\sigma$ is said to be on the boundary if none of its decremental neighbors is σ-feasible, i.e. $\mathbf{e}_j^- \notin \mathcal{E}_\sigma, \forall j$. The set of all nodes on the boundary is called the σ-*feasible boundary* and denoted as

$$\partial\,(\mathcal{E}_\sigma) := \left\{\mathbf{e} \in \mathcal{E}_\sigma \mid \mathbf{e}_j^- \notin \mathcal{E}_\sigma, \forall j\right\}$$

Definition 4: a node $\mathbf{e}^*$ is called σ-*optimal* if it is a σ-feasible node and has minimal complexity among all σ-feasible nodes, i.e. $\texttt{Cost}(\mathbf{e}^*) \leq \texttt{Cost}(\mathbf{e}), \forall \mathbf{e} \in \mathcal{E}_\sigma$

An immediate but important result can now be summarized as follows.

Boundary Property: the σ-optimal nodes are inside the σ-feasible boundary, i.e. $\mathbf{e}^* \in \partial(\mathcal{E}_\sigma) \subset \mathcal{E}_\sigma$.

The proof is straightforward. If $\mathbf{e}_j^{*-} \in \mathcal{E}_\sigma$, then

$$\texttt{Cost}\left(\mathbf{e}_j^{*-}\right) = \texttt{Cost}\left(\mathbf{e}^*\right) \cdot \frac{e_j^* - 1}{e_j^*} < \texttt{Cost}\left(\mathbf{e}^*\right)$$

contradicting the definition of node $\mathbf{e}^*$ being σ-optimal.

This property explains the fact that if making one less subkey guess at any subkey part from an optimal effort distributor, the achieved success rate does not reach the desired level σ. It indicates that the wML approach is to find an σ-optimal effort distributor from the σ-feasible boundary.

A Calculus Model for the Search Problem

Now we define some calculus tools for enabling an efficient search algorithm for finding the optimum node in the discrete search domain. For a function in continuous space, the partial derivative $\frac{\partial f}{\partial x_j}$ indicates the instantaneous change of the output of the function f caused by the change at the j-th coordinate x_j of the input. We define similar concepts for the objective function $\texttt{Cost}(\mathbf{e})$ and restriction condition $\texttt{Prob}(\mathbf{e})$.

The discrete nature of our search domain $[1 : 256]^{16}$ gives two situations: the change caused by unit *incrementing* or *decrementing* on each effort coordinate e_j.

More specifically, we define the incremental partial derivative of $\texttt{Prob}(\mathbf{e})$ with respect to e_j as

$$\nabla P_j^+ = \texttt{Prob}(\mathbf{e}_j^+) - \texttt{Prob}(\mathbf{e}) = [\frac{\mathsf{SR}^{k_j}(e_j+1) - \mathsf{SR}^{k_j}(e_j)}{\mathsf{SR}^{k_j}(e_j)}]\texttt{Prob}(\mathbf{e}) \tag{6.7}$$

Each ∇P_j^+ is a non-negative value[2] and it indicates the amount of additional success rate that could be achieved by incrementing effort by 1 at the j-th coordinate. Similarly, the decremental partial derivative of $Prob(\mathbf{e})$ is defined as

$$\nabla P_j^- = \texttt{Prob}(\mathbf{e}) - \texttt{Prob}(\mathbf{e}_j^-) = [\frac{\mathsf{SR}^{k_j}(e_j) - \mathsf{SR}^{k_j}(e_j-1)}{\mathsf{SR}^{k_j}(e_j-1)}]\texttt{Prob}(\mathbf{e}) \tag{6.8}$$

This is also a non-negative value and it tells the loss of full key success rate caused by decreasing effort by 1 at the j-th coordinate.

With the above defined partial derivatives, we can now obtain the *incremental gradient* $\nabla P^+ = (\nabla P_1^+, ...\nabla P_{16}^+)$ and the *decremental gradient* $\nabla P^- = (\nabla P_1^-, ...\nabla P_{16}^-)$ of the restriction condition $\texttt{Prob}(\mathbf{e})$. It is important to see that the coordinate for the largest partial derivative in the incremental (or decremental respectively) gradient vector tells the full key success rate is increased (or decreased resp.) mostly due to a unit effort increment (or decrement resp.).

The same concept is defined for the objective function $\texttt{Cost}(\mathbf{e})$. The gradient vectors in both incrementing and decrementing cases result in the same expression because

[2]The cases are considered separately if incrementing or decrementing is impossible, i.e. $e_j = 1$ or $e_j = 256$ for equations (6.7), (6.8) and (6.9).

$$\nabla C_j^+ = \texttt{Cost}(\mathbf{e}_j^+) - \texttt{Cost}(\mathbf{e}) = \prod_{i \neq j} e_i = \texttt{Cost}(\mathbf{e}) - \texttt{Cost}(\mathbf{e}_j^-) = \nabla C_j^- \tag{6.9}$$

For notational convenience, both ∇C_j^+ and ∇C_j^- are replaced by ∇C_j and the gradient of the full key complexity $\texttt{Cost}(\mathbf{e})$ becomes $\nabla C = (\nabla C_1, ...\nabla C_{16})$. Again, each coordinate is a non-negative value and it indicates the change in full key recovery complexity which is caused by incrementing/decrementing effort by 1 at the j-th entry of effort node $\mathbf{e}$.

Lastly, we consider the *direction vector* $\mathbf{u}$ which is the negation of the gradient $-\nabla C$ projected onto the hyper-surface that is perpendicular to the gradient ∇P.

$$\mathbf{u} = -\nabla C \text{ projected onto } (\nabla P)^\perp = \frac{\nabla P \cdot \nabla C}{\|\nabla P\|^2} \nabla P - \nabla C \tag{6.10}$$

where $\nabla P = (\nabla P_1, ..., \nabla P_{16})$ is the averaged gradient, i.e. $\nabla P_j = (\nabla P_j^+ + \nabla P_j^-)/2$. This direction vector $\mathbf{u}$ satisfies the intuition to keep the restriction condition $\texttt{Prob}(\mathbf{e})$ unchanged (seen from the vanishing of the inner product $\mathbf{u} \cdot \nabla P = 0$) while decreasing the objective function $\texttt{Cost}(\mathbf{e})$ as much as possible. A visualization can be seen in Figure 6.2.3.

6.2.3 An Optimized Key Space Finding Algorithm

We now show how to realize the weak maximum likelihood approach to find the optimum effort distributor by using the KSF algorithm.

The **inputs** of the algorithm include the desired full key success probability σ and the sorted posterior probabilities $p_{i,[j]}$ (and hence the subkey success rates

$\mathsf{SR}^{k_i}(t)$ according to equation (6.2)) for all subkey candidates $g_{i,[j]}$. The **inputs** of the algorithm include the desired full key success probability σ and the sorted posterior probabilities $p_{i,[j]}$ (and hence the subkey success rates $\mathsf{SR}^{k_i}(t)$ according to equation (6.2)) for all subkey candidates $g_{i,[j]}$. Note that this algorithm, unlike the VGS algorithm, does not require knowledge of the correct key, i.e. can also be used by a key recovering adversary. The applicability of this algorithm is not restricted to the profiling adversary. In [72], it is suggested that a non-profiling adversary can also assign likelihoods to subkey candidates to achieve a justified full key ranking, which could also be applied in our case.

There are two **outputs** returned by the algorithm: one is the minimum verification complexity $\min\{\texttt{Cost}(\mathbf{e}) \mid \mathbf{e} \in \mathcal{E}_\sigma\}$ that ensures the desired full key success rate σ ; the other one is an optimal effort distributor $\mathbf{e}^* = \text{argmin}\{\texttt{Cost}(\mathbf{e}) \mid \mathbf{e} \in \mathcal{E}_\sigma\}$ that achieves this complexity lower bound.

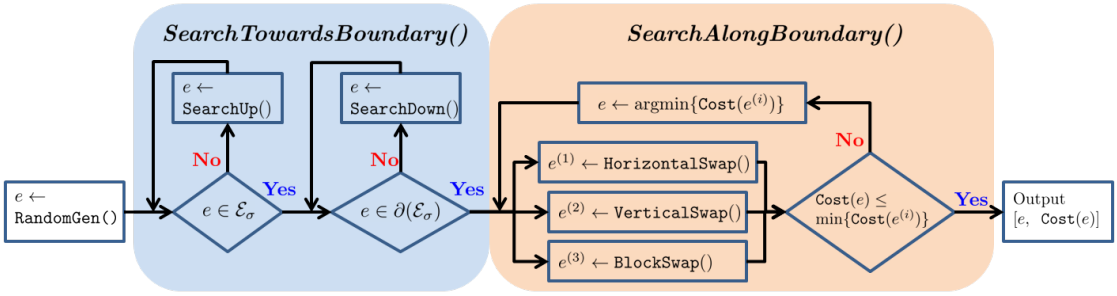


Figure 6.1: Flow Chart of the KSF algorithm.

The flow chart of the KSF algorithm is shown in Fig. 6.1. It uses several subroutines. The algorithm begins by generating a random node $\mathbf{e} \leftarrow [1:256]^{16}$ using `RandomGen`(). This node serves as the starting point in the searching space. The initial node is then passed sequentially into two subroutines: `SearchTowardsBoundary()` and `SearchAlongBoundary()`. The former moves a node onto the feasible bound-

Algorithm 6.1 SearchUp()

1: **while** $\texttt{Prob}(\mathbf{e}) < \sigma$ **do**
2: $\quad i \leftarrow \text{argmax}_j\{\nabla P_j^+\}$
3: $\quad e_i \leftarrow e_i + 1$
4: **end while**
5: **return** **e**

ary $\partial(\mathcal{E}_\sigma)$ by calling `SearchUp()` and `SearchDown()`. The latter searches for nodes within the boundary, which feature an even lower value of the objective $\texttt{Cost}(\mathbf{e})$. It uses the `Swap()` family of subfunctions. Note that the algorithm is a probabilistic algorithm to finding the point on the surface that has the minimal cost. It finds local minima. In practice, it is executed several times to ensure that the local optimization also yields the global minimum.

The `SearchTowardsBoundary()` Function

The task of this function is to move a node onto the feasible boundary $\partial(\mathcal{E}_\sigma)$. If the input node $\mathbf{e}$ does not satisfy the restriction condition, i.e. $\texttt{Prob}(\mathbf{e}) < \sigma$, it calls the function `SearchUp()` (as shown in Alg. 6.1) to search for a node that is σ-feasible. More specifically, `SearchUp()` iteratively increases the number of subkey guesses for some part of the subkey and updates the node. In each iteration, the search direction, i.e. the coordinate of the subkey part that needs to be incremented, is determined by the incremental gradient ∇P^+ as defined in Section 6.2.2. The effort coordinate that maximizes the gain in success rate through a unit effort increase is chosen, i.e. $i = \text{argmax}_j\{\nabla P_j^+\}$. The node is updated by a unit increment on the chosen effort coordinate. The process continues until a σ-feasible node is reached, namely, the restriction condition is satisfied as $\texttt{Prob}(\mathbf{e}) \geq \sigma$.

Algorithm 6.2 SearchDown()

1: **while** $\mathbf{e} \notin \partial(\mathcal{E}_\sigma)$ **do**
2: $\quad i \leftarrow \text{argmax}_j\{\nabla C_j \text{ s.t. } \texttt{Prob}(\mathbf{e}) - \nabla P_j^- \geq \sigma\}$
3: $\quad e_i \leftarrow e_i - 1$
4: **end while**
5: **return** $\mathbf{e}$

Now we have a σ-feasible node—either it is an initially generated node that already satisfies the restriction condition or it is a node returned from `SearchUp()`. The remaining task is to search for a node on the feasible boundary $\partial(\mathcal{E}_\sigma)$, since the optimal effort distributors can be found only on the boundary. The function `SearchDown()` is called to complete this task. In each iteration, the gradient vector ∇C of the objective function $\texttt{Cost}(\mathbf{e})$ is used to determine the search direction, i.e. the effort coordinate that needs to be decremented as shown in line 2 of Alg. 6.2. It reflects the direction where the objective function $\texttt{Cost}(\mathbf{e})$ has the biggest complexity drop through a unit effort decrementing while not violating the restriction condition. This means that the updated node is still σ-feasible. The process continues until the Boundary Property (as defined in Section 6.2.2) is satisfied. In other words, it returns a node $\mathbf{e} \in \partial(\mathcal{E}_\sigma)$.

The `SearchAlongBoundary()` Function

So far the search algorithm has found a node on the σ-feasible boundary. The next step is to search for nodes within the boundary, which achieve σ-feasibility at a lower cost $\texttt{Cost}(\mathbf{e})$. The subroutine `SearchAlongBoundary()` is called to accomplish this task. We have seen from the Boundary Property in Section 6.2.2 that any decremental neighbor of a node on the boundary is not σ-feasible. It implies that

the only way to find a node with lower full key cost is through trading-off (or swapping) efforts between different coordinates, which is realized in the `Swap()` family of subroutines.

More specifically, the coordinates for swapping are determined from the direction vector $\mathbf{u}$ defined in equation (6.10) as it follows the intuition that the search should decrease the overall guesswork while not compromising the full key success probability. The direction vector $\mathbf{u}$ suggests to increase effort on coordinate j if u_j is positive, and decrease if negative. The order of the effort coordinates being incremented or decremented is determined by the order of the absolute values of the entries u_j. The higher the absolute value, the higher the priority that is assigned to the coordinates for incrementing and decrementing.

Similar to search problems defined in continuous domain, the algorithm also handles the problem of local minima that prevents effective searching. In particular, we implement three different swapping modes –`HorizontalSwap`(), `VerticalSwap`() and `BlockSwap`() – to "escape" from many local minima and therefore mitigate the risk of being terminated in advance. The `HorizontalSwap()` allows trading-off multiple efforts between the positive most and negative most coordinates, i.e. u_i^+ and u_j^-. The `VerticalSwap()` in each iteration enables trading-off one effort between multiple coordinates where u_js are of different signs. Finally, the `BlockSwap()` mode enables trading-off multiple efforts on multiple coordinates. All three modes ensure that the swap does not compromise the required full key success probability, i.e. $\mathbf{e} \in \mathcal{E}_\sigma$ always hold. The updated node (after efforts being swapped) is again passed through `SearchDown()` to ensure that the search is still performed on the boundary. The three modes prevent infinite loops because the swap action occurs only if the

cost of the updated node is lower than the cost for the session node.

As shown in Alg. 6.3, a temporary node $\mathbf{e}'$ is returned from the `Swap()` family of functions in each iteration. If the cost for the temporary node is lower than the current session node, then the session node $\mathbf{e}$ is replaced by before being passed into the next iteration. Otherwise, the search is terminated and the algorithm outputs the current node $\mathbf{e}$ and its full key verification costs $\texttt{Cost}(\mathbf{e})$.

Algorithm 6.3 SearchAlongBoundary()

1: $\mathbf{e}' \leftarrow \texttt{Swap}()$
2: **while** $\texttt{Cost}(\mathbf{e}) > \texttt{Cost}(\mathbf{e}')$ **do**
3: $\quad \mathbf{e} \leftarrow \mathbf{e}'$
4: **end while**
5: **return** $[\mathbf{e}, \texttt{Cost}(\mathbf{e})]$

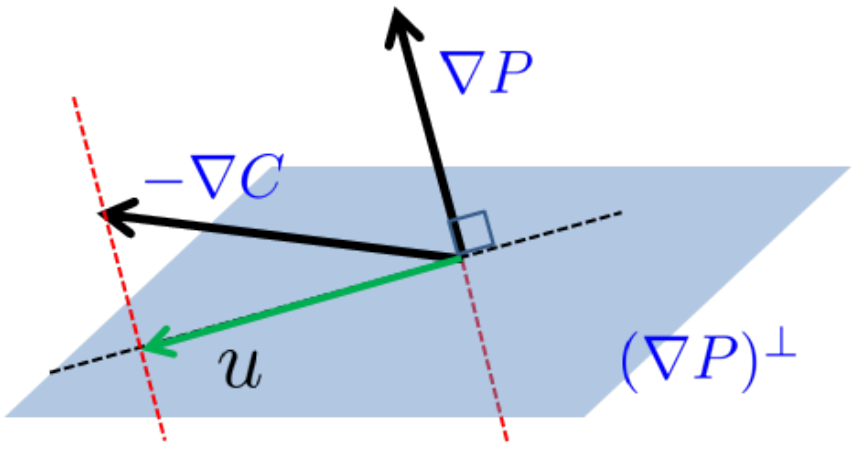


Figure 6.2: The direction vector $\mathbf{u}$ is the projection of cost gradient $-\nabla C$ onto $(\nabla P)^{\perp}$

6.2.4 Usage of the KSF algorithm

Full key security evaluation used to stay as an analysis that is beyond computing power. The KSF algorithm provides practical meaning to the security evaluation. Firstly, the adversary can use it to determine if the leakage is strong enough to enable full key recovery at her accessible computing power. More specifically, upon a particular set of observations $(\mathbf{x}^q, \mathbf{l}^q)$, the returned global minimum of $\texttt{Cost}(\mathbf{e})$ serves as an *individual* lower bound of the optimum guesswork w_σ. If the guesswork is acceptable, the associated optimal effort distributor $\mathbf{e}$ provides a winning strategy: checking all the full key candidates defined by the Cartesian product of this optimal node. This strategy ensures the adversary with success rate being at least σ. Even if in one session the observed leakages are not strong enough, namely requires high w_σ, she can just wait for the next session until a "good" observation appears. This can be the case if the guesswork is impacted a lot from different observations, which is in fact verified in our experiments in the next section.

Secondly, it can be used by a security evaluation lab. By feeding the algorithm with independently generated observations $(\mathbf{x}^q, \mathbf{l}^q)$, an evaluator can bootstrap the individual lower bounds and obtain the distribution of the guesswork w_σ at any fixed σ. This informs the evaluator the resistance of some DUT against a probabilistic SCA. In other words, if the adversary intends σ success rate, how much chance does she have by waiting until a strong enough leakage occurs. A simple example would be computing the *expected* lower bound of guesswork—the average of all individual lower bounds—and using it as a metric. The metric indicates the averaged level of security of the full key as the expectation is with respect to various experiments, i.e. not only different choices of input $\mathbf{x}^q$, but also leakages observations $\mathbf{l}^q$.

6.3 Experiment Results and Comparison

In this section we apply the proposed wML approach to practical side channel leakage evaluation. We first explain the experimental setup. Next, we verify the validity of the KSF algorithm and discuss its possible influencing factors. Finally, we compare our approach and VGS algorithm. .

6.3.1 Experiment Setup

We conduct the leakage evaluation experiments in two settings: real measurements and simulations. For the former, we target on an unprotected AES software implementation, the RjindaelFurious [53] running on an 8-bit AVR ATXMega A3B processor. A total of 200,000 measurements were taken using a Tektronix DPO 5104 oscilloscope at a sampling rate of 200MS/s. Among all the collected traces, 20,000 are used for building Gaussian templates. The remaining traces are used as needed for the evaluation step. In the other setting, we simulate side channel leakage using the widely accepted Hamming weight leakage model with additive Gaussian noise. In both cases the targeted leakage is that of the s-box output of the first round for each of the 16 state bytes.

6.3.2 Posterior Probabilities Derivation

As a preparation step of leakage evaluation, posterior probabilities for all subkey candidates need to be estimated from side channel observations. The probably most popular method is through Templates [11, 38] where the adversary creates a precise model of the leakage in the profiling phase and derives posterior probabilities in the

attack phase. An in-depth discussion of modeling errors for Gaussian templates can be found in [20]. For our experiments, we build Gaussian templates $\mathcal{N}(L; \mu_v, \Sigma_v^2)$ regarding the output v of Sboxes at each of the 16 bytes. In the attack phase, the adversary obtains the observations $(\mathbf{x}^q, \mathbf{l}^q)$. Since the predicted internal state for the j-th query is $y_{i,j,g} = f(x_{i,j}, g)$ under the subkey hypothesis g at the i-th subkey part, the observed leakage $l_{i,j}$ has conditional probability density $\mathrm{P}[l_{i,j} \mid g] = \mathcal{N}(l_{i,j}; \mu_v, \Sigma_v^2)$, where $v = y_{i,j,g}$. Since side channel leakages in different queries are independent, the conditional probability density $\mathrm{P}[\mathbf{l}_i^q \mid g]$ of observing the q leakages $\mathbf{l}_i^q = (l_{i,1}, ..., l_{i,q})$ on the i-th subkey part is the product of each $\mathrm{P}[l_{i,j} \mid g]$. Namely,

$$\mathrm{P}[l_{i,1}, ..., l_{i,q} \mid g] = \prod_{j=1}^{q} \mathrm{P}[l_{i,j} \mid g] = \prod_{j=1}^{q} \mathcal{N}(l_{i,j}; \mu_v, \Sigma_v^2). \tag{6.11}$$

Further, the Bayesian formula returns posterior probabilities $p_{i,g} := \Pr[g \mid \mathbf{l}_i^q]$ of subkey hypothesis g given the q observations $\mathbf{l}_i^q$ as

$$p_{i,g} := \Pr[g \mid \mathbf{l}_i^q] = \frac{\mathrm{P}[\mathbf{l}_i^q \mid g] \cdot \Pr[g]}{\sum_{g^*} \mathrm{P}[\mathbf{l}_i^q \mid g^*] \cdot \Pr[g^*]} = \frac{\mathrm{P}[\mathbf{l}_i^q \mid g]}{\sum_{g^*} \mathrm{P}[\mathbf{l}_i^q \mid g^*]} \tag{6.12}$$

Finally, the posterior probabilities $p_{i,g}$ are sorted into a descending sequence $p_{i,[g]}$ as detailed in Section 6.1.2. They determine the subkey success rates in equation (6.2) which are the inputs for the KSF algorithm and the VGS algorithm.

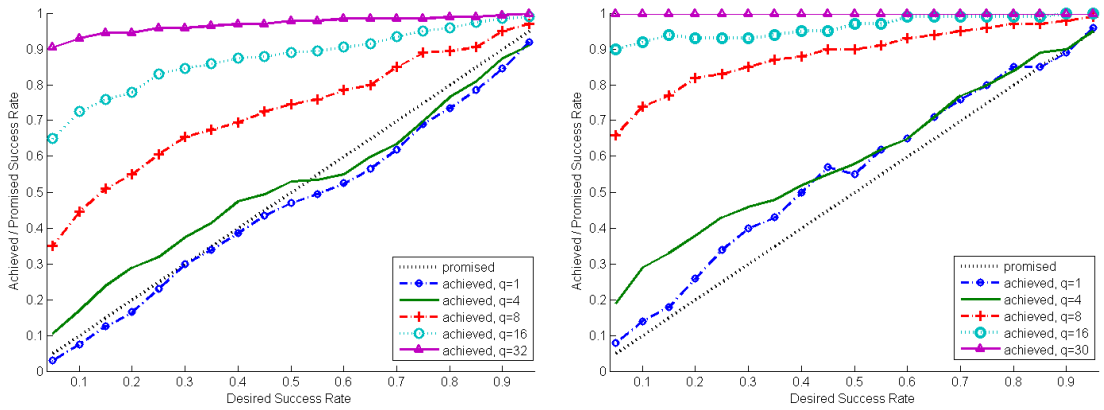
6.3.3 Correctness and Influencing Factors

Verifying the correctness of the KSF algorithm is rather simple: if the returned optimal effort distributor $\mathbf{e}^*$ covers the ranks of the posterior probability of every subkey k_i, then the search space defined by the Cartesian product includes the

correct full key as explained in Section 6.2.1. In the following, we check if the algorithm in fact achieves the promised success rate for various experiments. We provide a set of observations for a range of q from 1 to 40: higher value for q indicates more leaked information. We furthermore set 19 different levels of desired success rate from 0.05 to 0.95 incrementing at 0.05. For each possible (q, σ), 200 experiments are performed for the scenario using real measurements, and 100 experiments for the scenario using simulated leakage.

Figure 6.3(a) compares the promised full key success rate of the KSF algorithm with the actually achieved success rate for real measurements. One can see that when the leakage is strong (high value of q), the achieved success rate is far beyond what is promised. However, when the leakage is weak, the two rates only differ slightly. A probable reason for the achieved success rate being *lower* than the desired success rate for small values of q lies in the assumption that the Gaussian templates fully capture the underlying leakage distribution. In fact, the empirically obtained Gaussian templates only serve as approximation to the true leakage distribution, and hence the derived posterior probabilities are unavoidably biased. This claim is also supported by the results for simulated leakage, as given in Figure 6.3(b), where the underachieving never happens. Nevertheless, for almost all cases, especially when $q \geq 8$, the KSF algorithm fulfills the promised full key success rate.

Other influencing factors of the KSF algorithm are the leakage observations and the number of independent initial nodes used for finding local minima, as discussed in Section 6.2. To investigate their impact, we run 50 experiments associated with independent sets of observations $(\mathbf{x}^q, \mathbf{l}^q)$. In each experiment, we compare the performance of KSF algorithm at the fixed $\sigma = 50\%$ using 100 and 10000 initial nodes.



(a) (b)

Figure 6.3: Correctness verification for real measurements (a) and simulation (b); The success rate that KSF achieves (y-axis) is more than what it promised (x-axis).

The global minimum guessworks in each experiment are returned and compared in Figure 6.4(a). The x-axis is the index of experiments indicating a different set of observation $(\mathbf{x}^q, \mathbf{l}^q)$ and the y-axis is the guesswork in bits. As we can see, different leakage observations cause more than 40 bits guesswork differences while the influence from the number of initial nodes (the distance between the two curves) is rather small. In fact, the biggest difference between the two curves is less than 2.5 bits and most of the times the difference is smaller than one bit.

6.3.4 Comparing KSF algorithm with VGS algorithm

As mentioned in Section 6.1.1, the VGS algorithm estimates the rank of the correct key among all full key candidates. By bootstrapping this rank statistic, or namely, by repeating the rank estimation from different side channel observations, one can get a security evaluation based on the success percentiles to see the rank distributions given random side channel inputs.

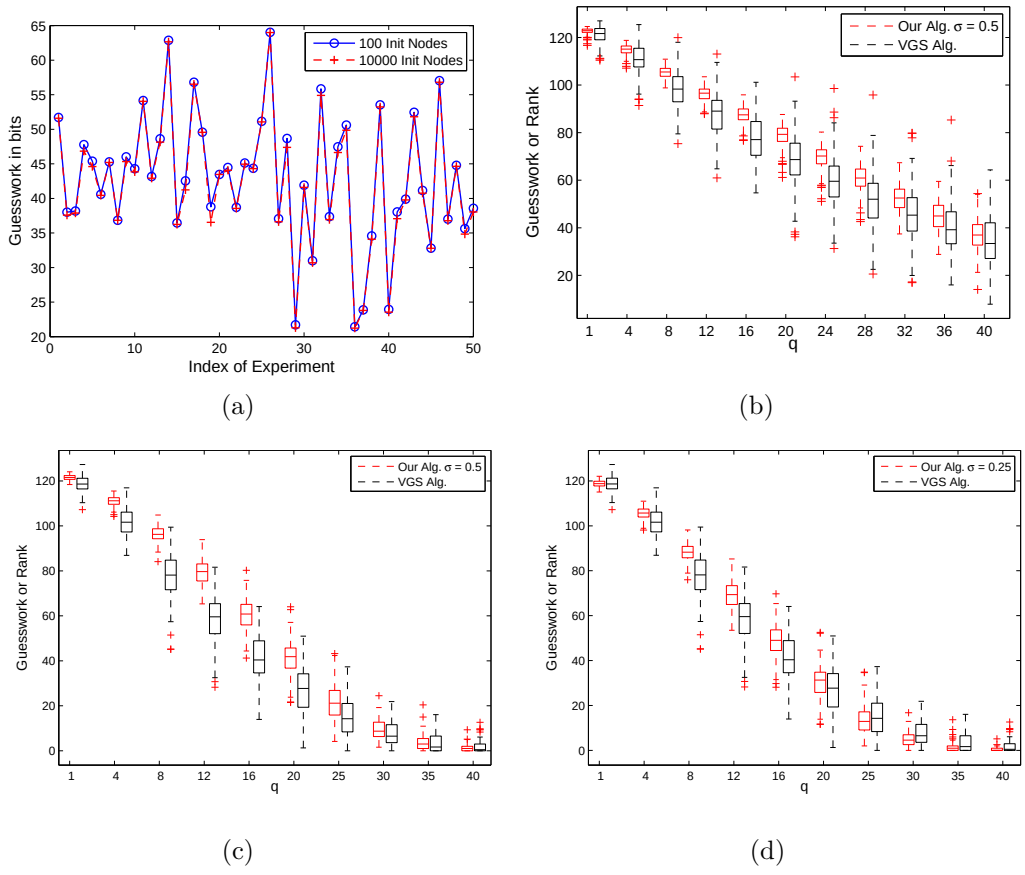
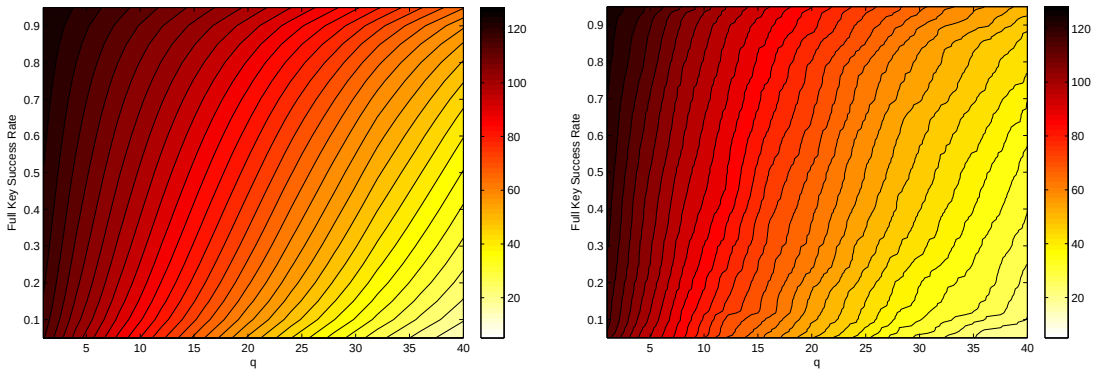


Figure 6.4: Figure (a) shows the impact on guesswork (y-axis) from the number of starting nodes for KSF algorithm is far less than the impact from the set of observations $(\mathbf{x}^q, \mathbf{l}^q)$ in each experiment (X-axis); Figure (b,c,d) compares the size of the key space from the KSF algorithm to the key rank from the VGS algorithmExperiments are performed over real measurement with success rate $\sigma = 50\%$ (b); over simulation with $\sigma = 50\%$ (c); and over simulation with $\sigma = 25\%$ (d)

We first provide several comparisons between the bootstrapping of the `rank` statistic from repeating VGS algorithm and the bootstrapping of guesswork w_σ KSF algorithm. Figure 6.4(b) compares the two over the real measurement. We fix the full key success rate in KSF algorithm to $\sigma = 50\%$. For each q (x-axis), we perform 200 experiments using the algorithms on the same sets of observations. The box plot indicates quartiles and outliers of the guesswork and rank statistics.

We see that the results from the two algorithms are relatively close to each other. Further, the impact of different leakages on the rank statistic using VGS algorithm is heavier than that on the guesswork returned from our algorithm. This can be seen from the difference of the height of boxes for the two algorithms. More importantly, we see that the medians of the two analyzed cases do not align exactly. In fact, ours are always slightly higher than the VGS algorithm. The reason is two folds. On one side, the KSF algorithm is following wML approach, which introduces ordering violation comparing to the true ML approach, as explained in Section 6.2. On the other, since in each individual experiment the VGS algorithm does not return a fixed success probability $\sum_{t=1}^{\texttt{rank}} p_{[t]}$ (the ML adversary should guesses all the top `rank` full key candidates), the 50th percentile of the `rank` does not necessarily ensure the adversary achieves 50% success rate in an averaged experiment either. This is even more clearly seen from the simulated leakage scenario as shown in Figure 6.4(c) (`rank` compared to $w_{0.5}$) and 6.4(d) (`rank` compared to $w_{0.25}$). In the simulated case, the ML approach is closer to the $w_{0.25}$ bootstrapping with the weak ML approach. It indicates that the guessing the top `rank` most likely full key candidates in the ML approach roughly returns winning probability of 25%. In general, it might suggest the evaluator to find the appropriate σ level such that the bootstrapping of the guesswork w_σ matches the bootstrapping of the key `rank`. By doing so, the evaluator can estimate the success rate $\sum_{t=1}^{\texttt{rank}} p_{[t]}$ in an average experiment that the top `rank` full key candidates contain.

The next comparison of the two leakage evaluation algorithms is between the *expected* guesswork lower bound (Figure 6.5(a)) and the bootstrapping of the `rank` (Figure 6.5(b)). Experiments use the data from the microcontroller measurements.



(a) (b)

Figure 6.5: Security evaluation using KSF algorithm showing the remaining guesswork (color in (a)) and using VGS algorithm showing the key rank (color in (b)) over the number of observations q (x-axis) and success rate/percentile (y-axis).

The x-axis for both represents the number q of accessible leakages in each experiment. In Figure 6.5(a), the y-axis is the desired full key success probability σ. The color or gray-scale for the pixel at coordinate $(x, y) = (q, \sigma)$ represents the *expected* lower bound (as explained in Section 6.2.4) of the guesswork in log scale. The darker a pixel is, the more guesswork is needed to achieve the specified success rate σ. In particular, the expected lower bound at each (q, σ) is derived from 200 independent experiments. Each experiment uses an independent set of observations $(\mathbf{x}^q, \mathbf{l}^q)$ which yields different posterior probabilities $p_{i,[g]}$ computed as described in Section 6.3.2. The number of initial node is set to 100 (Figure 6.4(a) already shows this number is sufficient). The global minimum guesswork from the 100 searches is returned as the *individual* lower bound of the guesswork for this single experiment. Upon completion of the 200 experiments, the average of the 200 individual lower bounds yields the *expected* lower bound as reflected in the color of pixel in Figure 6.5(a). In short, the color at pixel (q, σ) indicates the expected minimum guesswork that a q-limited adversary should spend in order to achieve full key recovery with probability σ. In

Figure 6.5(b), VGS algorithm is executed with the same sets of observations $(\mathbf{x}^q, \mathbf{l}^q)$. The returned 200 `ranks` (represented in the color of each pixel) derive the statistical bootstrapping of the success percentile (the same as in bootstrapping) which is represented on the y-axis. Two contour plots are fairly close to each other.

6.4 Conclusion

The presented algorithm finds the optimal key search space that allows the adversary to achieve a predefined probability of success. Unlike prior work, the algorithm provides a connection between the remaining full key security and the success probability even for a single set of side channel observations. It furthermore is a constructive algorithm, since it not only bounds the remaining key search space, but also provides an optimized yet simple strategy to search that space. As a consequence, the algorithm can be used by embedded security evaluators to quantify the resistance of a device to SCA. It can also be used by an adversary to determine whether the leakage suffices for a successful key recovery attack.

Chapter 7

Conclusion & Future Directions

As unsolved issues for embedded cryptographic solutions, side channel attacks have been challenging the physical security of embedded crypto-systems. They mitigate or even invalidate the protection from cryptographic algorithms used in a system. Consequently, the desired security objectives such as confidentiality, integrity and authenticity risk to be compromised. To address the physical threats, a comprehensive side channel security evaluation is demanded. It requires a deep understanding of the mechanism that a side channel adversary detects and exploits the leaked information. It also needs investigation on the maximum capability that the adversary converts the leaked information to the targeted secrets.

In this dissertation, we show that SCA has not reached its maximum potential and hence the threat is unfortunately underestimated. By analyzing side channel leakages from the perspectives of leakage detection, exploitation and quantification, we show that SCAs can be more powerful in the following three aspects.

1. SCA can be mounted more reliably using the wide collision detection methods and more efficiently with the leakage bundling technique. They can detect

both univariate and multivariate leakage. Therefore, they are applicable for both protected and unprotected crypto implementations.

2. SCA can succeed without relying on leakage modeling. This is achieved from the approach of *observation-to-observation* comparison, which is instantiated with collision generation techniques. It advances classical SCAs as it no longer suffers from biased or wrong leakage models. It also advances side channel collision attacks by mitigating the risk of false positives in collision detection.

3. Probabilistic success can be achieved with even very limited number of side channel information. The constructive strategy is provided by the weak maximum likelihood approach. It allows the adversary, even prior to a real attack, to determine whether the smart key search is at an acceptable computational complexities and can provide desirable success rate.

All these data analytic techniques produce more powerful SCAs. Preventing these attacks calls for better countermeasures, more advanced testing methods as well as more comprehensive security evaluations. We consider the following areas to be further investigated.

1. Side channel testing is to examine the existence of side channel leakage. Current techniques perform well for testing univariate leakage. For instance, the Welch t test performs pointwisely; mutual information based tests can handle the estimation of univariate leakage distribution relatively easily. However, if the leakage is in multivariate nature, then joint distribution of leakage needs to be considered. The Welch t test is no longer a qualified candidate. Applying mutual information based tests will confront significant decrease of the pre-

cision in the joint distribution estimation. Therefore, advanced side channel testing methods need to be studied to be capable of examining multivariate side channel leakage.

2. The unified framework in [66] presents an information theoretic metric and a security metric for evaluating side channel security. The entropy analysis has been widely employed in the theoretical leakage resilience cryptography. The guesswork analysis in this work provides a tool for practical quantification of the security loss in terms of full key recovery. The gap between the two evaluation metrics is narrowed but not completely bridged. One interesting question is whether the two metrics are equivalent in terms of preventing SCAs. In other words, a joint study needs to exam whether low entropy loss implies low level of security loss or vice versa. The question is important because if the causal relation does not hold, (for instance if there is leakage distribution that results in low key-entropy loss but high security loss), then it implies the current leakage resilient designs may not guarantee resisting SCAs.

Bibliography

[1] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side Channel (s). In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pages 29–45. Springer, 2003. 28

[2] D. Agrawal, J. R. Rao, and P. Rohatgi. Multi-Channel Attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2003*, pages 2–16. Springer, 2003. 28

[3] J. Alwen, Y. Dodis, and D. Wichs. Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. *Advances in Cryptology – CRYPTO 2009*, pages 36–54, 2009. 40

[4] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater. Template Attacks in Principal Subspaces. In *Cryptographic Hardware and Embedded Systems CHES 2006,LNCS volume4249*, pages 1–14. Springer, 2006. 56

[5] 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. 43, 44, 94

[6] A. Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In *Cryptographic Hardware and Embedded Systems CHES 2008*. Springer Berlin / Heidelberg, 2008. 94

[7] A. Bogdanov, T. Eisenbarth, C. Paar, and M. Wienecke. Differential Cache-Collision Timing Attacks on AES with Applications to Embedded CPUs. In J. Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 235–251. Springer Berlin / Heidelberg, 2010. 42, 45, 47

[8] A. Bogdanov and I. Kizhvatov. Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. *Computers, IEEE Transactions on*, PP(99):1, 2011. 54, 94

[9] A. Bogdanov, I. Kizhvatov, and A. Pyshkin. Algebraic Methods in Side-Channel Collision Attacks and Practical Collision Detection. In D. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 251–265. Springer Berlin / Heidelberg, 2008. 54

[10] 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. 33

[11] S. Chari, J. Rao, and P. Rohatgi. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*. Springer Berlin / Heidelberg, 2003. 34, 151

[12] K. Chatzikokolakis, T. Chothia, and A. Guha. Statistical Measurement of Information Leakage. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 390–404. Springer, 2010. 36

[13] T. Chothia and A. Guha. A Statistical Test for Information Leaks using Continuous Mutual Information. In *Computer Security Foundations Symposium (CSF), 2011 IEEE 24th*, pages 177–190. IEEE, 2011. 36, 75

[14] J.-S. Coron and L. Goubin. On Boolean and Arithmetic Masking against Differential Power Analysis. In *Cryptographic Hardware and Embedded Systems CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000. 38

[15] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012. 22

[16] M. H. DeGroot, M. J. Schervish, X. Fang, L. Lu, and D. Li. *Probability and Statistics*, volume 2. Addison-Wesley Reading, MA, 1986. 19

[17] A. A. Ding, L. Zhang, Y. Fei, and P. Luo. A statistical model for higher order dpa on masked devices. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 147–169. Springer, 2014. 67

[18] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert. Univariate Side Channel Attacks and Leakage Modeling. *Journal of Cryptographic Engineering*, 1:123–144, 2011. 114

[19] F. Durvaux, F.-X. Standaert, and N. Veyrat-Charvillon. How to Certify the Leakage of a Chip? In *Advances in Cryptology–EUROCRYPT 2014*, pages 459–476. Springer, 2014. 85

[20] F. Durvaux, F.-X. Standaert, and N. Veyrat-Charvillon. How to certify the leakage of a chip? In *to appear in the proceedings of Eurocrypt 2014*. Springer LNCS, 2014. 152

[21] S. Dziembowski and K. Pietrzak. Leakage-Resilient Cryptography. In *IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, 2008. FOCS'08*, pages 293–302, 2008. 40

[22] S. Faust, K. Pietrzak, and J. Schipper. Practical Leakage-Resilient Symmetric Cryptography. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer Berlin Heidelberg, 2012. 40, 133

[23] C. Flynn and Z. Chen. Power Analysis Using Low- Power Analysis Using Low-Cost Hardware: Lab Setup & Simple Targets. CHES 2013 Tutorial, 2013. 29

[24] G. Fumaroli, A. Martinelli, E. Prouff, and M. Rivain. Affine Masking Against Higher-Order Side Channel Analysis. In *Selected Areas in Cryptography*, pages 262–280. Springer, 2011. 38

[25] B. Gierlichs, L. Batina, B. Preneel, and I. Verbauwhede. Revisiting Higher-Order DPA Attacks. In *Topics in Cryptology-CT-RSA 2010*, pages 221–234. Springer, 2010. 67, 68

[26] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. *Cryptographic Hardware and Embedded Systems–CHES 2008*, pages 426–442, 2008. 35

[27] B. Gierlichs, K. Lemke-Rust, and C. Paar. Templates vs. stochastic methods. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 15–29. Springer Berlin Heidelberg, 2006. 134

[28] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi, et al. A Testing Methodology for Side-Channel Resistance Validation. In *NIST Non-invasive attack testing workshop*, 2011. 66, 67, 75

[29] J. Golic and C. Tymen. Multiplicative Masking and Power Analysis of AES. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003. 38

[30] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007. 27

[31] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology CRYPTO 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 789–789. Springer Berlin / Heidelberg, 1999. 28, 32

[32] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. I. Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer Verlag, 1996. 28, 31

[33] P. C. Kocher. Leak-resistant cryptographic indexed key update (US patent 6539092), 2003. 133

[34] K. Lemke-Rust and C. Paar. Analyzing Side Channel Leakage of Masked Implementations with Stochastic Methods. In J. Biskup and J. Lopez, editors, *Computer Security ESORICS 2007*, volume 4734 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007. 114

[35] K. Lemke-Rust and C. Paar. Gaussian Mixture Models for Higher-Order Side Channel Analysis. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007. 125

[36] F. Mace, F.-X. Standaert, and J.-J. Quisquater. *Information theoretic evaluation of side-channel resistant logic styles*. Springer, 2007. 134

[37] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 16. Elsevier, 1977. 11

[38] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smartcards*. Springer-Verlag, 2007. 56, 133, 151

[39] L. Mather, E. Oswald, J. Bandenburg, and M. Wójcik. Does my device leak information? an a priori statistical power analysis of leakage detection tests. In *Advances in Cryptology-ASIACRYPT 2013*, pages 486–505. Springer, 2013. 75, 85

[40] M. Medwed, F.-X. Standaert, and A. Joux. Towards super-exponential side-channel security with efficient leakage-resilient prfs. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems — CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 193–212. Springer Berlin Heidelberg, 2012. 133

[41] T. S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In *Cryptographic Hardware and Embedded Systems CHES 2000*, pages 238–251. Springer, 2000. 67

[42] S. Micali and L. Reyzin. Physically Observable Cryptography. *Theory of Cryptography*, pages 278–296, 2004. 39

[43] A. Moradi. Statistical Tools Flavor Side-Channel Collision Attacks. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 428–445. Springer Berlin / Heidelberg, 2012. 93

[44] A. Moradi, M. Kasper, and C. Paar. Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures. In O. Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2012. 133

[45] 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. 87, 96, 116

[46] M. Nassar, S. Guilley, and J.-L. Danger. Formal Analysis of the Entropy / Security Trade-off in First-Order Masking Countermeasures against Side-Channel Attacks. In *Progress in Cryptology INDOCRYPT 2011*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011. 108, 109, 117

[47] M. Nassar, Y. Souissi, S. Guilley, and J.-L. Danger. RSM: A small and fast Countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012. 108, 109, 117, 124

[48] E. Oswald and S. Mangard. Template Attacks on Masking – Resistance Is Futile. In M. Abe, editor, *Topics in Cryptology CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 243–256. Springer Berlin Heidelberg, 2006. 114

[49] C. Paar and J. Pelzl. *Understanding Cryptography: a textbook for students and practitioners*. Springer, 2009. 27

[50] K. Pietrzak. Provable Security for Physical Cryptography. *the proceedings of WEWORC*, 2009. 40

[51] J. Pliam. The disparity between work and entropy in cryptology. Cryptology ePrint Archive, Report 1998/024, 1998. `http://eprint.iacr.org/`. 136

[52] J. Pliam. On the Incomparability of Entropy and Marginal Guesswork in Brute-Force Attacks. In B. Roy and E. Okamoto, editors, *Progress in Cryptology — INDOCRYPT 2000*, volume 1977 of *Lecture Notes in Computer Science*, pages 67–79. Springer Berlin Heidelberg, 2000. 136

[53] B. Poettering. Rijndael Furious. Implementation. `http://point-at-infinity.org/avraes/`. 77, 97, 151

[54] E. Prouff and M. Rivain. A Generic Method for Secure SBox Implementation. In S. Kim, M. Yung, and H.-W. Lee, editors, *Information Security Applications*, volume 4867 of *Lecture Notes in Computer Science*, pages 227–244. Springer Berlin Heidelberg, 2007. 108

[55] E. Prouff and M. Rivain. Theoretical and Practical Aspects of Mutual Information-based Side Channel Analysis. *International Journal of Applied Cryptography*, 2(2):121–138, 2010. 36

[56] E. Prouff, M. Rivain, and R. Bévan. Statistical analysis of second order differential power analysis. *Computers, IEEE Transactions on*, 58(6):799–811, 2009. 67

[57] Pub, NIST FIPS. 197: Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication*, 197:441–0311, 2001. 14

[58] F. Regazzoni, S. Badel, T. Eisenbarth, J. Großschädl, A. Poschmann, Z. T. Deniz, M. Macchetti, L. Pozzi, C. Paar, Y. Leblebici, and P. Ienne. A simulation-based methodology for evaluating the DPA-resistance of cryptographic functional units with application to CMOS and MCML technologies. In *International Symposium on Systems, Architectures, Modeling and Simulation (SAMOS VII)*, 2007. 134

[59] 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. 85

[60] O. Reparaz, B. Gierlichs, and I. Verbauwhede. Selecting Time Samples for Multivariate DPA Attacks. In *Cryptographic Hardware and Embedded Systems–CHES 2012*, pages 155–174. Springer, 2012. 42, 68, 76

[61] M. Rivain. On the exact success rate of side channel analysis in the gaussian model. In R. Avanzi, L. Keliher, and F. Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 165–183. Springer Berlin Heidelberg, 2009. 133

[62] J. J. Rotman. *Advanced Modern Algebra*. American Mathematical Soc., 2002. 8

[63] 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*. Springer Berlin Heidelberg, 2005. 35, 86, 114

[64] 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. 43, 44, 94, 116

[65] 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. 94

[66] 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. 103, 106, 127, 133, 135, 161

[67] F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. Leakage Resilient Cryptography in Practice. In A.-R. Sadeghi and D. Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 99–134. Springer Berlin Heidelberg, 2010. 40, 133

[68] TELECOM ParisTech SEN research group. Dpa contest (versions 1 and 2). `http://www.dpacontest.org/home/`. 133

[69] TELECOM ParisTech SEN research group. The DPA Contest V4, 2013-2014. `http://www.dpacontest.org/v4`. 77, 124

[70] A. Thillard, E. Prouff, and T. Roche. Success through confidence: Evaluating the effectiveness of a side-channel attack. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*, pages 21–36. Springer Berlin Heidelberg, 2013. 139

[71] K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European*, pages 403 –406, sept. 2002. 133

[72] N. Veyrat-Charvillon, B. Gérard, M. Renauld, and F.-X. Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer Berlin Heidelberg, 2013. 134, 145

[73] 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. 96, 134

[74] N. Veyrat-Charvillon and F.-X. Standaert. Mutual Information Analysis: How, When and Why? In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 429–443. Springer Berlin Heidelberg, 2009. 36, 112

[75] C. Whitnall and E. Oswald. A Fair Evaluation Framework for Comparing Side-Channel Distinguishers. *Journal of Cryptographic Engineering*, 1(2):145–160, 2011. 102

[76] 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. 112

[77] C. Whitnall, E. Oswald, and F.-X. Standaert. The Myth of Generic DPA and the Magic of Learning. In *Topics in Cryptology–CT-RSA 2014*, pages 183–205. Springer, 2014. 85

[78] X. Ye, C. Chen, and T. Eisenbarth. Non-Linear Collision Analysis. In *RFID Security*. Springer Berlin Heidelberg, 2014. 5

[79] 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. 5, 94

[80] X. Ye and T. Eisenbarth. On the Vulnerability of Low Entropy Masking Schemes. In *CARDIS 2013*. CARDIS, 2013. 5

[81] X. Ye, T. Eisenbarth, and W. Martin. Bounded, yet Sufficient? How to Determine Whether Limited Side Channel Information Enables Key Recovery. In *to appear in CARDIS 2014*, 2014. 5