Mutual information analysis for side channel attacks

Timmerman, T.R.

Award date:
2011

Eindhoven University of Technology
Department of Mathematics and Computer Science

MASTER'S THESIS

Mutual Information Analysis for Side Channel Attacks

Thijs R. Timmerman

Supervisors:
Dr.ir. L.A.M. (Berry) Schoenmakers (TU/e)
J. (Jing) Pan MSc (Riscure)

October 2011

Acknowledgements

This thesis is the result of my graduate internship at Riscure BV, Delft. Moreover, seven wonderful years of study at Eindhoven University of Technology are concluded by this work. Numerous people have contributed in various ways to help me completing my magnum opus, for which I would like to express my deep gratitude.

At Riscure BV, I was given the opportunity to perform an innovative and challenging master's thesis project. I am grateful to my daily supervisor, Jing Pan, who weekly provided me with valuable and useful advice. Besides that, Jing patiently explained many concepts in side channel analysis and symmetric cryptography, giving me a clear understanding of this matter. Furthermore, I would like to thank my colleagues at Riscure for the nice time.

At Eindhoven University of Technology, my supervisor Berry Schoenmakers gave useful feedback and helped me to maintain a mathematical approach to this project. Berry provided me with critical questions and kept a fresh view on my work, for which I am thankful. I also would like to thank Benne de Weger and Jerry den Hartog for participating in my graduation committee.

The time in Eindhoven would not have been as exciting without the presence of many persons. Gentlemen of "In Vino Veritas", Batsers, fellow bartenders: merci beaucoup for your vivid participation in numerous discussions, speeches, borrels and weekends abroad. It were very good years.

My parents have offered me unlimited care, love and support. Furthermore, our many sailing trips provided me with appropriate leisure opportunities. Thank you for every warm welcome home, for your dedication and for facilitating this inspirational time. Inez, thank you for your love and for knowing exactly how to handle me.

Thijs Timmerman
October 2011

Abstract

Mutual information analysis is a recently proposed method in side channel analysis. In this thesis, a literature study of theoretical developments and current state of mutual information analysis is provided. Furthermore, two methods for mutual information estimation are assessed: histograms and cumulants. A Java-implementation of mutual information analysis is developed and design decisions are discussed. Results on the performance of mutual information analysis on both simulations and practical cryptographic embeddings, obtained using the implemented module, are provided, as well as a theoretical and practical comparison of mutual information analysis with correlation power analysis. Mutual information analysis is shown to be of specific interest on practical high security cryptographic embeddings where predictions and power consumption observations show nonlinear dependence.

Contents

Chapter 1

Introduction

It is the mark of an instructed mind to rest satisfied with the degree of precision to which the nature of the subject admits and not to seek exactness when only an approximation of the truth is possible.

— Aristotle, *Nicomachean Ethics*, I.1094b24

1.1 Motivation

Cryptography aims at protecting data from possibly malicious third parties. Implementation of protection should be such that security and computational complexity are balanced; these two properties of a cryptographic system go hand in hand. Consider, for example, the recently introduced system for electronic payment in the Dutch public transport that uses contactless smart cards to carry travel credit. A small cryptographic protocol is executed when a commuter checks in at a station. However, cryptography should not consume much time; the train might soon depart. This requirement does not benefit security; multiple parties have already shown attacks on the public transport chipcard.

The balance phenomenon on computational complexity versus security also applies to attacks on cryptographic systems. A very detailed attack might yield computational overhead or even be infeasible. A rough estimation, however, might already be adequate to reveal information about, e.g., secret keys. Evidently, the extent to which approximations are sound is of crucial influence on the success probability of an attack.

Side channel analysis is a general type of cryptographic attack on embedded cryptography, aiming at retrieving information about an embedded secret key via publicly available interfaces, such as power consumption or electromagnetic radiation. Differential power analysis is a type of side channel analysis, specifically focusing on retrieving this information via statistical analysis of power consumption of devices that perform cryptographic operations. Statistical analysis involves adequately predicting and modeling power consumption based on key guesses, and testing for which key guess predictions and power consumption measurements match the best. The matching criterion is referred to as a distinguisher.

To predict power consumption, multiple models are proposed in literature (e.g., by Mangard et al. [MOP07]). Furthermore, many ways to compare predictions and observations exist. Most contemporary attacks compute correlation coefficients of predictions and observations, this method is called correlation power analysis. Correlation coefficients are lightweight and fast; however, only linear relations are in scope of this distinguisher. In 2008, Gierlichs et al. [GBTP08] introduced *mutual information analysis*, a more generic side channel distinguisher able of assessing relations between predictions and observations more accurately, that is, not limited to linear relations. Therefore, key guesses are distinguished based on both linear and nonlinear relations between

predictions and observations, yielding more accurate likelihoods for the respective key guesses. However, improving accuracy again implies an increase in computational complexity.

Multiple alternatives to correlation power analysis have been published in the open cryptographic community. Mangard et al. ([MOP07], Section 6.5) summarize some alternatives to usage of correlation coefficients, including difference of means and generalized maximum-likelihood testing. However, no mutual information-based distinguisher is proposed. Mutual information analysis is, in that perspective, a true innovation in the field of side channel analysis. Since mutual information analysis is easily generalized to more dimensions, a mutual information attack is also applicable on protected implementations, making it into an even more interesting alternative.

This thesis is the result of a graduation project on mutual information analysis, that has been carried out at Riscure BV, Delft. Riscure (http://www.riscure.com) is an independent security test laboratory, specialized in evaluating the security of products that perform cryptographic operations in public environments, such as smart cards, SIM cards or digital television decoders. Riscure's security tests on cryptographic implementations aim at assessing the ease and cost of successful attacks. Therefore, maintaining and extending a wide variety of cryptographic attacks is an important aspect in Riscure's business.

The project and this thesis contribute to Riscure's intelligence and knowledge, by thoroughly evaluating the benefits of mutual information analysis on practical cryptographic implementations. Furthermore, an extensive comparison with contemporary correlation analysis is carried out, indicating how the new mutual information analysis relates to current attacks. Part of the investigation consists of developing a software implementation of mutual information analysis, to actually perform attacks on power consumption measurements.

1.2 Goals

The following goals have been set at the start of this research project.

- To describe mutual information analysis extensively and appoint crucial decisions or steps in the analysis. To investigate into mutual information analysis on both unprotected and protected cryptographic implementations.
- To compare mutual information analysis with correlation power analysis and investigate into a hypothetical preference rule for the methods in different scenarios.
- To evaluate mutual information analysis on applicability for Riscure's attack portfolio. To provide evidence by means of results on simulations and practical cryptographic embeddings. This involves the creation of a software prototype capable of performing mutual information analysis on power consumption measurements. When the prototype shows promising results, it can serve as model for a complete software module. Then, an additional goal is to define the necessities of a MIA module.

1.3 Thesis Outline

This chapter serves as introduction to the thesis, explaining motivation and goals. Preliminaries on information-theoretic concepts and probability theory are given in Chapter 2. Differential power analysis, of which mutual information analysis is an instance, and correlation power analysis are explained in Chapter 3. Furthermore, multiple prediction methods for differential power analysis are treated. Then, first order mutual information analysis is extensively explained in Chapter 4. Second order mutual information analysis is explained afterwards, in Chapter 5. In Chapter 6, information about implementing mutual information analysis for practical attacks is given. Moreover, design decisions are explained and motivated. Results on both simulations and attacks on real world embeddings are presented in Chapter 7. This thesis is concluded in Chapter 8.

Chapter 2

Preliminaries

In this chapter, information-theoretic preliminaries are provided. These definitions, properties and relations serve as foundation for the remainder of this thesis. Many sections have a strongly theoretical nature. The reader is referred to these sections from later chapters, when theoretical concepts are applied.

The reader is assumed to have basic knowledge of random variables and probability density functions. The structures of the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are assumed to be familiar; however, a description of both cryptographic algorithms is included in the appendix.

2.1 Entropy

Entropy of a random variable is a measure of information. Entropy measures the uncertainty in a random variable; the higher the entropy, the more unpredictable the outcome is and the more information arises from one realization. The concept of entropy was introduced by Shannon [Sha48]. In this section, entropy and related concepts are defined and explained.

Let p denote the probability mass function of a discrete random variable $\mathbf{X}$ over a space $\mathcal{X}$ with n outcomes.

Definition 2.1.1. *(Entropy)*
The entropy of a discrete random variable $\mathbf{X}$ *is defined as:*

$$H(\mathbf{X}) = -\sum_{i=1}^{n} p(x_i) \log(p(x_i)).^1 \tag{2.1}$$

Entropy is completely defined by the probability mass function. Therefore, an alternative notation for entropy of a random variable with distribution p is $H(p)$. Logarithms are taken in base 2, so entropy is measured in bits. In case $p(x_i) = 0$ for some i, the summand is defined to be equal to zero according to

$$\lim_{x \to 0} x \log(x) = 0.$$

Consequently, adding terms of zero probability does not affect entropy. Furthermore, entropy of $\mathbf{X}$ does not depend on the values attained by $\mathbf{X}$; only on the probabilities. Note that entropy is nonnegative, since $0 \leq p(x_i) \leq 1$ for all $i = 1, \ldots, n$.

The empirical probability mass function of a discrete random variable $\mathbf{X}$ is estimated from realizations of $\mathbf{X}$. To obtain this function, one draws a sample of size q from $\mathbf{X}$ and determines

[1] Througout this thesis, the base of the logarithm is 2.

for each possible outcome x_i $(i = 1, \ldots, n)$ the number of times q_i that $\mathbf{X} = x_i$. The empirical probability distribution is then defined as

$$p' = \left(\frac{q_1}{q}, \frac{q_2}{q}, \ldots, \frac{q_n}{q} \right).$$

Consequently, the empirical probability mass function (EPMF) is defined as

$$P(\mathbf{X} = x_i) = \frac{q_i}{q},$$

for $i = 1, \ldots, n$. Note that $\sum_{i=1}^{n} q_i = q$.

Entropy of an empirical distribution is also referred to as empirical entropy. This quantity measures the amount of information one obtains from realizations of a random variable. Empirical entropy is defined analogously to regular entropy.

Definition 2.1.2. *(Empirical entropy)*
The entropy of an empirical distribution p' is defined as:

$$\begin{aligned} H(p') &= -\sum_{i=1}^{n} \frac{q_i}{q} \log \left(\frac{q_i}{q} \right) \\ &= \frac{1}{q} \sum_{i=1}^{n} q_i \log \left(\frac{q}{q_i} \right). \end{aligned} \tag{2.2}$$

Entropy of a random variable has a natural generalization to suit multidimensional random variables. Since entropy is completely determined by a probability distribution, entropy of a joint distribution of components of a multidimensional random variable exists as well. For a two-dimensional random variable, joint entropy is defined as follows.

Definition 2.1.3. *Let $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ have probability mass function $p(z) = p(x, y)$. The entropy of $\mathbf{Z}$ is defined as:*

$$H(\mathbf{Z}) = H(\mathbf{X}, \mathbf{Y}) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log(p(x_i, y_j)).$$

Note that $H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{Y}, \mathbf{X})$.

Analogous to Definition 2.1.1, joint entropy is nonnegative with equality if there is a pair i, j such that $q_{i,j} = q$. The definition can accordingly be extended to n-dimensional random variables $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_n)$ by using an appropriate joint density function $p(z_1, \ldots, z_n)$ and summing over each possible combination of outcomes. However, including the same random variable twice does not give more information.

Lemma 2.1.4. *Let $\mathbf{X}$ be a random variable. Then $H(\mathbf{X}, \mathbf{X}) = H(\mathbf{X})$.*

Proof.

$$\begin{aligned} H(\mathbf{X}, \mathbf{X}) &= -\sum_{i=1}^{n} \sum_{j=1}^{n} p(x_i, x_j) \log(p(x_i, x_j)) \\ &= -\sum_{i=1}^{n} \sum_{j=1}^{n} p(x_i|x_j) p(x_j) \log(p(x_i|x_j) p(x_j)) \\ &= -\sum_{i=1}^{n} p(x_i) \log(p(x_i)) \\ &= H(\mathbf{X}). \end{aligned}$$

The third equality holds since $p(x_i|x_j) = 1$ for $j = i$ and zero otherwise. Hence, the sum over all j consists of only the term $j = i$. □

Analogous to the empirical entropy of one random variable, the empirical joint entropy of a collection of random variables (or components of a multidimensional random variable) is defined using the empirical joint probability function.

Definition 2.1.5. *(Empirical joint entropy)*
Analogous to empirical entropy, empirical joint entropy of $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$ *is defined as*

$$\begin{aligned} H(\mathbf{Z}) &= -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log(p(x_i, y_j)) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{q_{i,j}}{q}\log\left(\frac{q_{i,j}}{q}\right) \\ &= \frac{1}{q}\sum_{i=1}^{n}\sum_{j=1}^{m} q_{i,j}\log\left(\frac{q}{q_{i,j}}\right). \end{aligned}$$

Joint probability functions have interesting properties; besides carrying information about joint behavior, information about individual and conditional behavior of random variables can be extracted. This latter information can be reflected in conditional entropy. Intuitively, this is the uncertainty one has about one random variable when one knows a realization of another random variable.

Definition 2.1.6. *(Conditional entropy)*
Let $\mathbf{X}$ *have density function* $p(x)$ *and let* $\mathbf{Y}$ *have density function* $p(y)$*. Denote by* $p(x, y)$ *the joint density function. The conditional entropy of* $\mathbf{X}$ *given* $\mathbf{Y}$ *is defined as:*

$$\begin{aligned} H(\mathbf{X}|\mathbf{Y}) &= -\sum_{j=1}^{m}\sum_{i=1}^{n} p(x_i, y_j)\log(p(x_i|y_j)) \\ &= \sum_{j=1}^{m}\sum_{i=1}^{n} p(x_i, y_j)\log\left(\frac{p(y_j)}{p(x_i, y_j)}\right). \end{aligned}$$

The intuition of conditional entropy might be explained in another way. Conditional entropy represents the amount of uncertainty one has about the outcomes of two random variables, minus the amount of information one of the two random variables gives. This intuition is reflected in the following expression, linking conditional entropy to joint entropy.

Lemma 2.1.7. *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be random variables. Then* $H(\mathbf{X}|\mathbf{Y}) = H(\mathbf{X}, \mathbf{Y}) - H(\mathbf{Y})$.

Proof.

$$\begin{aligned} H(\mathbf{X}|\mathbf{Y}) &= \sum_{j=1}^{m}\sum_{i=1}^{n} p(x_i, y_j)\log\left(\frac{p(y_j)}{p(x_i, y_j)}\right) \\ &= -\sum_{j=1}^{m}\sum_{i=1}^{n} p(x_i, y_j)\log(p(x_i, y_j)) + \sum_{j=1}^{m}\sum_{i=1}^{n} p(x_i, y_j)\log(p(y_j)) \\ &= H(\mathbf{X}, \mathbf{Y}) + \sum_{j=1}^{m} p(y_j)\log(p(y_j)) \\ &= H(\mathbf{X}, \mathbf{Y}) - H(\mathbf{Y}). \end{aligned}$$

□

Corollary 2.1.8. *By symmetry and since* $H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{Y}, \mathbf{X})$*:*

$$H(\mathbf{Y}|\mathbf{X}) = H(\mathbf{X}, \mathbf{Y}) - H(\mathbf{X}).$$

A rather trivial but interesting property is that conditional entropy of one variable equals zero. Obviously, uncertainty on $\mathbf{X}$ vanishes once one conditions on its outcome.

Corollary 2.1.9. *By Corollary 2.1.8 and Lemma 2.1.4,* $H(\mathbf{X}|\mathbf{X}) = H(\mathbf{X}, \mathbf{X}) - H(\mathbf{X}) = 0$.

In probability theory, studies on the quantification of distance between two probability functions are widely performed. One of the measures of distance between two probability functions is the Kullback-Leibler divergence, also referred to as relative entropy (see, e.g., Cover and Thomas [CT91]). The Kullback-Leibler divergence of two random variables equals the Kullback-Leibler divergence of the respective probability distributions.

Definition 2.1.10. *(Kullback-Leibler divergence)*
Let $\mathbf{X}$ *and* $\mathbf{Y}$ *be random variables over* $\mathcal{X}$ *with probability density functions* $p(x)$ *and* $p(y)$ *respectively. The Kullback-Leibler divergence from* $\mathbf{X}$ *to* $\mathbf{Y}$ *is defined as:*

$$D_{KL}(\mathbf{X}||\mathbf{Y}) = \sum_{i=1}^{n} p(x_i) \log \left(\frac{p(x_i)}{p(y_i)} \right). \tag{2.3}$$

An alternative notation for $D_{KL}(\mathbf{X}||\mathbf{Y})$ *is* $D_{KL}(p(x)||p(y))$*, to explicitly denote the distribution functions of* $\mathbf{X}$ *and* $\mathbf{Y}$*. The summand is defined for those* i *where*

$$p(x_i) > 0 \Rightarrow p(y_i) > 0.$$

When $p(x_i) = 0$*, the summand is treated as zero, according to*

$$\lim_{x \to 0} x \log x = 0.$$

Note that Kullback-Leibler divergence is asymmetric.

Lemma 2.1.11. *Kullback-Leibler divergence is nonnegative.*

Proof. Recall that $\log(a) = \ln(a)/\ln(2)$ and that $\ln(x) \leq x - 1$, with equality if and only if $x = 1$.

$$\begin{aligned}
D_{\mathrm{KL}}(\mathbf{X}||\mathbf{Y}) &= \sum_{i=1}^{n} p(x_i) \log \left(\frac{p(x_i)}{p(y_i)} \right) \\
&= -\sum_{i=1}^{n} p(x_i) \log \left(\frac{p(y_i)}{p(x_i)} \right) \\
&= -\sum_{i:p(x_i)>0} p(x_i) \log \left(\frac{p(y_i)}{p(x_i)} \right) \\
&\geq \sum_{i:p(x_i)>0} p(x_i) \left(1 - \frac{p(y_i)}{p(x_i)} \right) \\
&= \sum_{i:p(x_i)>0} p(x_i) - \sum_{i:p(x_i)>0} p(y_i) \\
&= 1 - \sum_{i:p(x_i)>0} p(y_i) \geq 0.
\end{aligned}$$

This result is known as Gibbs' inequality. Equality follows when $p(x_i) = p(y_i)$ for all i, i.e., when $\mathbf{X}$ and $\mathbf{Y}$ have equal distributions. □

Kullback-Leibler divergence is a useful tool to examine relations between random variables, as well as properties of entropy. In particular, the upper bound of entropy can be derived.

Lemma 2.1.12. *Entropy of a random variable* $\mathbf{X}$ *over* $\mathcal{X}$ *is bounded by the entropy of a uniform random variable over* $\mathcal{X}$*.*

Proof. Let $|\mathcal{X}| = n$ and let $\mathbf{Y}$ be a uniform random variable over $\mathcal{X}$, i.e., the outcomes of $\mathbf{Y}$ are equiprobable with probability $\frac{1}{n}$.

$$\begin{aligned} D_{\mathrm{KL}}(\mathbf{X}||\mathbf{Y}) &= \sum_{i=1}^{n} p(x_i) \log\left(\frac{p(x_i)}{p(y_i)}\right) \\ &= \sum_{i=1}^{n} p(x_i)(\log(p(x_i)) - \log\left(\frac{1}{n}\right)) \\ &= \log n - H(\mathbf{X}). \end{aligned}$$

By Lemma 2.1.11, $0 \leq D_{\mathrm{KL}}(\mathbf{X}||\mathbf{Y}) = \log n - H(\mathbf{X})$, hence $H(\mathbf{X}) \leq \log n$.

Evidently, uncertainty is maximal when all events are equally likely, i.e., no single event is more likely than one other. □

In the previous definitions, only discrete random variables are examined. However, entropy is easily generalized to random variables with continuous probability distributions. Entropy of continuous random variables is also referred to as differential entropy and allows estimation of entropy of some well-known continuous probability functions, in terms of their parameters.

Definition 2.1.13. *(Continuous entropy)*
Let $\mathbf{X}$ *be a continuous random variable with probability density function* $f_{\mathbf{X}}$*. Then*

$$H(\mathbf{X}) = -\int_{-\infty}^{\infty} f_{\mathbf{X}}(x) \log(f_{\mathbf{X}}(x)) dx. \tag{2.4}$$

Entropy is completely defined by probability and hence the measure is invariant under, e.g., a transformation $\mathbf{X} \longmapsto \mathbf{X} + a$ ($a \in \mathbb{R}$). Hence, equivalence classes of random variables arise when a random variable differs from another by a constant.

Lemma 2.1.14. *Let* $\mathbf{X} \sim \mathcal{N}(\mu, \sigma^2)$ *and* $\mathbf{Y} \sim \mathcal{N}(0, \sigma^2)$*. Then* $H(\mathbf{X}) = H(\mathbf{Y})$.

Proof. Substitute $y = x - \mu$ in the definition of entropy. As a consequence, $dy = dx$.

$$\begin{aligned} H(\mathbf{X}) &= -\int_{-\infty}^{\infty} f_{\mathbf{X}}(x) \log(f_{\mathbf{X}}(x)) dx \\ &= -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) dx \\ &= -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}\right) dy \\ &= H(\mathbf{Y}). \end{aligned}$$

□

In the following example, entropy of a normally distributed random variable $\mathbf{X}$ is estimated. Since entropy of a normally distributed random variable is translation invariant, $\mathbf{X}$ is chosen to have mean zero.

Example 2.1.15. *Let* $\mathbf{X} \sim \mathcal{N}(0, \sigma^2)$*. The probability density function of* $\mathbf{X}$ *is given by:*

$$f_{\mathbf{X}}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}.$$

For the entropy of $\mathbf{X}$ *holds:*

$$\begin{aligned}
H(\mathbf{X}) &= -\int_{-\infty}^{\infty} f_{\mathbf{X}}(x)\log(f_{\mathbf{X}}(x))dx \\
&= -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}\right) dx \\
&= -\frac{1}{\ln 2}\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} (-\frac{x^2}{2\sigma^2} - \frac{1}{2}\ln(2\pi\sigma^2))dx \\
&= \frac{1}{\ln 2}\int_{-\infty}^{\infty} \frac{x^2}{2\sigma^2}\cdot\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} dx + \frac{1}{2}\ln(2\pi\sigma^2)\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} dx \\
&= \frac{1}{\ln 2}\left(\frac{E[\mathbf{X}^2]}{2\sigma^2} + \frac{1}{2}\ln(2\pi\sigma^2)\right) \\
&= \frac{1}{\ln 2}\left(\frac{1}{2} + \frac{1}{2}\ln(2\pi\sigma^2)\right) \\
&= \frac{1}{\ln 2}\left(\frac{1}{2}\ln(2e\pi\sigma^2)\right) \\
&= \frac{1}{2}\log(2e\pi\sigma^2).
\end{aligned}$$

Here, $E[\mathbf{X}^2]$ *denotes expectation of the second raw moment of* $\mathbf{X}$*, see Section 2.4. As can be seen, entropy of a normally distributed continuous random variable can easily be computed once the variance is known.*

2.2 Mutual Information

Mutual information is the central concept in mutual information analysis. Therefore, an extensive treatment of mutual information is given here (more information in, e.g., Cover and Thomas [CT91]). Intuitively, mutual information measures the amount of information shared by two random variables. A high mutual information means that two random variables share much information, hence are strongly related. Low mutual information means that random variables appear to be weakly related. Note, however, that the exact way in which two random variables are related is not important. Indeed, mutual information quantifies any kind of dependence among random variables.

Let $\mathbf{X}$ and $\mathbf{Y}$ be two random variables over spaces $\mathcal{X}$, resp. $\mathcal{Y}$, with probability density functions $f_{\mathbf{X}}$, resp. $f_{\mathbf{Y}}$. Denote the joint density function by $f_{\mathbf{X},\mathbf{Y}}(x,y)$, $x \in \mathcal{X}, y \in \mathcal{Y}$.

Definition 2.2.1. *(Mutual information)*

Mutual information (MI) of $\mathbf{X}$ *and* $\mathbf{Y}$ *is defined in terms of Shannon entropy as:*

$$I(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{X},\mathbf{Y}) \tag{2.5}$$
$$= H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) \tag{2.6}$$
$$= H(\mathbf{Y}) - H(\mathbf{Y}|\mathbf{X}). \tag{2.7}$$

Equalities 2.6 and 2.7 follow from Lemma 2.1.7 and Corollary 2.1.8 respectively. Note that this implies symmetry:

$$I(\mathbf{X};\mathbf{Y}) = I(\mathbf{Y};\mathbf{X}).$$

Figure 2.1 summarizes the relations between mutual information and entropy.

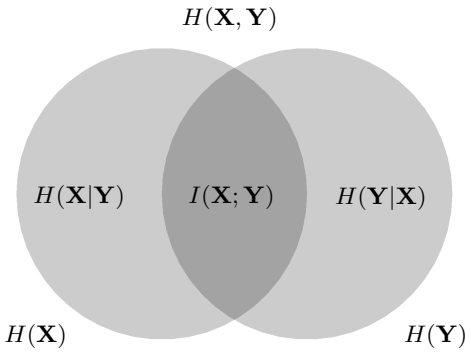


Figure 2.1: Information diagram of mutual information and entropy.

Mutual information can be generalized to random variables from higher dimensions in multiple ways. One generalization is called *total correlation* and measures the amount of information shared among at least any two components. Three other generalizations are introduced in Section 5.4.2.

Definition 2.2.2. *Let* $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$*. The total correlation of the components of* $\mathbf{X}$ *is given by:*

$$C(\mathbf{X}) = \sum_{i=1}^{n} H(\mathbf{X}_i) - H(\mathbf{X}). \tag{2.8}$$

Mutual information can directly be estimated from probability distributions using the definitions of entropy from the previous section, and any of the equations from Definition 2.2.1. One obtains the following expression for discrete random variables.

Definition 2.2.3. *(Mutual information, discrete)*
In terms of probability distributions, mutual information of discrete random variables $\mathbf{X}$ *and* $\mathbf{Y}$ *is defined as*

$$I(\mathbf{X};\mathbf{Y}) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} f_{\mathbf{X},\mathbf{Y}}(x,y) \log \left(\frac{f_{\mathbf{X},\mathbf{Y}}(x,y)}{f_{\mathbf{X}}(x) f_{\mathbf{Y}}(y)} \right). \tag{2.9}$$

Mutual information can be interpreted as an instance of Kullback-Leibler divergence (Definition 2.1.10):

$$I(\mathbf{X};\mathbf{Y}) = D_{KL}(f_{\mathbf{X},\mathbf{Y}} || f_{\mathbf{X}} f_{\mathbf{Y}}). \tag{2.10}$$

Mutual information of continuous random variables is defined analogous to entropy of continuous random variables.

Definition 2.2.4. *(Mutual information, continuous)*
Mutual information of continuous random variables $\mathbf{X}$ *and* $\mathbf{Y}$ *is defined as*

$$I(\mathbf{X};\mathbf{Y}) = \int_{\mathcal{X}} \int_{\mathcal{Y}} f_{\mathbf{X},\mathbf{Y}}(x,y) \log \left(\frac{f_{\mathbf{X},\mathbf{Y}}(x,y)}{f_{\mathbf{X}}(x) f_{\mathbf{Y}}(y)} \right) dy \; dx.$$

Since mutual information is an instance of Kullback-Leibler divergence, properties of Kullback-Leibler divergence apply to mutual information. An important one is nonnegativity.

Lemma 2.2.5. $I(\mathbf{X};\mathbf{Y}) \geq 0$ *with equality if and only if* $\mathbf{X}$ *and* $\mathbf{Y}$ *are independent.*

Proof. From Lemma 2.1.11, one obtains:

$$I(\mathbf{X};\mathbf{Y}) = D_{\mathrm{KL}}(f_{\mathbf{X},\mathbf{Y}} || f_{\mathbf{X}} f_{\mathbf{Y}}) \geq 0.$$

Equality follows if and only if $f_{\mathbf{X},\mathbf{Y}} = f_{\mathbf{X}} f_{\mathbf{Y}}$. This characterization is equivalent to $\mathbf{X}$ and $\mathbf{Y}$ being independent. □

This expression yields a bound for conditional entropy. Evidently, uncertainty about one random variable can not increase once one knows the outcome of another random variable. This is reflected in the following corollary.

Corollary 2.2.6. *From* $0 \leq I(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y})$*, one obtains* $H(\mathbf{X}|\mathbf{Y}) \leq H(\mathbf{X})$.

The amount of information shared by two identical random variables equals the amount of information provided by random variable. Due to the following identity, entropy is alternatively called self-information.

Corollary 2.2.7. *By Corollary 2.1.9,*

$$I(\mathbf{X};\mathbf{X}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{X}) = H(\mathbf{X}).$$

Mutual information is bounded below by zero. The upper bound can easily be obtained from previously given relations.

Definition 2.2.8. *The fraction*

$$I_N(\mathbf{X};\mathbf{Y}) = \frac{I(\mathbf{X};\mathbf{Y})}{\min\{H(\mathbf{X}), H(\mathbf{Y})\}}$$

is called normalized mutual information *of* $\mathbf{X}$ *and* $\mathbf{Y}$.

Note that mutual information is bounded:

$$I(\mathbf{X};\mathbf{Y}) \leq \min\{H(\mathbf{X}), H(\mathbf{Y})\}.$$

Equality is obtained when $\mathbf{X}$ fully determines $\mathbf{Y}$, or the other way around. As a consequence, normalized mutual information attains values between zero and one.

A higher order generalization of mutual information, introduced in Definition 2.2.2 is total correlation. Total correlation can be obtained by evaluating the Kullback-Leibler divergence from the joint probability density function to the product of marginal probability density functions.

Definition 2.2.9. *Let* $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$ *be defined over an* n*-dimensional space* $\mathcal{X}$*. The total correlation of the components of* $\mathbf{X}$ *is defined as:*

$$C(\mathbf{X}_1, \ldots, \mathbf{X}_n) = \sum_{(x_1,\ldots,x_n)\in\mathcal{X}} f_{\mathbf{X}_1,\ldots,\mathbf{X}_n}(x_1, \ldots, x_n) \log\left(\frac{f_{\mathbf{X}_1,\ldots,\mathbf{X}_n}(x_1, \ldots, x_n)}{f_{\mathbf{X}_1}(x_1) \cdot \ldots \cdot f_{\mathbf{X}_n}(x_n)}\right). \tag{2.11}$$

Mutual information of discrete random variables is an information preserving measure under bijective mappings. For example, mutual information is invariant under affine transformations.

Lemma 2.2.10. *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be random variables with arbitrary distributions* $f_{\mathbf{X}}$ *resp.* $f_{\mathbf{Y}}$ *defined over* $\mathcal{X}$ *resp.* $\mathcal{Y}$*. Let* $a, b, c, d \in \mathbb{R}$ *with* $b, d \neq 0$*. Let*

$$\tilde{\mathbf{X}} = \frac{\mathbf{X} - a}{b} \text{ and } \tilde{\mathbf{Y}} = \frac{\mathbf{Y} - c}{d}$$

be defined over $\mathcal{X}'$ *resp.* $\mathcal{Y}'$*. Then*

$$I(\mathbf{X};\mathbf{Y}) = I(\tilde{\mathbf{X}};\tilde{\mathbf{Y}}).$$

Proof. Since affine transformations are bijective, for each $x' \in \mathcal{X}'$ there is a unique $x \in \mathcal{X}$ such that $x = bx' + a$. Analogously for each $y' \in \mathcal{Y}'$ there is a unique $y \in \mathcal{Y}$ such that $y = dy' + c$. The distribution of $\tilde{\mathbf{X}}$ is equal to $f_{\tilde{\mathbf{X}}}(x') = f_{\mathbf{X}}(bx' + a)$ for $x' \in \mathcal{X}'$ and the distribution of $\tilde{\mathbf{Y}}$ is equal

to $f_{\tilde{\mathbf{Y}}}(y') = f_{\mathbf{Y}}(dy' + c)$ for $y' \in \mathcal{Y}'$. Consequently, for mutual information holds:

$$\begin{aligned} I(\tilde{\mathbf{X}};\tilde{\mathbf{Y}}) &= \sum_{x'\in\mathcal{X}'}\sum_{y'\in\mathcal{Y}'} f_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}}}(x',y') \log\left(\frac{f_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}}}(x',y')}{f_{\tilde{\mathbf{X}}}(x')f_{\tilde{\mathbf{Y}}}(y')}\right) \\ &= \sum_{x'\in\mathcal{X}'}\sum_{y'\in\mathcal{Y}'} f_{\mathbf{X},\mathbf{Y}}(bx'+a,dy'+c) \log\left(\frac{f_{\mathbf{X},\mathbf{Y}}(bx'+a,dy'+c)}{f_{\mathbf{X}}(bx'+a)f_{\mathbf{Y}}(dy'+c)}\right) \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} f_{\mathbf{X},\mathbf{Y}}(x,y) \log\left(\frac{f_{\mathbf{X},\mathbf{Y}}(x,y)}{f_{\mathbf{X}}(x)f_{\mathbf{Y}}(y)}\right) \\ &= I(\mathbf{X};\mathbf{Y}). \end{aligned}$$

□

2.3 Correlation

A more restricted measure of dependence is the correlation coefficient, sometimes referred to as Pearson's correlation (see, e.g., Bain and Engelhardt [BE92]). This is a measure of linear dependence of two random variables. It indicates to what extent the relationship of two random variables is linear-like.

Let $\mathbf{X}$ and $\mathbf{Y}$ be random variables having arbitrary distributions with respective means $\mu_{\mathbf{X}}$ and $\mu_{\mathbf{Y}}$ and standard deviations $\sigma_{\mathbf{X}}$ and $\sigma_{\mathbf{Y}}$.

Definition 2.3.1. *(Correlation coefficient)*
The correlation coefficient of random variables $\mathbf{X}$ *and* $\mathbf{Y}$ *is defined as the covariance of* $\mathbf{X}$ *and* $\mathbf{Y}$*, divided by the product of their standard deviations.*

$$\rho(\mathbf{X},\mathbf{Y}) = \frac{cov(\mathbf{X},\mathbf{Y})}{\sigma_{\mathbf{X}}\sigma_{\mathbf{Y}}} = \frac{E[(\mathbf{X}-\mu_{\mathbf{X}})(\mathbf{Y}-\mu_{\mathbf{Y}})]}{\sigma_{\mathbf{X}}\sigma_{\mathbf{Y}}}. \tag{2.12}$$

The sample correlation coefficient from a set of paired data can be used to estimate the correlation coefficient of underlying distributions, when one has incomplete information about the exact means or standard deviations.

Definition 2.3.2. *(Sample correlation)*
Suppose $(x_1,y_1),\ldots,(x_n,y_n)$ *are paired data samples from* ($\mathbf{X}$, $\mathbf{Y}$)*. The sample correlation coefficient is defined as*

$$r(\mathbf{X},\mathbf{Y}) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \tag{2.13}$$

where $\bar{x}$ *and* $\bar{y}$ *are the respective sample means.*

For large n, the sample correlation coefficient approximates the actual correlation coefficient.

Correlation coefficients attain values in $[-1,1]$, where a correlation of 1 indicates an increasing, perfectly linear relationship and a correlation of -1 indicates a decreasing perfectly linear relationship. If two variables are independent, their correlation coefficient equals zero. However, the converse does not hold, since correlation is only sensitive to linear dependence.

2.4 Moments and Cumulants

The moments of a probability distribution $f_{\mathbf{X}}$ associated with a random variable $\mathbf{X}$ are quantities providing information about the shape and location of the distribution. Let $E[\mathbf{X}]$ denote the expectation of $\mathbf{X}$.

Definition 2.4.1. *The i-th raw moment μ_i' of a random variable $\mathbf{X}$ is defined as*

$$\mu_i' = E[\mathbf{X}^i].$$

Analogously, the i-th raw sample moment m_i' of a sample from $\mathbf{X}$ is defined as

$$m_i' = \frac{1}{n}\sum_{k=1}^{n}(x_k)^i,$$

where $x_1, \ldots, x_n$ are realizations of $\mathbf{X}$.

The first raw moment of a random variable $\mathbf{X}$ is equal to the mean: $\mu_1' = E[\mathbf{X}] = \mu_{\mathbf{X}}$. The standard deviation of $\mathbf{X}$ can be expressed in terms of raw moments by

$$\sigma_{\mathbf{X}}^2 = E[\mathbf{X}^2] - E[\mathbf{X}]^2 = \mu_2' - \mu_1'^2.$$

The raw moments of a random variable can be calculated by evaluating the moment generating function at zero, if this function is known.

Definition 2.4.2. *The moment generating function of a random variable $\mathbf{X}$ is defined as*

$$M_{\mathbf{X}}(t) = E\left[e^{t\mathbf{X}}\right], \quad t \in \mathbb{R}.$$

Since $e^{t\mathbf{X}} = 1 + t\mathbf{X} + \frac{t^2\mathbf{X}^2}{2!} + \frac{t^3\mathbf{X}^3}{3!} + \ldots$, one obtains

$$E\left[e^{t\mathbf{X}}\right] = 1 + t\mu_1' + \frac{t^2\mu_2'}{2!} + \frac{t^3\mu_3'}{3!} + \ldots. \tag{2.14}$$

The i-th moment μ_i' can be obtained by calculating $\frac{d^i M_{\mathbf{X}}}{dt^i}(0) = M_{\mathbf{X}}^{(i)}(0)$, the i-th derivative of $M_{\mathbf{X}}$ evaluated at $t = 0$. Conversely, the moment generating function can be expressed in terms of moments as:

$$M_{\mathbf{X}}(t) = 1 + \sum_{i=1}^{\infty}\mu_i'\frac{t^i}{i!}.$$

After subtracting the mean from a random variable (i.e., centralizing the random variable), central moments can be estimated analogously to raw moments.

Definition 2.4.3. *The i-th central moment μ_i of a random variable $\mathbf{X}$ having mean $\mu_{\mathbf{X}}$ is defined as*

$$\mu_i = E[(\mathbf{X} - \mu_{\mathbf{X}})^i].$$

Accordingly, the i-th central sample moment m_i of a random variable $\mathbf{X}$ with mean $\mu_{\mathbf{X}}$ is defined as

$$m_i = \frac{1}{n}\sum_{k=1}^{n}(x_k - \mu_{\mathbf{X}})^i,$$

where $x_1, \ldots, x_n$ are realizations of $\mathbf{X}$.

The first central moment of $\mathbf{X}$ equals zero, since $E[\mathbf{X} - \mu_{\mathbf{X}}] = E[\mathbf{X}] - \mu_{\mathbf{X}} = 0$. The second central moment of $\mathbf{X}$ equals the variance of $\mathbf{X}$:

$$\mu_2 = E[(\mathbf{X} - \mu_{\mathbf{X}})^2] = E[\mathbf{X}^2] - 2\mu_{\mathbf{X}}E[\mathbf{X}] + \mu_{\mathbf{X}}^2 = E[\mathbf{X}^2] - \mu_{\mathbf{X}}^2 = \sigma_{\mathbf{X}}^2.$$

Moments can be generalized for multiple random variables to obtain joint moments. Every finite product of random variables gives rise to a set of joint moments.

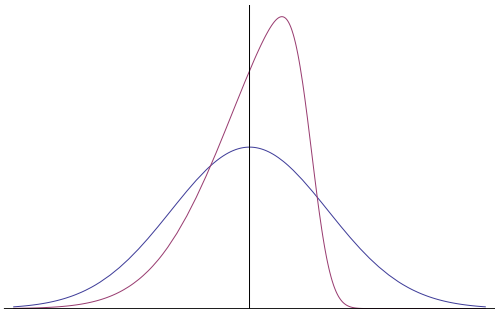


Figure 2.2: A symmetric (blue) and negatively skewed (red) distribution.

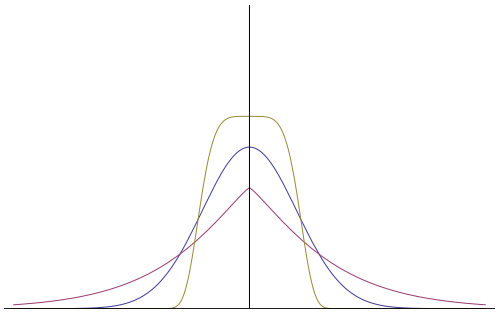


Figure 2.3: Distributions with excess kurtosis zero (blue), negative (red) and positive (gold).

Definition 2.4.4. *Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$ be an n-dimensional random variable. The joint raw moment $\mu'_{i_1,\ldots,i_n}$ is defined as*

$$\mu'_{i_1,\ldots,i_n} = E[\mathbf{X}_1^{i_1} \ldots \mathbf{X}_n^{i_n}].$$

The joint central moment $\mu_{i_1,\ldots,i_n}$ is defined as

$$\mu_{i_1,\ldots,i_n} = E[(\mathbf{X}_1 - \mu_{\mathbf{X}_1})^{i_1} \cdot \ldots \cdot (\mathbf{X}_n - \mu_{\mathbf{X}_n})^{i_n}].$$

The joint raw moment generating function of $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$ has the form

$$M_{\mathbf{X}}(t_1, \ldots, t_n) = E\left[e^{t_1\mathbf{X}_1 + \cdots + t_n\mathbf{X}_n}\right]. \tag{2.15}$$

The joint raw moment $\mu'_{i_1,\ldots,i_n}$ of $\mathbf{X}$ can be obtained by evaluating

$$\mu'_{i_1,\ldots,i_n} = E[\mathbf{X}_1^{i_1} \ldots \mathbf{X}_n^{i_n}] = \frac{\partial^{i_1+\cdots+i_n} M_{\mathbf{X}}}{\partial t_i^{i_1} \ldots \partial t_n^{i_n}}(0, \ldots, 0).$$

The mean of a probability distribution is a characterization of location, the standard deviation indicates spread. Two measures of shape of a probability distribution are skewness and excess kurtosis. Skewness is a measure for asymmetry of a distribution: the more positive (negative) the skewness, the longer the right (left) tail of the distribution is. Excess kurtosis is a measure for data concentration around a peak of a probability distribution. A high kurtosis arises when more infrequent large deviations occur, instead of frequent small deviations. Figures 2.2 and 2.3 give a graphical representation of these two concepts. Both skewness and excess kurtosis are defined in terms of moments.

Definition 2.4.5. *(Skewness)*
The skewness γ_1 of a probability density function $f_{\mathbf{X}}$ is defined as

$$\gamma_1 = E\left[\left(\frac{\mathbf{X} - \mu_{\mathbf{X}}}{\sigma_{\mathbf{X}}}\right)^3\right] = \frac{\mu_3}{\sigma_{\mathbf{X}}^3}.$$

Definition 2.4.6. *(Excess kurtosis)*
The excess kurtosis γ_2 of a probability density function $f_{\mathbf{X}}$ is defined as

$$\gamma_2 = E\left[\left(\frac{\mathbf{X} - \mu_{\mathbf{X}}}{\sigma_{\mathbf{X}}}\right)^4\right] - 3 = \frac{\mu_4}{\sigma_{\mathbf{X}}^4} - 3.$$

The fraction $\frac{\mu_4}{\sigma_{\mathbf{X}}^4} = \frac{\mu_4}{\mu_2^2}$ is called kurtosis. Because the kurtosis of the normal distribution is equal to 3, excess kurtosis is defined in such a way that the normal distribution has excess kurtosis

zero. The sample skewness and sample kurtosis are derived by evaluating the second, third and fourth sample moment:

$$g_1 = \frac{m_3}{(m_2)^{\frac{2}{3}}}, \quad g_2 = \frac{m_4}{m_2^2} - 3.$$

As an alternative to the collection of moments, the collection of cumulants is considered to provide the same information; however, in certain cases computation involving cumulants reduces the complexity of expressions.

Definition 2.4.7. *The cumulants of a random variable* $\mathbf{X}$ *can be obtained by means of the cumulant generating function* G*. This function is equal to the logarithm of the moment generating function.*

$$G(t) = \log\left(E\left[e^{t\mathbf{X}}\right]\right) = -\sum_{n=1}^{\infty} \frac{1}{n}\left(1 - E\left[e^{t\mathbf{X}}\right]\right)^n = -\sum_{n=1}^{\infty} \frac{1}{n}\left(-\sum_{i=1}^{\infty} \mu_i' \frac{t^i}{i!}\right)^n \tag{2.16}$$

$$= \mu_1' t + (\mu_2' - \mu_1'^2)\frac{t^2}{2!} + (\mu_3' - 3\mu_2\mu_1' + 2\mu_1^3)\frac{t^3}{3!} + \dots. \tag{2.17}$$

Analogous to the i*-th moment, the* i*-th cumulant* κ_i *can be derived by evaluating the* i*-th derivative of* G *at zero:* $\kappa_i = G^{(i)}(0)$*.*

The fist four cumulants expressed in terms of central moments are:

$$\begin{aligned}
\kappa_1 &= \mu_1' \\
\kappa_2 &= \mu_2' - \mu_1'^2 = \mu_2 \\
\kappa_3 &= \mu_3' - 3\mu_2\mu_1' + 2\mu_1^3 = \mu_3 \\
\kappa_4 &= \mu_4' - 4\mu_3'\mu_1' - 3\mu_2'^2 + 12\mu_2'\mu_1 - 6\mu_1'^4 = \mu_4 - 3\mu_2^2
\end{aligned}$$

Note that the first cumulant equals the first *raw* moment, whereas the second and third cumulant are equal to the second and third *central* moment. The skewness and kurtosis can be defined in terms of cumulants: skewness is $\gamma_1 = \frac{\kappa_3}{(\kappa_2)^{\frac{3}{2}}}$; excess kurtosis is $\gamma_2 = \frac{\kappa_4}{\kappa_2^2}$.

Like joint moments, a product of random variables can be associated with a set of mixed cumulants, giving information about the joint distribution.

Definition 2.4.8. *The mixed cumulant* $\kappa_{i_1,\dots,i_n}$ *of order* l *of the* n*-dimensional random variable* $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_n)$ *with moment generating function* $M_{\mathbf{X}}$ *is defined as:*

$$\kappa_{i_1,\dots,i_n} = \log\left(E[\mathbf{X}_1^{i_1} \dots \mathbf{X}_n^{i_n}]\right) = \frac{\partial^{i_1+\dots+i_n} \log(M_{\mathbf{X}})}{\partial t_i^{i_1} \dots \partial t_n^{i_n}}(0, \dots, 0)$$

The order l *equals the sum of the powers:* $\sum_{s=1}^{n} i_s = l$*.*

Strictly mixed cumulants of independent random variables are zero, where strictly means that at least two random variables are included.

Lemma 2.4.9. *Let* $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_n)$ *such that* $f_{\mathbf{X}}(x_1, \dots, x_n) = f_{\mathbf{X}_1}(x_1) \dots f_{\mathbf{X}_n}(x_n)$*: i.e., the components are independent. Then* $\kappa_{i_1,\dots,i_n} = 0$ *if at least two indices* i_p, i_q *are nonzero.*

Proof. In case of complete independence among the components, $M_{\mathbf{X}}(t_1, \dots, t_n) = M_{\mathbf{X}_1}(t_1) \dots M_{\mathbf{X}_n}(t_n)$. To simplify the proof, consider the minimal case $(i_1, \dots, i_n) = (1, 1, 0, \dots, 0)$. Other instances are

analogous or can be reduced to this case.

$$\begin{aligned}
\kappa_{1,1,0,\ldots,0} &= \frac{\partial^2}{\partial t_1 \partial t_2} \log(M_{\mathbf{X}})(0,\ldots,0) \\
&= \frac{\partial^2}{\partial t_1 \partial t_2} \log(M_{\mathbf{X}_1} M_{\mathbf{X}_2} \ldots M_{\mathbf{X}_n})(0,\ldots,0) \\
&= \frac{\partial}{\partial t_2} \left(\frac{1}{M_{\mathbf{X}_1} M_{\mathbf{X}_2} \ldots M_{\mathbf{X}_n}} \frac{\partial}{\partial t_1} M_{\mathbf{X}_1} M_{\mathbf{X}_2} \ldots M_{\mathbf{X}_n} \right) (0,\ldots,0) \\
&= \frac{d}{dt_2} \left(\frac{1}{M_{\mathbf{X}_1}} \frac{d}{dt_1} M_{\mathbf{X}_1} \right) (0,\ldots,0) \\
&= 0
\end{aligned}$$

□

Finally, of special interest are Chebyshev-Hermite polynomials. These polynomials can be derived from Gaussian functions and have interesting orthogonality properties.

Definition 2.4.10. *(Chebyshev-Hermite polynomials)*
The Chebyshev-Hermite polynomial $h_i(x)$ is defined (e.g., in Kolassa [Kol94]) as:

$$h_i(x) = (-1)^i e^{\frac{x^2}{2}} \frac{d^i}{dx^i} (e^{-\frac{x^2}{2}}).$$

The first four Chebyshev-Hermite polynomials are explicitly given by:

$$\begin{array}{ll}
h_1(x) = x & h_3(x) = x^3 - 3x \\
h_2(x) = x^2 - 1 & h_4(x) = x^4 - 6x^2 + 3
\end{array}$$

The collection of Chebyshev-Hermite polynomials is orthogonal with respect to the standard normal probability distribution.

$$\int_{-\infty}^{\infty} h_i(x) h_j(x) \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = 0 \text{ for } i \neq j. \tag{2.18}$$

Chapter 3

Differential Power Analysis

In a quest for methods to attack cryptographic implementations, adversaries focus on directly accessible interfaces of the implementation, such as power consumption or electromagnetic radiation. An interface providing information about internal processes is referred to as a *side channel*. Side channel analysis (SCA) is a non-invasive, passive attack on a cryptographic device aiming at revealing information about the cipher key, which is embedded inside the device.

Differential power analysis (DPA) is a type of side channel analysis, focussing on power consumption. Due to the non-invasive nature of the attack, no permanent changes to the device are made, leaving no sign of an attack to a future user. Moreover, these attacks do not require sophisticated equipment, compared to invasive attacks. However, acquisition of power consumption measurements does require specified hardware which is not widely available.

The concept of differential power analysis was first introduced by Kocher et al. [KJJ99]. Power consumption of a cryptographic device partly depends on the data it processes. For example, transferring data from memory to a CPU consumes an amount of power. This data passes registers and buses, where every bit flip requires a small amount of power. Moreover, at transistor-level power is consumed for processing bits. Since data depends on (parts of) the key, and power consumption (partly) depends on data, information about the key might be obtained from examining power consumption.

In this chapter, an introduction to differential power analysis is provided. Furthermore, multiple ways to predict power consumption are treated. Correlation power analysis, which is a type of DPA, is explained thereafter.

3.1 General Differential Power Analysis

A general description of differential power analysis is provided in this section. Mutual information analysis, the central topic of this thesis, is an instance of DPA; mutual information analysis is thoroughly discussed in Chapter 4.

3.1.1 Description of Analysis

In a cryptographic algorithm such as AES (Appendix A.2), multiple intermediate states are sequentially updated until the final ciphertext is computed. Intermediate states depend on both plaintext and cipher key. More specifically, an intermediate state generally depends on a small number of plaintext bits and a small number of key bits (a subkey). Power consumption of a cryptographic device partly depends on the data processed, hence power consumption incorporates information about intermediate states, as explained before.

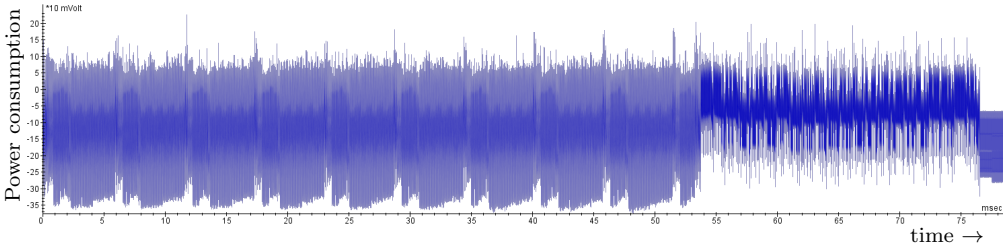


Figure 3.1: Example of a power trace.

If ciphertext is known, but plaintext is unknown, a known-ciphertext attack can equivalently be performed. Especially in symmetric cryptographic algorithms (where encryption and decryption depend on the same key) these attacks are as powerful as a known-plaintext attack. However, in the remainder of this thesis, only known-plaintext attacks are considered.

Power consumption of a smart card performing cryptographic operations is typically measured by a (digital) oscilloscope. The collection of measurements from one cryptographic run is called a power trace. Each plaintext processed by a cryptographic device gives rise to one power trace. A graphical representation of a power trace is shown in Figure 3.1. This power trace is obtained from measuring power consumption of a cryptographic device, performing encryption via AES. One might clearly observe a repetitive pattern, which corresponds to the AES round executions.

Differential power analysis exploits information leakage via power consumption by comparing power consumption with adequate predictions. An attacker predicts the exact value of the intermediate state, based on the corresponding known plaintext bits and a subkey guess. Then, power consumption is modeled as function of the intermediate value. Prediction of power consumption is a rather involved technique. First of all, an attacker must have a clue about the type of cryptography that is embedded in a smart card. From this knowledge, appropriate intermediate values might be selected for prediction. Evidently, a good prediction model (or power model) is not always available, even when one knows how data is processed by the cryptographic device. Hence, the degree of successfulness of an attack depends on the adequacy of the power model. Practical realizations of power models are introduced in Section 3.2.

After predicting power consumption, an attacker compares actual power consumption with predicted power consumption and accordingly estimates a likelihood for each subkey. The most likely subkeys can be recomputed into a cipher key, this is the attacker's best guess. The likelihood is created by means of a *distinguisher*. This is a function that evaluates and quantifies a relation between predictions and power consumption observations, based on a certain model. Correlation power analysis, which is described in Section 3.3, uses correlation-based distinguishers; mutual information analysis (the main topic of this thesis) uses mutual information-based distinguishers.

Besides power consumption, electromagnetic radiation is another side channel appropriate for analysis. Electric current causes proportional electromagnetic radiation, hence analysis of electromagnetic radiation does not require any cryptographic adjustments compared to analysis of power consumption. However, there is a difference in acquisition of measurements. Since electric current might differ locally on a card, electromagnetic radiation might provide more information when measurements are taken from the exact location where cryptographic operations are performed. However, this location generally is unknown to the attacker. Moreover, acquisition of electromagnetic traces requires more sophisticated equipment than acquisition of power traces. Besides that, more advanced signal processing techniques are required in order to prepare electromagnetic traces for analysis.

3.1.2 Complexity of Key Retrieval

The complexity of differential power analysis, in comparison with a brute force attack, is explained by means of an example. Here, the device under attack performs AES-encryptions of plaintexts.

A regular AES key is 128 bits in size. It can be viewed as a concatenation of 16 bytes. A total of 2^{128} different keys exist. This number is in the order of 10^{38}, hence it would be infeasible to exhaustively search for the cipher key. Imagine that everyone in the world ($= 7 \cdot 10^9$ persons) would be able to test one billion ($= 10^9$) keys per second, it would still take 110 times the estimated age of the universe to have investigated every possible key. In about half of that time, one is expected to find the correct key.

In DPA, however, one attacks subkeys, i.e., one attacks one key byte at the time. In that way, the workload is reduced from 2^{128} to $16 \cdot 2^8 = 4096$ trials to obtain the complete key. To illustrate this principle, consider the four digits in a PIN. If one has to try each distinct PIN possibility, one would need $10^4 = 10000$ trials. Now suppose one can attack the PIN digit by digit. This would only require $4 \cdot 10 = 40$ trials: 10 trials for each digit, independent from the other digits. This 'divide and conquer' algorithm strongly reduces the number of attempts to find the correct key. It actually transforms the complexity of the problem from exponential to linear in the size of the key.

The S-box function in AES acts on one byte of the state, which in turn depends on one key byte and one (known) plaintext byte. Hence, attacking an S-box output allows the attacker to recover the complete key byte by byte. Moreover, any intermediate value in a cryptographic algorithm, that has a deterministic relation with a part of the cipher key, is adequate for attack. However, whether information about an intermediate value leaks through power consumption depends on the properties of the implementation. Multiple intermediate values are discussed in Section 3.2.

3.1.3 Analysis Overview

In a nutshell, differential power analysis consists of the following steps.

1. Obtain knowledge about the device, e.g., which cryptographic algorithm is implemented.
2. Select an intermediate value that depends on key and plaintext (or ciphertext), and that is likely to cause information leakage via power consumption.
3. Acquire power traces by processing known (random) plaintexts.
4. Select a prediction method to model information leakage of the intermediate value.
5. For each plaintext, predict power consumption based on information leakage of the intermediate value, according to power models and subkey guesses.
6. Select a distinguisher and quantify relations between observations and predictions.
7. Rank subkey candidates by means of the distinguisher's score. The subkey with the highest score is the adversary's best guess.

In this thesis, information from Step 1 is assumed to be known. Step 2 is explained in Section 3.2.1. The acquisition of power traces (Step 3) is not treated, since it is out of scope. Step 4 is explained in Section 3.2.2. Predicting leakage via power consumption (Step 5) is straight-forward once an intermediate value and a prediction method are selected.

The remainder of this thesis deals with mutual information-based distinguishers (Step 6); however, distinguishers based on correlation coefficients are used for comparison. Hence, the majority of this thesis is focused on Step 6. After rating every key guess, ranking candidates (Step 7) is a matter of reordering possibilities.

3.2 Prediction Methods for Power Consumption

One of the core steps in side channel analysis consists of adequately predicting intermediate values. This section gives an overview of multiple prediction methods. First, different possible intermediate values are presented and assessed. Next, different power consumption models are discussed and evaluated. An extensive treatment can be found in Mangard et al. [MOP07].

Prediction under key guess $\hat{k}$ is written as $\mathbf{H}_{\hat{k}}$. A realization of this random variable, arising from the prediction under key guess $\hat{k}$ and using random plaintext d_i, is denoted as $h_{d_i,\hat{k}}$. The way to obtain this prediction value is described in this section. Formally, $h_{d_i,\hat{k}}$ is modeled in the following way:

$$h_{d_i,\hat{k}} = \phi(x(d_i, \hat{k})).$$

The function x defines an intermediate value, this is explained in Section 3.2.1. The function ϕ models the information leakage arising from the selected intermediate value. The realization of this function is discussed in Section 3.2.2.

3.2.1 Intermediate Values

When attacking a cryptographic implementation, one specifically aims at recovering certain intermediate values. When the correct intermediate values are recovered, the cipher key can be determined and the system is broken. The intermediate value under attack should be chosen such that its behavior can physically be observed. In other words, an adversary must be able to measure power consumption caused by the target intermediate value.

A possible value that can be considered is the output. This is no intermediate value, but it does depend on the cipher key. However, there is a very complex relation between the cipher key and the output. As observed by Shannon [Sha49], confusion is a necessary property of a secure cryptographic system. This forces an attacker to aim at intermediate values that have a less complex relation with the cipher key.

A cryptographic algorithm such as DES or AES typically operates in rounds. During each round, a relatively simple procedure is performed, using the output of the previous round and a round key. The round key is derived from the cipher key by means of a key schedule. In each round, the relation between input, key and intermediate values increases in complexity.

To attack an intermediate value during a round, one needs to know either the input or the output for that round. In the latter case, one can perform a reverse analysis, or attack the next round. However, the input of each round depends on the output of the previous round. Only the first round is independent of any other, since it has no predecessor. Its input can be calculated easily; for example, in AES this is the bitwise XOR of plaintext and cipher key. Analogously, the last round can be attacked with a reverse analysis in a known-ciphertext attack, since one knows the output of the last round: this is the ciphertext. Since information leakage from the first round and from the last round are equivalent, the first and last round are candidates for a first attack. However, analysis of multiple rounds might be required when not all key bytes can be compromised in one round. Ergo, where to start the attack depends on what is known to the attacker. In this thesis, known-plaintext attacks are presented, so only the first round is attacked.

Depending on the cryptographic system (viz. AES or DES), different intermediate values are eligible for an attack. The selected intermediate value is denoted by x. Specifically, the intermediate value arising from processing plaintext d_i under key k is denoted by $x(d_i, k)$. In this section, various intermediate values are discussed and their suitability is assessed.

AES Intermediate Values

The following intermediate values appear in the AES algorithm. Depending on the implementation of AES on a smart card, information from one or more intermediate values might leak through power consumption.

S-box output

The AES S-box output is a widely used intermediate value for attack. An S-box operates on byte-level and depends on only one round key byte and one state byte. Since the first round key equals the cipher key, a first round attack is even more easy than any other round. In case leakage from the output of S-boxes occurs, this intermediate must be attacked.

One can also focus at one ore more specific bits in the S-box output, in case some bits are more vulnerable for leakage, or noise. When one has obtained sufficient knowledge about some bits, the missing bits can be compromised exhaustively.

S-box output is likely to be reflected in power consumption, since the output of an S-box is written to a certain location in a register, or data bus. This writing process consumes power. In case the register is empty before information is written, power consumption can be related to the value written to the register.

S-box input and output

In some hardware implementations, S-box output is written to the same register (or data bus) as where the input comes from. This might cause information leakage; however, this leakage is related to both input and output. More specific, it is related to the XOR of S-box input and output. In that case, one can use these intermediates in a prediction model for leakage. Specifically, an attacker can predict the XOR of input and output, to indicate the difference between input and output. Then, power consumption prediction is related to the difference of input and output.

DES Intermediate Values

The following intermediate values appear in the DES algorithm. From which intermediate values information might leak through power consumption, is related to physical properties of the implementation of DES on a smart card.

S-box output

In DES, S-box output is a widely used intermediate value for attack, as in AES. However, whereas AES has only one type of S-box, there are 8 different S-boxes in the DES algorithm. It might happen that one S-box is more vulnerable for leakage or noise than another. This can be due to the design of the S-box (algorithmic noise), or to different implementations in the chip. The latter case is unpredictable for an attacker and must therefore be discarded in prediction power consumption. However, in that case an attacker can perform attacks on one output bit, instead of four output bits jointly. This reduces the uncertainty arising from joint behavior of four in differently implemented bits.

Round output

In some hardware implementations, one DES round is performed during one clock cycle. This means that the complete round execution is performed in parallel, and hence no leakage on individual S-boxes occurs. In that case, the output of a round might cause leakage via power consumption. Therefore, predicting round output bits descending from one S-box is useful when attacking the DES round key.

Here, round output means the left half of the complete round output, which is in turn equal to the right half of the next round input. Information about this intermediate value is likely to leak since information about the output is written to a register, or data bus, from where the next round input is read. Writing information to a register consumes an amount of power, that is related to the actual written value.

Round input and output

In case the round output is directly written to the location of the round input, an attacker might obtain leakage depending on the XOR of input and output. Then, these intermediates can be used to predict information leakage. An attacker can compute the XOR of round

input and round output, (cf. AES S-box input and output in the previous section) and relate power consumption to the computed difference of these two values.

3.2.2 Leakage Models

As explained in Section 3.1, DPA uses prediction models for intermediate values. Suitable prediction models yield more efficient attacks, hence multiple prediction models exist; each applicable in some situation. A leakage model , sometimes referred to a power model, is a function that assigns a power consumption prediction value to a realization of an intermediate value which is likely to cause information leakage. In this section, three leakage prediction models are explained.

Recall that power consumption predictions are solely based on the value of the corresponding intermediate value. Thus, power consumption is assumed to be independent from time or from other intermediate values. However, noise from parallel processes might occur in practice.

Correctly modeling power consumption is a difficult task, since an attacker generally has limited knowledge about the implementation. As pointed out in Mangard et al. [MOP07], the extent to which an attacker is able to adequately model power consumption is a major factor in the success of an attack.

Hamming Weight Model

The Hamming weight (HW) of an element in $\{0,1\}^*$ is equal to the number of ones. For example, the Hamming weight of the string 00100101 is 3. In general, the Hamming weight of a string is defined as the number of nonzero entries. Nonzero here means different from the zero-element of the alphabet over which the string is defined.

Let $\mathcal{S} = \{0,1\}^n$. Define the function $HW : \mathcal{S} \to \{0,1,\ldots,n\}$ as follows: for $s \in \mathcal{S}$: $s \mapsto HW(s) :=$ the number of ones in s, i.e., the Hamming weight of s.

A leakage model can be based on the Hamming weight model in the following way. Denote $h_{i,\hat{k}}$ for the prediction of power consumption of processing plaintext d_i under key guess $\hat{k}$. The Hamming weight model defines $h_{i,\hat{k}}$ as follows:

$$h_{i,\hat{k}} = f(HW(x(d_i,\hat{k}))),$$

where $f(t)$ is usually assumed to be an affine function, i.e., $f(t) = ct + w$. Here the values $c, w \in \mathbb{R}$ are power consumption constants. The value $x(d_i,\hat{k})$ represents the intermediate value depending on plaintext d_i and key $\hat{k}$, as introduced in Section 3.2.1.

Basically, the Hamming weight model indicates the number of bits that flip from zero to one. The power consumption constant c represents the power consumption of one such bit flip. For simplicity and since only relative leakage matters, one can use $c = 1$ and $w = 0$ to obtain $h_{i,\hat{k}} = HW(x(d_i,\hat{k}))$. Here, it is assumed that bits have equal contribution to leakage.

A Hamming weight model is appropriate in case information is written to a register that has been reset to the all-zero state. Power consumption is then related to the number of transitions from zeros to ones in the intermediate value.

Hamming Distance Model

The Hamming distance (HD) between two strings of zeros and ones is equal to the number of unequal entries. Hence, the Hamming distance between 00111001 and 00100101 is 3. The Hamming distance between two strings is equal to the Hamming weight of their difference, i.e.,

$$HD(v_1, v_2) = HW(v_1 \oplus v_2).$$

For example:

$$HD(00111001, 00100101) = HW(00111001 \oplus 00100101) = HW(00011100) = 3.$$

Like Hamming weight, a leakage model can be based on the Hamming distance model by evaluating the distance of two intermediate values x_1 and x_2, both descending from the same plaintext and key guess:

$$h_{i,\hat{k}} = f(HD(x_1(d_i, \hat{k}), x_2(d_i, \hat{k}))).$$

The function f is usually assumed to be affine, i.e., $f(t) = ct + w$. Usually $c = 1, w = 0$ is chosen, since only relative power consumption matters. Furthermore, it is assumed that 0-1 flips and 1-0 flips contribute equally to power consumption. A Hamming distance model is appropriate in case information is written to a register that has *not* been reset. This means that the previous state is overwritten by the new state. The intermediate values are in this case two consecutive states.

In their founding DPA manuscript, Kocher et al. [KJJ99] already noted that consumption often depends on the number of transitions (viz. Hamming distance) or the Hamming weight of specific values.

Identity Model

The identity model (ID) relaxes any other prediction method. Instead of mapping an intermediate value to, for example, its Hamming weight, the identity model assigns a unique value for each distinct realization of an intermediate value:

$$h_{i,\hat{k}} = f(x(d_i, \hat{k})).$$

Choosing $f(t) = t$ implies that $h_{i,\hat{k}} = x(d_i, \hat{k})$.

In practice it means that, for example, predictions on an DES S-box output range from 0 to 15, whereas Hamming weight based predictions range from 0 to 4. Hence, more information about the intermediate is included in identity-based predictions. An identity model is appropriate in case the leakage is not (completely) related to Hamming weight.

3.3 Correlation Power Analysis

Correlation power analysis (CPA) is an instance of differential power analysis. Every subkey must be assigned a score indicating the likelihood of being correct. In CPA, this scoring is done by calculating (the absolute value of) the correlation coefficient of predictions and observations.

First order CPA aims at unprotected implementations. In contrast, higher order CPA is designed to attack protected implementations. This section treats first order and second order CPA. Furthermore, the technique of preprocessing in second order CPA is explained.

3.3.1 First Order Correlation Power Analysis

Correlation power analysis is a type of differential power analysis, using Pearson's correlation coefficient as side channel distinguisher. That means that correlation coefficients of predictions and observations are estimated. The key guess giving rise to the highest absolute correlation coefficient, is the adversary's best guess.

The correlation coefficient of two random variables $\mathbf{X}$ and $\mathbf{Y}$ having arbitrary distributions with respective means $\mu_{\mathbf{X}}$ and $\mu_{\mathbf{Y}}$ and standard deviations $\sigma_{\mathbf{X}}$ and $\sigma_{\mathbf{Y}}$ is given by:

$$\rho(\mathbf{X}, \mathbf{Y}) = \frac{E[(\mathbf{X} - \mu_{\mathbf{X}})(\mathbf{Y} - \mu_{\mathbf{Y}})]}{\sigma_{\mathbf{X}}\sigma_{\mathbf{Y}}}. \tag{3.1}$$

Correlation coefficients quantify to what extend a relation is linear. This implies that, in correlation analysis, an attacker assumes a linear relation between predictions and observations. A nearly linear relation yields a (absolute) correlation coefficient close to one, whereas the lack of an obvious linear relation can result in a lower score. Hence, in case the assumption of linear dependence holds, correlation analysis is a powerful tool to reveal secret keys. A preliminary description of correlation coefficients is given in Section 2.3.

Assuming linearity is not just an assumption to simplify a model; it is valid in many situations. For example, consider an artificial intermediate value of 8 bits and a register to which this value is written. Suppose that for every register bit, a small amount of power is consumed when a 0 is transformed into a 1. Then, power consumption is linearly related to the number of ones in a bit string (the Hamming weight). When an adversary models power consumption based on the Hamming weight of intermediate values, these predictions have a linear relation with power consumption. Then, the key guess that yields the highest correlation coefficients is the one most likely to be correct.

However, this linear model does not always apply. That is where more generic dependence measurement comes in. Mutual information analysis relaxes the assumption of linear relations between prediction and observation to *any* relation. Advantages and disadvantages of this relaxation are assessed in this thesis, theoretically in Section 4.5.2 and practically in Chapter 7.

3.3.2 Second Order Correlation Power Analysis

In some instances, it might be useful to jointly examine multiple leakages. For example, a second order attack (as treated in Chapter 5) focuses on two leakage observations and one prediction value. These attacks make use of information coming from measurements at two moments in time, assuming that power consumption at these moments is due to processing two values that have a deterministic relation in the cryptographic algorithm. In this way, more information is exploited for attack.

Second order mutual information attacks are thoroughly explained in Chapter 5. Here, a brief overview of second order correlation analysis is provided. Since correlation coefficients are only defined for two random variables, modifications are required to remain applicable as distinguisher.

One widely used method is to map the two instantaneous leakages to one variable by means of a combining function, so-called preprocessing. The result of this combining function is tested for correlation with predictions; a relatively high correlation coefficient indicates a relative likely key guess.

Multiple combining functions (preprocessors) are proposed in literature (for example in Prouff et al. [PRB09] or in Mangard et al. [MOP07]). Two are introduced here: the normalized product and the absolute difference.

Definition 3.3.1. *(Normalized product)*
The normalized product of two random variables $\mathbf{X}$ *and* $\mathbf{Y}$ *is defined as:*

$$c(\mathbf{X}, \mathbf{Y}) = (\mathbf{X} - E[\mathbf{X}])(\mathbf{Y} - E[\mathbf{Y}]).$$

The normalized product of two intermediate values represents the product of deviations of two random variables to their respective means. This method is applicable for predicting difference between intermediate values, if this difference is reflected in the product of deviations of power consumption from their respective means.

Definition 3.3.2. *(Absolute difference)*
The absolute difference of two random variables $\mathbf{X}$ *and* $\mathbf{Y}$ *is defined as:*

$$d(\mathbf{X}, \mathbf{Y}) = |\mathbf{X} - \mathbf{Y}|.$$

The absolute difference of power values might be applicable to predict differences between intermediate values, assuming that difference between two intermediate values is linearly reflected in the difference between power consumption. This combining function is often used in correlation attacks, due to its linear nature. Magnard et al. [MOP07] indicate that the absolute difference preprocessor is adequate on devices that leak information about Hamming weight.

Preprocessing functions have several drawbacks, one of which is the amplification of noise. Moreover, a linear relation between predictions and preprocessed values must exist for a correlation attack to succeed. Consequently, the choice of a combining function might affect the strength of an attack.

Chapter 4

First Order Mutual Information Analysis

In this chapter, mutual information analysis (MIA) is explained. Mutual information analysis is an instance of differential power analysis, as described in Section 3.1. Thus, mutual information analysis is a tool to compromise cipher keys from cryptographic devices. To which extent this tool is powerful, and in which scenarios this tool is beneficial, is assessed in this work. A focus on the applicability of mutual information analysis on real smart cards, as opposed to simulated devices, is especially reflected in design decisions and results. However, the theoretical model as explained in this chapter holds for both simulations and practical embeddings.

The concept of mutual information analysis was introduced by Gierlichs et al. in 2008 [GBTP08], who characterized this method as being a generic side channel distinguisher capable of effectively detecting both linear and nonlinear dependence between prediction and observation of information leakage. Mutual information analysis is concerned with finding the best combination of key guess $\hat{k}$ and time point t that maximizes dependence between two random variables: prediction $\mathbf{H}_{\hat{k}}$ and observation $\mathbf{O}_t$ (power consumption). For the best combination, predictions under key guess $\hat{k}$ show the most similarity to observations at time t. Hence, $\hat{k}$ is the one most likely to be correct. The dependence among the random variables is measured by means of the mutual information of $\mathbf{H}_{\hat{k}}$ and $\mathbf{O}_t$.

Univariate or first order MIA treats each time point independently and henceforth can not benefit from any time-relation. However, mutual information analysis over a collection of time points jointly is also possible. This multivariate or higher order case is described in Chapter 5. In this chapter, mutual information analysis is first explained conceptually, followed by a brief summary on the state of the art. Then, theoretical foundations are explained, as well as multiple ways to estimate mutual information. This chapter is concluded by a theoretical comparison of MIA and CPA, where benefits and drawbacks of both methods are assessed.

4.1 Concept

Mutual information analysis aims at finding any dependence among random variables. As described in the previous chapter, mutual information of two variables $\mathbf{H}_{\hat{k}}$ and $\mathbf{O}_t$ can be calculated by evaluating a difference of entropies:

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) = H(\mathbf{O}_t) - H(\mathbf{O}_t|\mathbf{H}_{\hat{k}}). \tag{4.1}$$

Since entropy is a measure of information, or equivalently, a measure of uncertainty, the right-hand side of Equation 4.1 represents the difference between the uncertainty one has about observations and the uncertainty one has about the same observations, conditioned on a prediction value.

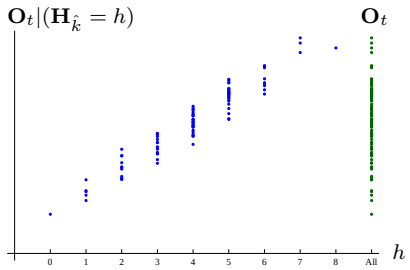


Figure 4.1: Observations for correct key guess in case of linear leakage.

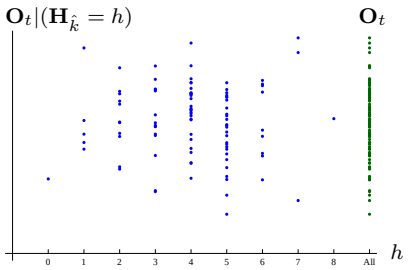


Figure 4.2: Observations for wrong key guess in case of linear leakage.

The random variable $\mathbf{H}_{\hat{k}}$ represents power consumption prediction of an intermediate value, under key guess $\hat{k}$, modeled by means of a leakage model (or power model). Intermediate values and power models are introduced in Section 3.2; in this chapter, prediction is treated as an abstract concept: an attacker makes *some* prediction. Power consumption at time t is denoted by $\mathbf{O}_t$; information about intermediate values is assumed to leak via this power consumption. Therefore, power consumption observations and information leakage are used as analogues.

When examining mutual information at one time point t, entropy $H(\mathbf{O}_t)$ is a fixed value independent from $\hat{k}$. Hence, maximizing mutual information for a fixed t reduces to minimizing $H(\mathbf{O}_t|\mathbf{H}_{\hat{k}})$ over all $\hat{k}$. This translates to the idea that mutual information is maximal in case one obtains few information from observations, once a prediction is known. In other words, when one knows the prediction value, the observation must be no great surprise.

Note that no assumption is made about *how* the power consumption should be related to predictions. There is, for example, no need for a linear relationship with predictions, nor does it need to depend on Hamming weight; as long as there is *any* relationship, mutual information has power to find it.

The benefit of relaxing the linear assumption is illustrated by means of an artificial example in Figures 4.1 to 4.4. Suppose storing an 8-bit AES S-box output value consumes an amount of power that has an affine relation with the Hamming weight of the S-box output, plus a small noise component. In CPA, an attacker predicts the Hamming weight of the S-box output and estimates correlation of predictions and observations. The observations can be categorized in 9 groups, $h = 0, \ldots, h = 8$, corresponding to the predicted value. For the correct key guess, observation are likely to be categorized as in Figure 4.1 (in blue). For an incorrect key guess, the observations might be categorized as in Figure 4.2 (in blue). In both figures, the unconditioned observations are displayed at the right side of the figure, in green.

The correlation coefficient of observation and prediction is around 0.95 for the correct key guess, and around 0.03 for a wrong key guess. Hence, prediction and observation are strongly correlated, and Pearson's correlation coefficient is a sufficiently good distinguisher.

Mutual information focuses at uncertainty about observations, given predictions. As can be seen in Figure 4.1, predictions in a certain category are concentrated in a subdomain of the overall observation domain. Hence, conditioning on a prediction value strongly reduces uncertainty about observations. In Figure 4.2, observations are not as nicely concentrated. There, more uncertainty remains when one conditions on a prediction value.

Now suppose a different implementation exists where power consumption has an equivalent, but not affine, relation with the Hamming weight of S-box output values. Particularly, an attack on this implementation yields observations categorized as in Figure 4.3 for the correct key guess. For a wrong key guess, observations can be categorized as in Figure 4.4.

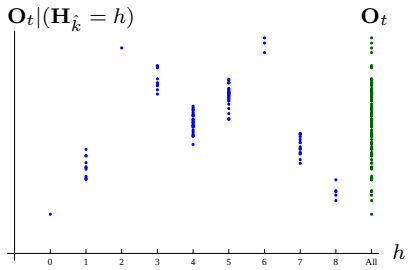


Figure 4.3: Observations for correct key guess in case of nonlinear leakage.

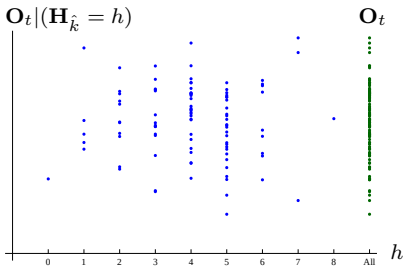


Figure 4.4: Observations for wrong key guess in case of nonlinear leakage.

Since the dependence of power consumption and Hamming weight of prediction is no longer linear, the correlation coefficient is close to zero for every key guess, including the correct one. Hence, correlation analysis will no longer be an adequate distinguisher. However, the behavior of observations conditioned on a prediction is equivalent to the behavior shown in Figure 4.1. Hence, mutual information is equally strong in both cases.

The behavior of observations and predictions has to be monitored simultaneously to distinguish correct and wrong key hypotheses. To determine mutual information of prediction and observation via, e.g., Equation 4.1, single and joint entropies of these random variables must be estimated. For estimation of joint entropy, one requires (empirical) joint probability distribution functions. Hence, the need for a joint distribution appears. This joint distribution can be obtained by keeping track of the (observation,prediction)-pairs $(o_{i,t}, h_{i,\hat{k}})$, both descending from the same plaintext d_i, for all i. In other words, one must keep track of where prediction and observation from each plaintext end up.

Multiple ways to estimate probability distributions from data exist, e.g., using histograms or kernels. In Section 4.4, two density estimation models are explained thoroughly; more methods are only introduced briefly. Density estimation is a crucial factor in the success of a mutual information-based attack. Therefore, the influence of density estimation is a widely investigated aspect in this thesis.

In many unprotected implementations, power consumption actually has a strongly linear relation with (the Hamming weight of) intermediate values. Since a bit flip consumes power, power consumption tends to be linearly related to the number of bit flips. Hence, a correlation attack is likely to be successful. Because no density estimation is required in CPA, it is likely that correlation analysis is more efficient in terms of number of traces required to compromise a key. However, in many protected implementations, intermediate values are randomized, providing no useful target for a correlation attack. Mutual information might gain better results under these circumstances.

4.2 State of the Art

Mutual information analysis was introduced by Gierlichs et al. [GBTP08] at CHES 2008. The authors explicitly baptized their concept *a generic information-theoretic side-channel distinguisher* to emphasize that MIA requires few assumptions, in relation to other side-channel analysis methods. Inspired by this paper, multiple researchers investigated into the efficiency of the proposed method when attacking different cryptographic embeddings. Moreover, several improvements and extensions are published; two notable studies among these are on higher order MIA (explained in the next chapter). In this section, the cutting edge of mutual information analysis is presented by means of contemporary papers on this topic.

The first paper on MIA, by Gierlichs et al. [GBTP08], introduces the concept of a mutual information-based distinguisher. The strong property of this distinguisher is that it is does not

require knowledge of the power consumption properties of the device under attack in advance. Where other SCA methods aim at finding the most adequate leakage model having linear relationship with power measurements, MIA focuses at the general joint statistical behavior of prediction and observation. The relaxation of assumptions induces a reduction in efficiency, which is quantized by comparison with CPA on an AES embedding. Due to the loss of efficiency, CPA yields higher success rates when compared to MIA, in case linearity assumptions hold. However, MIA is addressed to be particularly promising for attacks on protected devices, since it is not limited to linear dependence and can be extended to more dimensions easily.

Mutual information analysis inspired multiple authors to writing a contribution. Veyrat-Charvillon and Standaert [SVC09] reviewed the concept in their paper "Mutual information analysis: How, When and Why?". The notion of mutual information analysis as a toolbox in which different density estimation models and probability distance measures can be used was introduced. Adequate density estimation is appointed to be of crucial interest for MIA. A treatment of multiple density estimation techniques including histograms, kernel density estimation and nonparametric statistical tests such as Kolmogorov-Smirnov and Cramér-von-Mises is fulfilled; however, no method is considered universally favorite. The authors indicate that mutual information analysis could become an important tool in case an adversary is not able to obtain an adequate leakage profile: when it is unclear what to predict. Correlation tests fail in that case, whereas mutual information analysis still seems to be promising. Moreover, the understanding of mutual information analysis as an evaluation metric for side channel attacks is introduced and evaluated.

The essential issue in probability density estimation is described by Flament et al. [FGD+10]. Estimation errors are quantized and the consequences arising from this erroneous estimation in the analysis are investigated. Parametric estimation methods are considered prevalent; however, that requires knowledge, or at least evidence, of underlying distributions. Practically, this requirement is generally infeasible, particularly on high-end implementations.

Moradi et al. [MMPS09] extensively investigated mutual information analysis versus correlation analysis under a Gaussian model. They confirm a statement from Gierlichs et al. [GBTP08], saying that MIA performs worse in case of a near-linear leakage since it requires more measurements and computations to adequately model a density function. Moreover, mutual information analysis is argued to be less resistant to noise.

The benefits of density estimation using B-splines were presented by Venelli [Ven10]. Comparison with histogram-based density estimation and Kolmogorov-Smirnov tests yields an improvement in noise resistance. Whereas measurements near neighboring histogram bins can be assigned to the wrong bin due to noise, B-splines estimation aims at a continuous approximation of the underlying probability density function and is henceforth less affected by incorrectly assigned values. However, density estimation still requires more measurements that correlation tests, which is in favor of CPA. Moreover, estimating densities using B-splines is computationally more intensive than, e.g., estimating a histogram.

The first extensive elaboration on higher order mutual information analysis (HO-MIA) was published by Prouff and Rivain [PR09], targeting a masked implementation. A generalized HO-MIA attack is indicated to be much more efficient than similar correlation attacks on masked implementations; this statement is supported by results on simulations of a masked DES S-box.

Independently but simultaneously, Gierlichs et al. [GBPV10] studied higher order mutual information analysis. Their slightly different approach from Prouff and Rivain [PR09] drove the authors to enroll a comparison between both methods. Both methods endorse the quality of MI-based attacks on masked DES implementations, by pointing out the fact that, in contrast to higher order CPA, HO-MIA does not require a preprocessing step (see Section 3.3.2). Moreover, a mutual information-based measure is easily generalized to higher order cases. These two papers serve as motivation for further investigation of HO-MIA in Chapter 5.

Another approach combining both the density estimation issue and the applicability to higher order comes from Le and Berthier [LB10]. Mutual information estimation based on cumulants is proposed, typically feasible in case of near-Gaussian density models. Simulations for first order analysis on unprotected DES implementations and second order analysis on masked DES implementations support the already presented statement that HO-MIA is particularly promising. Cumulant-based probability densities might be determined incrementally, yielding a much more memory-efficient algorithm. However, since Gaussian distributions are assumed, this only covers the part where Gaussian assumptions hold.

Merging the results of multiple papers, Batina et al. [BGP+11] presented the state of the art in "Mutual information analysis: a Comprehensive Study". Different density estimation techniques are assessed, for both first order MIA and HO-MIA. The prevalence for CPA over MIA on first order DES attacks is affirmed; moreover, once again MIA was designated to be most adequate in a higher DES order attack.

Recently, Veyrat-Charvillon and Standaert contributed to mutual information analysis in their paper "Generic Side-Channel Distinguishers: Improvements and Limitations" [VCS11], by proposing an alternative approach to density estimation. This method, using a copula instead of joint distributions, is useful in template attacks; however, assumptions on the shape of the copula are made. Moreover, mutual differences between observations and predictions are calculated, which has quadratic complexity in the number of traces. While this method works efficiently on small sets, attacking a high-end card might be computationally demanding.

In almost all publications on mutual information analysis, emphasis is put on the generality of the concept, as well as on the significance of adequate density estimation. However, no general *best* decision has been indicated yet, yielding potential to further explore this topic.

4.3 Theoretical Foundation

Since mutual information analysis is a type of side channel analysis, more precisely an instance of differential power analysis, it suits the description of power analysis given in Section 3.1. In this section, variables and functions are assigned to the introduced terms. Moreover, the necessary calculation steps to perform MIA are explained.

Mutual information analysis aims at recovering the cipher key by means of key guesses. First, one needs to perform measurements and predict the outcome for different plaintexts. After having obtained the observations and performed the predictions, the data is analyzed in order to find the most likely key guess. One starts with constructing joint probability density functions of predictions and observations. At this moment, no distinction between a continuous or a discrete probability density function is made.

Estimating the underlying probability density function of a collection of data is a widely studied topic in probability theory. A broad range of techniques exist, each one with different advantages and disadvantages. Globally, there are two ways of fitting a distribution to a set of data. Either, one assumes a certain (mixture of) probability model(s) as underlying function and one estimates the parameter values, or one assumes no specific underlying distribution and creates a nonparametric probability density function. Multiple probability density function estimation techniques and their applications to MIA are discussed in Section 4.4.

In Section 4.3.2, the concept of mutual information analysis is further explained using abstract probability density functions, i.e., no assumption on their shape or behavior is made. In Section 4.4.1, probability density function estimation using histograms is explained, as well as applications of this method in a mutual information analysis. Section 4.4.2 describes mutual information analysis using cumulants for estimation. A brief overview of other density estimation techniques is given in Section 4.4.3.

4.3.1 Prediction and Observation

In this section, a theoretical approach on mutual information analysis is given. This serves as uniform framework for definition and notation in the remainder of this thesis.

Let $\mathcal{D}$ be the plaintext space. For example, $\mathcal{D} = \{0,1\}^8$ represents the space of all binary plaintexts of length 8, i.e., the collection of distinct bytes. One often requires a collection of q random plaintexts chosen uniformly from $\mathcal{D}$. These are denoted by $d_i \in \mathcal{D}, i = 1, \ldots, q$. Each d_i can be modeled as the realization of a random variable $\mathbf{D}$ uniformly distributed over $\mathcal{D}$.

In some cases a deliberate plaintext choice can be made in order to obtain sharper results. However, this only applies when an algorithm output shows strong dependence on (specific parts of) the plaintext. Since AES and DES, which both are in the scope of this thesis, in general do not show such dependencies due to the diffusion property (See Appendix A.2.2), no need for specific plaintext choices arises, as indicated by Mangard et al. [MOP07]. Therefore, in this thesis power traces are derived from a cryptographic device processing random plaintexts, which are uniformly distributed over the plaintext space.

Let $\mathcal{K}$ be the key space. For example, $\mathcal{K} = \{0,1\}^{128}$ represents the space of all 128-bits keys. When one wants to obtain n key guesses $\hat{k}_j \in \mathcal{K}, j = 1, \ldots, n$, randomly, each $\hat{k}_j$ can be modeled as the realization of a random variable $\mathbf{K}$ over $\mathcal{K}$.

However, a brute force key recovery using randomly chosen key guesses is computationally infeasible. Moreover, in many implementations only small parts of the cipher key (sometimes called subkeys) are used successively. Consequently, determining the key part by part could be a clever solution. Since this strongly reduces the number of different guesses, a brute force attack over a set of subkeys is computationally feasible. The complexity of side channel analysis is already explained is Section 3.3.

Focusing on parts of the key induces focusing on parts of processed states. A designated part of an intermediate state in the execution of a cryptographic algorithm is called an intermediate value . If an intermediate value depends on parts of both plaintext and cipherkey, this value can be exploited for attack. By retrieving information about the behavior of intermediate values, information about a part of the cipher key might appear. Adequate choices for the focus of an attack might actually reveal information about the complete cipher key.

Let $\mathcal{H}$ be the leakage prediction space. Calculating the outcome of an intermediate value that depends on a key guess $\hat{k}$ and a plaintext d and modeling power consumption based on the predicted value gives rise to a power consumption prediction $h \in \mathcal{H}$, also called the hypothetical leakage value. The prediction can be modeled as a random variable $\mathbf{H}$ over $\mathcal{H}$, since predictions are based on a randomly chosen plaintext. More information on prediction methods and intermediate values is presented in Section 3.2.

Let $\mathcal{O}$ be the observation space; observations represent power consumption values at certain time points. A power trace (see Figure 3.1) is a sequence of observations $o \in \mathcal{O}$. . Suppose each trace consists of l observations (measured a time points $t_1, \ldots, t_l$), then a trace can be seen as an element of $\mathcal{O}^l$. Each plaintext d_i gives rise to one power trace, the corresponding observations (i.e., power consumption values) are denoted by $o_{i,1}, \ldots, o_{i,l}$. Since observations depend on randomly chosen plaintexts, observation can be modeled as a random variable $\mathbf{O}$ over $\mathcal{O}$.

To attack a cryptographic implementation, one first draws q plaintexts uniformly random from $\mathcal{D}$. During the processing of these plaintexts, one measures power consumption (Step 3 from Section 3.1.3). This yields q traces, each consisting of l observations. Since l can be particularly large in practice, this process can be computationally intensive. In Section 6.3 multiple solutions to this practical problem are presented.

Calculations in the cryptographic device only involves the cipher key κ; observations are independent from key guesses. The observations of power consumption $o_{i,j}$ for plaintexts $d_i, i = 1, \ldots, q$, at times $t_j, j = 1, \ldots, l$ are:

$$O = \begin{pmatrix} o_{1,1} & \cdots & o_{1,l} \\ \vdots & & \vdots \\ o_{q,1} & \cdots & o_{q,l} \end{pmatrix}$$

Next, one makes q predictions for each key guess $\hat{k}_1, \ldots, \hat{k}_n$, using the plaintexts $d_1, \ldots, d_q$ (Step 5 from Section 3.1.3). These predictions concern power consumption at the point in time τ where the actual key guess and plaintext are processed by the device. This point in time might be unknown to the attacker. The predicted power consumption $h_{i,j}$ for plaintexts $d_i, i = 1, \ldots, q$ and key guess $\hat{k}_j, j = 1, \ldots, n$ are:

$$H = \begin{pmatrix} h_{1,1} & \cdots & h_{1,n} \\ \vdots & & \vdots \\ h_{q,1} & \cdots & h_{q,n} \end{pmatrix}$$

Ergo, one measures $o_{d,t,\kappa}$ and predicts $h_{d,\tau,k}$. When looking for a relation between the prediction and the observation for each plaintext, the columns of the matrices H and O must be compared. This is done using a comparison function $C : \mathcal{O}^q \times \mathcal{H}^q \to \mathbb{R}$. Using statistical techniques, one maximizes the relation $C(\mathbf{H}_{\hat{k}}, \mathbf{O}_t)$ over $\hat{k}, t$. When a maximum occurs, it is most likely that plaintexts are processed at time t under key $\hat{k}$. In correlation analysis, the function C represents Pearson's correlation coefficient. In this thesis, C represents mutual information.

4.3.2 Estimation of Mutual Information

In Definition 2.2.1, mutual information is characterized in terms of entropy:

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) = H(\mathbf{H}_{\hat{k}}) + H(\mathbf{O}_t) - H(\mathbf{H}_{\hat{k}}, \mathbf{O}_t).$$

This leads to the following expression in case of discrete probability mass functions:

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) = \sum_{h \in \mathcal{H}} \sum_{o \in \mathcal{O}} f_{\mathbf{H}_{\hat{k}}, \mathbf{O}_t}(h, o) \log \left(\frac{f_{\mathbf{H}_{\hat{k}}, \mathbf{O}_t}(h, o)}{f_{\mathbf{H}_{\hat{k}}}(h) \cdot f_{\mathbf{O}_t}(o)} \right).$$

When $\mathbf{H}_{\hat{k}}$ and $\mathbf{O}_t$ are continuous, one obtains:

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) = \int_{h \in \mathcal{H}} \int_{o \in \mathcal{O}} f_{\mathbf{H}_{\hat{k}}, \mathbf{O}_t}(h, o) \log \left(\frac{f_{\mathbf{H}_{\hat{k}}, \mathbf{O}_t}(h, o)}{f_{\mathbf{H}_{\hat{k}}}(h) \cdot f_{\mathbf{O}_t}(o)} \right) do \; dh.$$

For each key guess $\hat{k}$ and time instance t, mutual information $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t)$ is estimated. This yields the mutual information matrix M:

Definition 4.3.1 (Mutual Information matrix)**.**

$$M = \begin{pmatrix} I(\mathbf{H}_{\hat{k}_1}; \mathbf{O}_{t_1}) & \ldots & I(\mathbf{H}_{\hat{k}_1}; \mathbf{O}_{t_l}) \\ \vdots & & \vdots \\ I(\mathbf{H}_{\hat{k}_n}; \mathbf{O}_{t_1}) & \ldots & I(\mathbf{H}_{\hat{k}_n}; \mathbf{O}_{t_l}) \end{pmatrix}.$$

Suppose $I(\mathbf{H}_{\hat{k}_i}; \mathbf{O}_{t_j})$ is the largest value in this matrix. This means that predictions under key guess $\hat{k}_i$ and observations at time t_j show the largest dependence. Then key guess $\hat{k}_i$ and time point t_j are most likely to be correct. Hence, this is an adversary's best guess.

To arrive at this point, one has to estimate mutual information for every time point and subkey guess. For this estimation, knowledge about probability densities is required. In the next section, multiple models to estimate probability density functions and mutual information are explained.

4.4 Mutual Information Estimation Models

Estimating mutual information values generally requires knowledge of the joint probability distribution of prediction and observation. However, this distribution might be a priori unknown. Hence, the density has to be determined by means of measurements, i.e., empirically. Fitting a distribution to a collection of data is a well-studied problem in probability theory. To be adequate for mutual information analysis in practice, the density function estimation method must balance accuracy and computational complexity.

In this section, multiple ways of density estimation are discussed. Both accuracy and complexity are assessed, as well as properties regarding the mutual information analysis framework.

Mutual information is estimated by means of entropy. Entropy of one random variable is a nonnegative number, derived from a probability distribution. Hence, adequately modeling a distribution is of crucial interest for entropy to be representative. Often, entropy is estimated by integrating (continuous) or summing (discrete) probability functions. Especially integration of arbitrary functions might suffer from discretization errors and can be computationally intensive, hence estimating densities as accurate as possible might not yield the desired result within reasonable time.

4.4.1 Histogram Method

A computationally simple, yet accurate way of density estimation makes use of histograms. A histogram yields a discrete probability density function, i.e. a probability mass function. In this section, mutual information analysis using histogram-based density estimations is explained.

A histogram often is visualized as a bar chart. Essentially, a histogram is made by grouping adjacent observations. A group is usually called *bin* . Histogram density estimation is a non-parametric method in the sense that no underlying distribution of the data is assumed. However, one has to decide on the number of bins. When the number of bins is chosen too low, the variance of data is not sufficiently reflected and the resolution vanishes. When the number of bins is chosen too high, this can yield a scattered perception; one can not determine any 'shape' in the data. Moreover, errors can emerge for observations near a bin bound; they might be assigned wrongly. Appointing the number of bins is therefore a point of attention in the actual analysis and is explained in Section 6.1.3.

Theoretical Foundation

Let $\mathbf{X}$ be a one-dimensional random variable over a space $\mathcal{X}$. Suppose the distribution of $\mathbf{X}$ is a-priori unknown; it has to be determined empirically using histograms. The following procedure provides the empirical probability density function based on a histogram model.

- Decide on the number of bins u. Methods to decide on this number are treated in Section 6.1.3.

- Divide the space $\mathcal{X}$ into u bins. In this thesis, it is assumed that all bins are equally wide. Hence, each bin has size $|\mathcal{X}|/u$, where $|\mathcal{X}|$ represents the size (or interval length) of $\mathcal{X}$. One obtains
$$\mathcal{X} = \mathcal{X}_1 \uplus \cdots \uplus \mathcal{X}_u.$$
This means that $\mathcal{X}$ is partitioned into disjoint subsets (i.e. the bins) $\mathcal{X}_1, \ldots, \mathcal{X}_u$.

- Next, perform q observations $\theta_1, \ldots, \theta_q$ from $\mathbf{X}$. Since $\theta_i \in \mathcal{X} = \mathcal{X}_1 \uplus \cdots \uplus \mathcal{X}_u$ for all $i = 1, \ldots, q$, for each θ_i there is exactly one bin $\mathcal{X}_j$ such that $\theta_i \in \mathcal{X}_j$. In words: observation θ_i is in bin $\mathcal{X}_j$. Determine for every observation the bin to which it belongs. Define the function $F : \{1, \ldots, u\} \to \mathbb{N}$ as follows: $F(j)$ gives the number of observations in bin $\mathcal{X}_j$.

Note that

$$\sum_{j=1}^{u} F(j) = q.$$

- For $j = 1, \ldots, u$, define the probability p_j of an observation ending up in bin j as

$$p_j = \frac{F(j)}{q}.$$

This is well defined since

$$\sum_{j=1}^{u} p_j = \sum_{j=1}^{u} \frac{F(j)}{q} = \frac{\sum_{j=1}^{u} F(j)}{q} = 1.$$

When one denotes the empirical probability density function of $\mathbf{X}$ as $f_{\mathbf{X}}$, then $f_{\mathbf{X}}(j) = p_j$ for $j = 1, \ldots, u$.

In a mutual information analysis, the joint probability density function of prediction and observation must be estimated. Hence, the histogram method is used for two random variables at the same time. Extending the notion of histograms to more than two dimensions is straightforward and is discussed in Section 5.5.1. The theoretical foundation for two random variables is applied in the following model.

Model

In first order mutual information analysis, two random variables are used: prediction $\mathbf{H}_{\hat{k}}$ over $\mathcal{H}$ and observation $\mathbf{O}_t$ over $\mathcal{O}$. Mutual information is estimated every key guess and time point. However, assume for now that $\hat{k}$ and t are fixed. After explaining how to obtain mutual information from two random variables, this assumption is relaxed.

In the definition above, empirical probability density functions of prediction under key guess $\hat{k}$ and observation at time t are denoted by $f_{\mathbf{H}_{\hat{k}}}$ and $f_{\mathbf{O}_t}$ respectively. However, mutual information analysis aims at finding a relation between $\mathbf{H}_{\hat{k}}$ and $\mathbf{O}_t$, hence their joint distribution $f_{\mathbf{H}_{\hat{k}},\mathbf{O}_t}$ has also to be examined. The pair $(\mathbf{H}_{\hat{k}},\mathbf{O}_t)$ is modeled as a two-dimensional variable over $\mathcal{H} \times \mathcal{O}$.

First the number of bins u in the histogram for $f_{\mathbf{H}_{\hat{k}}}$, as well as v, the number of bins in the histogram for $f_{\mathbf{O}_t}$, have to be assessed. This implies that the two-dimensional histogram for $(\mathbf{H}_{\hat{k}},\mathbf{O}_t)$ consists of $u \times v$ bins.

It is assumed that the number of bins for observation is equal for all t, and that the number of bins for prediction is equal for all $\hat{k}$. This assumption is most general, since an attacker has no prior knowledge about the distribution of observation. Since predictions are based on randomly chosen plaintexts, the theoretical probability distribution should be equal for all $\hat{k}$.

After deciding on the number of bins, $\mathcal{H}$ can be partitioned into disjoint subsets:

$$\mathcal{H} = \mathcal{H}_1^{\hat{k}} \uplus \cdots \uplus \mathcal{H}_u^{\hat{k}},$$

where $\mathcal{H}_i^{\hat{k}}$ corresponds to the i-th bin under key guess $\hat{k}$. In the same way, for the observations one obtains

$$\mathcal{O} = \mathcal{O}_1^t \uplus \cdots \uplus \mathcal{O}_v^t,$$

where $\mathcal{O}_j^t$ corresponds to the j-th bin at time t.

The way an interval is partitioned into bins is ambiguous. One can choose to use equally size bins, or adapt the bin size to some given information. For example, when one wants more distinguishing properties in an a priori known part of the domain (when, e.g., one already has knowledge about the underlying distribution), one can choose to use smaller bins in that specific range.

In this thesis, only equally spaced bins are considered. Since in general underlying distributions of observations are unknown, no intelligent scheme to assign bins to specific subdomains exists. Furthermore, one does not have to use a weight function to correctly estimate entropy from such a histogram.

Formally, a joint distribution can be estimated from histograms in the following way. For each plaintext d, prediction $h_{d,\hat{k}}$ is assigned to a particular bin $\mathcal{H}_i^{\hat{k}}$ and observation $o_{d,t}$ is assigned to a particular bin $\mathcal{O}_j^t$. Recall that the time point t and key guess $\hat{k}$ are still fixed. The total of these assignments define the function $F(i,j)$ for every bin pair (i,j), indicating the number of plaintexts giving rise to a prediction in bin $\mathcal{H}_i^{\hat{k}}$ and observation in bin $\mathcal{O}_j^t$:

$$F(i,j) = |\{d \in \mathcal{D} : h_{d,\hat{k}} \in \mathcal{H}_i^{\hat{k}}, o_{d,t} \in \mathcal{O}_j^t\}|.$$

The corresponding empirical probabilities are given by

$$p_{i,j} = \frac{F(i,j)}{q}.$$

Now the joint probability density function is equal to

$$f_{\mathbf{H}_{\hat{k}},\mathbf{O}_t}(i,j) = \frac{F(i,j)}{q}, i = 1,\ldots,u; j = 1,\ldots,v.$$

To visualize the joint probability density function, the matrix $J_{\hat{k},t} \in \mathbb{Q}^{u\times v}$ is defined by giving the (i,j)-element of $J_{\hat{k},t}$ the value of $f_{\mathbf{H}_{\hat{k}},\mathbf{O}_t}(i,j)$:

$$J_{\hat{k},t} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,v} \\ \vdots & & \vdots \\ p_{u,1} & \cdots & p_{u,v} \end{pmatrix}.$$

This matrix has an important property: the row sums constitute the marginal probability density function of the prediction and the column sums yield the probability density function of the observation.

$$\sum_{j=1}^{v} p_{i,j} = \sum_{j=1}^{v} P(h_{d,\hat{k}} \in \mathcal{H}_i^{\hat{k}}, o_{d,t} \in \mathcal{O}_j^t) = P(h_{d,\hat{k}} \in \mathcal{H}_i^{\hat{k}}).$$

Here $P(x \in \mathcal{X})$ denotes the empirical probability of the event $x \in \mathcal{X}$. Analogously:

$$\sum_{i=1}^{u} p_{i,j} = \sum_{i=1}^{u} P(h_{d,\hat{k}} \in \mathcal{H}_i^{\hat{k}}, o_{d,t} \in \mathcal{O}_j^t) = P(o_{d,t} \in \mathcal{O}_j^t).$$

This is important: not only does the matrix J contain the values $f_{\mathbf{H}_{\hat{k}},\mathbf{O}_t}(i,j)$; moreover, from J one can derive the marginal distributions $f_{\mathbf{H}_{\hat{k}}}$ and $f_{\mathbf{O}_t}$, hence the values $f_{\mathbf{H}_{\hat{k}}}(i)$ and $f_{\mathbf{O}_t}(j)$.

$$J_{\hat{k},t} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,v} \\ \vdots & & \vdots \\ p_{u,1} & \cdots & p_{u,v} \\ \downarrow & & \downarrow \\ f_{\mathbf{O}_t}(1) & \cdots & f_{\mathbf{O}_t}(v) \end{pmatrix} \begin{matrix} \\ \\ \rightarrow f_{\mathbf{H}_{\hat{k}}}(1) \\ \vdots \\ \rightarrow f_{\mathbf{H}_{\hat{k}}}(u) \end{matrix}$$

With these values, mutual information of prediction under key guess $\hat{k}$ and observation at time t is estimated using Definition 2.2.3:

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) = \sum_{i=1}^{u}\sum_{j=1}^{v} f_{\mathbf{H}_{\hat{k}},\mathbf{O}_t}(i,j) \log\left(\frac{f_{\mathbf{H}_{\hat{k}},\mathbf{O}_t}(i,j)}{f_{\mathbf{H}_{\hat{k}}}(i)\cdot f_{\mathbf{O}_t}(j)}\right). \tag{4.2}$$

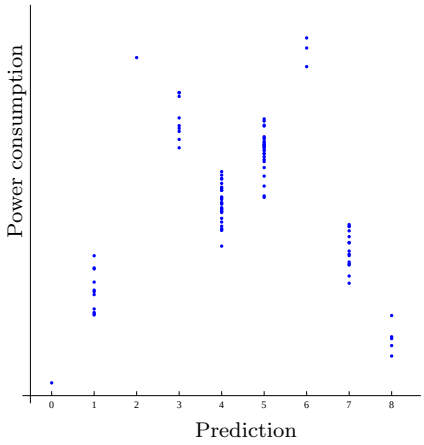


Figure 4.5: Power consumption observations classified by their prediction value.

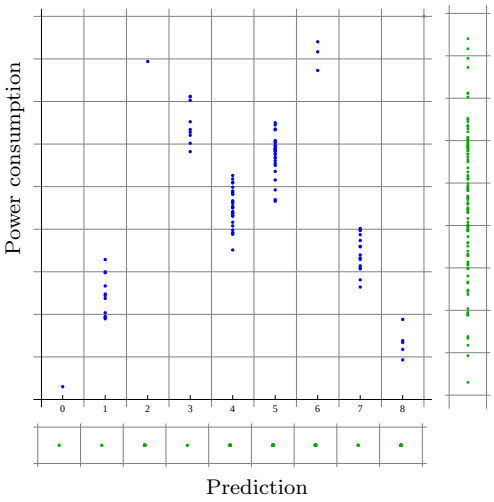


Figure 4.6: Prediction, observation and joint occurrence divided into bins.

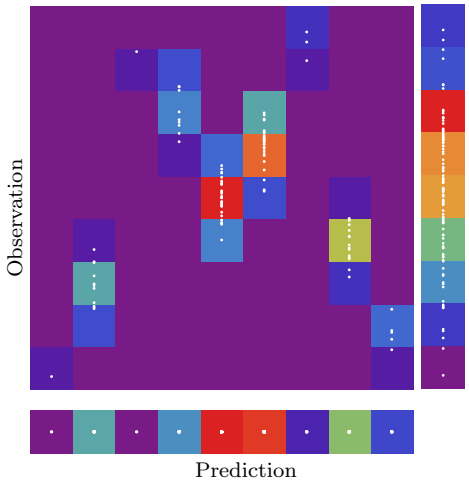


Figure 4.7: Prediction, observation and joint occurrence divided into bins.

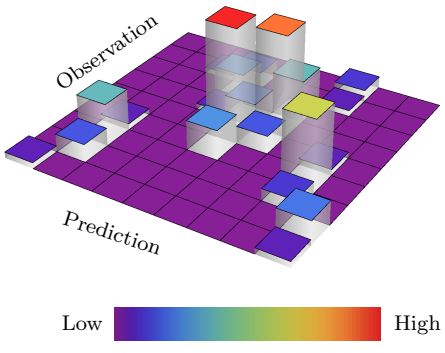


Figure 4.8: Visual representation of a two-dimensional histogram, with scale.

A visual interpretation of creating a histogram from a collection of (prediction,observation)-pairs is given in Figure 4.5 to Figure 4.8. In Figure 4.5, power consumption observations are grouped by corresponding prediction values, in this case Hamming weight of an AES S-box output. In Figure 4.6, a grid is drawn that corresponds to bins for prediction and observation. Figure 4.7 visualizes the frequency matrix F, Figure 4.8 is a three-dimensional representation of this frequency matrix.

A histogram has to be created for each $\hat{k}$ and t. This means that one computes the entries of the mutual information matrix M (from Definition 4.3.1).

$$M = \begin{pmatrix} I(\mathbf{H}_{\hat{k}_1}; \mathbf{O}_{t_1}) & \dots & I(\mathbf{H}_{\hat{k}_1}; \mathbf{O}_{t_l}) \\ \vdots & & \vdots \\ I(\mathbf{H}_{\hat{k}_n}; \mathbf{O}_{t_1}) & \dots & I(\mathbf{H}_{\hat{k}_n}; \mathbf{O}_{t_l}) \end{pmatrix}$$

Now the mutual information values are estimated, one can continue the analysis by making a key candidate ranking.

Evaluation

Histograms provide an estimate for an empirical probability density. Advantages and disadvantages of histograms are summarized in this section.

One advantage of histograms for random variables is that no underlying distribution is assumed. This contributes to its generality and wide applicability. In side channel analysis, the probability distribution of power consumption is generally unknown, hence a nonparametric method suits this scenario. Moreover, no complex calculations (such as integration) are involved, yielding a fast method to estimate a probability distribution.

In order to obtain a proper histogram, several conditions must be met. For example, since one needs to known the absolute bounds for observation, filtering for outliers must be executed before a histogram is constructed. Furthermore, an attacker must decide on the number of bins for prediction and observation. That these parameters have significant influence is motivated by results in Chapter 7. For example, using many bins might increase the vulnerability for noise, since a noisy observation might easily be assigned to a wrong bin. How to select the number of bins is discussed in Section 6.1.3.

In many papers on mutual information analysis (including Gierlichs et al. [GBTP08] and Prouff and Rivain [PR09]) histograms are used for density estimation. A disadvantage of histograms might be that no overall best bin decision can be made. Depending on the underlying distribution of observations, other density methods might be more appropriate; a histogram could show a relatively slow convergence towards the underlying distribution. Moreover, to adequately model a distribution via a histogram, more measurements are required than, for instance, by the calculation of a correlation coefficient. However, in case of nonlinear or unpredictable behavior, a histogram is a nonparametric and fast way to adequately model an empirical density.

4.4.2 Cumulant Method

Recently, mutual information analysis based on cumulants has been proposed by Le and Berthier [LB10]. Where most of the mutual information estimation methods involve density estimation, the cumulant method provides a way to directly estimate mutual information values from measurements. This is one of the main advantages of this method; not only does this reduce the computational complexity, it also avoids bin size decisions. However, a nearly Gaussian distribution of prediction and observation is assumed, hence this is a parametric method. In this section, the cumulant method is explained and evaluated.

Theoretical Foundation

Both prediction and observation are assumed to follow a probability distribution close to the normal distribution. This restricts prediction methods to only nearly normal distributions. Hamming weight of strings chosen uniformly random from $\mathcal{X} = \{0,1\}^n$ suffices this restriction, as proven here.

The number of n-tuples with k ones, viz. having Hamming weight k, is equal to $\binom{n}{k}$. Let $\mathbf{X}$ be a uniformly distributed random variable over $\mathcal{X}$. Since $|\mathcal{X}| = 2^n$, the probability distribution of $HW(\mathbf{X})$ is given by

$$P(HW(\mathbf{X}) = i) = \frac{1}{2^n}\binom{n}{i}.$$

Suppose $\mathbf{X} = (a_1, \ldots, a_n)$ is chosen uniformly random. Then each position a_i in the string is filled randomly and independently with a zero or a one, both with probability one half. Hence, a string with Hamming weight k is an outcome of a binomial experiment $B(n, k, \frac{1}{2})$. The binomial distribution has probability mass function

$$f(n,k,p) = \binom{n}{k} p^k (1-p)^{n-k},$$

for $k = 0, \ldots, n$, mean $\mu = np$ and variance $\sigma^2 = np(1-p)$. The case considered here ($p = 1/2$) yields a probability mass for $k = 0, \ldots, n$:

$$f(n,k,\frac{1}{2}) = \frac{1}{2^n}\binom{n}{i},$$

mean $\mu = n/2$ and variance $\sigma^2 = n/4$. Consequently, the Hamming weight of an 8-bit string (such as an AES S-box output) has mean 4 and variance 2.

An interesting property of the binomial distribution $\mathrm{Bin}(n,p)$ is that it converges towards the normal distribution $\mathcal{N}(np, np(1-p))$. Consequently, for large n the Hamming weight of a random string of length n is approximately normally distributed with mean $\mu = np = n/2$ and variance $\sigma^2 = np(1-p) = n/4$.

Recall from Equation (2.10) that the mutual information of two random variables $\mathbf{X}$ and $\mathbf{Y}$ can be calculated by means of the Kullback-Leibler divergence from their joint distribution to the product of their marginal distributions:

$$I(\mathbf{X};\mathbf{Y}) = D_{\mathrm{KL}}(f_{\mathbf{X},\mathbf{Y}} || f_{\mathbf{X}} f_{\mathbf{Y}}).$$

Denote the standard normal distribution by $\mathcal{N}(0,1)$ and the standard normal probability density function by:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

For a random variable that follows a nearly normal distribution, the *exact* distribution can be approximated by Edgeworth series of arbitrary order. Since the error of an o-th order Edgeworth approximation for q measurements is proportional to $q^{1-o/2}$ (Kolassa and McCullagh [KM90]), an approximation of order $o = 4$ is selected to balance computational cost and convergence.

Let $\mathbf{X}$ have a distribution close to the standard normal distribution. The fourth order Edgeworth approximation of the distribution of $\mathbf{X}$ is given by:

$$f_{\mathbf{X}}(x) \approx \Phi(x)\left(1 + \frac{k_2(\mathbf{X})-1}{2!}h_2(x) + \frac{k_3(\mathbf{X})}{3!}h_3(x) + \frac{k_4(\mathbf{X})}{4!}h_4(x)\right). \tag{4.3}$$

with i-th Chebyshev-Hermite polynomial $h_i(x)$ (Definition 2.4.10) and i-th order cumulant $k_i(\mathbf{X})$ of $\mathbf{X}$ as in Definition 2.4.7 for $i = 2, 3, 4$. For a full coverage of this series expansion, see e.g. McCullagh [McC87].

Lemma 4.4.1. *(Kullback-Leibler divergence approximation)*

Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$ *and* $\mathbf{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_n)$ *be two* n*-dimensional random variables with components that follow a nearly standard normal distribution. Define the following expressions for an* $\mathbf{X}$ *and analogously for* $\mathbf{Y}$*:*

$$\begin{aligned} R_{i,j}(\mathbf{X}) &= E[\mathbf{X}_i\mathbf{X}_j]; \\ T_{i,j,k}(\mathbf{X}) &= E[\mathbf{X}_i\mathbf{X}_j\mathbf{X}_k]; \\ Q_{i,j,k,l}(\mathbf{X}) &= E[\mathbf{X}_i\mathbf{X}_j\mathbf{X}_k\mathbf{X}_l] - R_{i,j}(\mathbf{X})R_{k,l}(\mathbf{X}) - R_{i,k}(\mathbf{X})R_{j,l}(\mathbf{X}) - R_{i,l}(\mathbf{X})R_{j,k}(\mathbf{X}). \end{aligned}$$

The fourth order approximation of the Kullback-Leibler divergence from $\mathbf{X}$ *to* $\mathbf{Y}$ *(as defined in Definition 2.1.10) is given by:*

$$\begin{aligned} D_{KL}(\mathbf{X}||\mathbf{Y}) \approx \frac{1}{4}\sum_{i,j=1}^{n}(R_{i,j}(\mathbf{X}) - R_{i,j}(\mathbf{Y}))^2 + \frac{1}{12}\sum_{i,j,k=1}^{n}(T_{i,j,k}(\mathbf{X}) - T_{i,j,k}(\mathbf{Y}))^2 \\ + \frac{1}{48}\sum_{i,j,k,l=1}^{n}(Q_{i,j,k,l}(\mathbf{X}) - Q_{i,j,k,l}(\mathbf{Y}))^2. \end{aligned} \tag{4.4}$$

For a proof, see, e.g., Le and Berthier, [LB10].

Using single random variables $\mathbf{X}$ and $\mathbf{Y}$, one obtains the regular Kullback-Leibler divergence.

Corollary 4.4.2. *The Kullback-Leibler divergence* $D_{KL}(\mathbf{X}||\mathbf{Y})$ *of one-dimensional random variables* $\mathbf{X}$ *and* $\mathbf{Y}$ *can be approximated by*

$$D_{KL}(\mathbf{X}||\mathbf{Y}) \approx \frac{(k_2(\mathbf{X}) - k_2(\mathbf{Y}))^2}{4} + \frac{(k_3(\mathbf{X}) - k_3(\mathbf{Y}))^2}{12} + \frac{(k_4(\mathbf{X}) - k_4(\mathbf{Y}))^2}{48}. \tag{4.5}$$

Using this result, mutual information values can be approximated by means of the Kullback Leibler approximation, introduced in Lemma 4.4.1. To simplify calculations, random variables are assumed to have a distribution that is close to the standard normal distribution $\mathcal{N}(0,1)$. Arbitrary variables that follow a near-normal distribution can be normalized in order to fulfill this assumption.

First, some notation is introduced. The cumulant method is theoretically equivalent in any dimension, a generalized definition involving n-dimensional random variables is given (cf. Definition 2.2.9). Estimation of mutual information of two random variables is only an instance of the following definitions.

For an n-dimensional random variable $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$, define the expressions analogous to those in Lemma 4.4.1:

$$\begin{aligned} R_{i,j}(\mathbf{X}) &= E[\mathbf{X}_i\mathbf{X}_j]; \\ T_{i,j,k}(\mathbf{X}) &= E[\mathbf{X}_i\mathbf{X}_j\mathbf{X}_k]; \\ Q_{i,j,k,l}(\mathbf{X}) &= E[\mathbf{X}_i\mathbf{X}_j\mathbf{X}_k\mathbf{X}_l] - R_{i,j}(\mathbf{X})R_{k,l}(\mathbf{X}) - R_{i,k}(\mathbf{X})R_{j,l}(\mathbf{X}) - R_{i,l}(\mathbf{X})R_{j,k}(\mathbf{X}). \end{aligned}$$

The values R, T and Q can be regarded as second, third and fourth order (mixed) cumulants.

Theorem 4.4.3. *(Mutual information approximation)*

Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$ *be an* n*-dimensional random variable, where each component follows a nearly standard normal distribution. The components are not independent in general. Write* $f_{\mathbf{X}}(x_1, \ldots, x_n)$ *as a shorter notation for the joint distribution* $f_{\mathbf{X}_1,\ldots,\mathbf{X}_n}(x_1, \ldots, x_n)$ *and denote* $\mathbf{Y} = \mathbf{X}_1 \cdot \ldots \cdot \mathbf{X}_n$*, so* $f_{\mathbf{Y}}(x_1, \ldots, x_n) = f_{\mathbf{X}_1}(x_1) \cdot \ldots \cdot f_{\mathbf{X}_n}(x_n)$*.*

Denote $W = \{1, \ldots, n\}$ and define the following index sets:

$$I_2 = W^2 \backslash \{(i,i) : i \in W\};$$
$$I_3 = W^3 \backslash \{(i,i,i) : i \in W\};$$
$$I_4 = W^4 \backslash \{(i,i,i,i) : i \in W\}.$$

The mutual information between the components of $\mathbf{X}$ can be estimated as follows.

$$I(\mathbf{X}_1; \ldots; \mathbf{X}_n) \approx \frac{1}{4} \sum_{i,j \in I_2} (R_{i,j}(\mathbf{X}))^2 + \frac{1}{12} \sum_{i,j,k \in I_3} (T_{i,j,k}(\mathbf{X}))^2 + \frac{1}{48} \sum_{i,j,k,l \in I_4} (Q_{i,j,k,l}(\mathbf{X}))^2. \quad (4.6)$$

Proof. The mutual information of the components of $\mathbf{X}$ can be determined by computing the Kullback-Leibler divergence from $f_{\mathbf{X}_1,\ldots,\mathbf{X}_n} = f_{\mathbf{X}}$ to $f_{\mathbf{X}_1} \cdot \ldots \cdot f_{\mathbf{X}_n} = f_{\mathbf{Y}}$. The result of Lemma 4.4.1 is used to approximate this Kullback-Leibler divergence.

$$D_{\mathrm{KL}}(f_{\mathbf{X}} || f_{\mathbf{Y}}) \approx \frac{1}{4} \sum_{i,j=1}^{n} (R_{i,j}(\mathbf{X}) - R_{i,j}(\mathbf{Y}))^2 + \frac{1}{12} \sum_{i,j,k=1}^{n} (T_{i,j,k}(\mathbf{X}) - T_{i,j,k}(\mathbf{Y}))^2 + \frac{1}{48} \sum_{i,j,k,l=1}^{n} (Q_{i,j,k,l}(\mathbf{X}) - Q_{i,j,k,l}(\mathbf{Y}))^2.$$

The mixed cumulants of independent components (viz. those of $\mathbf{Y}$) are equal to zero, according to Lemma 2.4.9. Hence, the only cumulants contributing to the divergence are the single cumulants. Using Kronecker's delta notation (which is equal to one if and only if all indices are identical and zero otherwise), the expression becomes:

$$D_{\mathrm{KL}}(f_{\mathbf{X}} || f_{\mathbf{Y}}) \approx \frac{1}{4} \sum_{i,j=1}^{n} (R_{i,j}(\mathbf{X}) - \delta_{i,j} R_{i,j}(\mathbf{Y}))^2 + \frac{1}{12} \sum_{i,j,k=1}^{n} (T_{i,j,k}(\mathbf{X}) - \delta_{i,j,k} T_{i,j,k}(\mathbf{Y}))^2 + \frac{1}{48} \sum_{i,j,k,l=1}^{n} (Q_{i,j,k,l}(\mathbf{X}) - \delta_{i,j,k,l} Q_{i,j,k,l}(\mathbf{Y}))^2. \quad (4.7)$$

In case of equal indices, $R_{i,i}(\mathbf{X}) = E[\mathbf{X}_i \mathbf{X}_i] = E[\mathbf{X}_i^2]$, which is the second cumulant of $\mathbf{X}_i$. In the same way one obtains $R_{i,i}(\mathbf{Y}) = E[\mathbf{X}_i \mathbf{X}_i] = E[\mathbf{X}_i^2]$. This means that

$$R_{i,j}(\mathbf{X}) - \delta_{i,j} R_{i,j}(\mathbf{Y}) = (1 - \delta_{i,j}) R_{i,j}(\mathbf{X}) = \begin{cases} 0 & \text{if } i = j; \\ R_{i,j}(\mathbf{X}) & \text{otherwise.} \end{cases}$$

Analogously one obtains that $T_{i,i,i}(\mathbf{X}) = T_{i,i,i}(\mathbf{Y}) = E[\mathbf{X}_i^3]$, hence

$$T_{i,j,k}(\mathbf{X}) - \delta_{i,j,k} T_{i,j,k}(\mathbf{Y}) = (1 - \delta_{i,j,k}) T_{i,j,k}(\mathbf{X}) = \begin{cases} 0 & \text{if } i = j = k; \\ T_{i,j,k}(\mathbf{X}) & \text{otherwise.} \end{cases}$$

Now, it might be no surprise that $Q_{i,i,i,i}(\mathbf{X}) = Q_{i,i,i,i}(\mathbf{Y}) = E[\mathbf{X}_i^4] - 3E[\mathbf{X}_i^2]^2$, hence

$$Q_{i,j,k,l}(\mathbf{X}) - \delta_{i,j,k,l} Q_{i,j,k,l}(\mathbf{Y}) = (1 - \delta_{i,j,k,l}) Q_{i,j,k,l}(\mathbf{X}) = \begin{cases} 0 & \text{if } i = j = k = l; \\ Q_{i,j,k,l}(\mathbf{X}) & \text{otherwise.} \end{cases}$$

Replacing this in Equation 4.7 yields:

$$D_{\mathrm{KL}}(f_{\mathbf{X}} || f_{\mathbf{Y}}) \approx \frac{1}{4} \sum_{i,j \in I_2} (R_{i,j}(\mathbf{X}))^2 + \frac{1}{12} \sum_{i,j,k \in I_3} (T_{i,j,k}(\mathbf{X}))^2 + \frac{1}{48} \sum_{i,j,k,l \in I_4} (Q_{i,j,k,l}(\mathbf{X}))^2. \quad (4.8)$$

Recalling that $I(\mathbf{X}_1; \ldots; \mathbf{X}_n) = D_{\mathrm{KL}}(f_{\mathbf{X}} || f_{\mathbf{Y}})$ yields the desired result. □

Model

A collection of q traces yields q predictions $h_{1,\hat{k}}, \ldots, h_{q,\hat{k}}$ under key guess $\hat{k}$, as well as q observations $o_{1,t}, \ldots, o_{q,t}$ at time t. Unbiased estimators for the sample mean and standard deviation are defined as:

$$m_{\mathbf{H}_{\hat{k}}} = \frac{1}{q}\sum_{i=1}^{q} h_{i,\hat{k}}, \quad s^2_{\mathbf{H}_{\hat{k}}} = \frac{1}{q-1}\left(\sum_{i=1}^{q} h^2_{i,\hat{k}} - (\sum_{i=1}^{q} h_{i,\hat{k}})^2\right);$$
$$m_{\mathbf{O}_t} = \frac{1}{q}\sum_{i=1}^{q} o_{i,t}, \quad s^2_{\mathbf{O}_t} = \frac{1}{q-1}\left(\sum_{i=1}^{q} o^2_{i,t} - (\sum_{i=1}^{q} o_{i,t})^2\right).$$

Denote the normalized prediction under key guess $\hat{k}$ by $\tilde{\mathbf{H}}_{\hat{k}}$ and the normalized observation at time t by $\tilde{\mathbf{O}}_t$, i.e.

$$\tilde{\mathbf{H}}_{\hat{k}} = \frac{\mathbf{H}_{\hat{k}} - m_{\mathbf{H}_{\hat{k}}}}{s_{\mathbf{H}_{\hat{k}}}}, \tilde{\mathbf{O}}_t = \frac{\mathbf{O}_t - m_{\mathbf{O}_t}}{s_{\mathbf{O}_t}}.$$

According to Lemma 2.2.10, $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) = I(\tilde{\mathbf{H}}_{\hat{k}}; \tilde{\mathbf{O}}_t)$ and hence

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) = D_{\mathrm{KL}}(f_{\tilde{\mathbf{H}}_{\hat{k}}, \tilde{\mathbf{O}}_t} || f_{\tilde{\mathbf{H}}_{\hat{k}}} f_{\tilde{\mathbf{O}}_t}).$$

In order to approximate $I(\tilde{\mathbf{H}}_{\hat{k}}; \tilde{\mathbf{O}}_t)$ using Theorem 4.4.3, the summands are taken over $(\mathbf{X}_1, \ldots, \mathbf{X}_n) = (\tilde{\mathbf{H}}_{\hat{k}}, \tilde{\mathbf{O}}_t)$. The cumulants R, T and Q can be expressed in terms of moments $\mu_{a,b}$ to obtain:

$$\begin{aligned}
\sum_{i,j\in I_2} (R_{i,j})^2 =& \binom{2}{1} E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t]^2 = 2\mu^2_{1,1};\\
\sum_{i,j,k\in I_3} (T_{i,j,k})^2 =& \binom{3}{1} E[\tilde{\mathbf{H}}^2_{\hat{k}}\tilde{\mathbf{O}}_t]^2 + \binom{3}{2} E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}^2_t]^2 = 3(\mu^2_{2,1} + \mu^2_{1,2});\\
\sum_{i,j,k,l\in I_4} (Q_{i,j,k,l})^2 =& \binom{4}{1}(E[\tilde{\mathbf{H}}^3_{\hat{k}}\tilde{\mathbf{O}}_t] - 3E[\tilde{\mathbf{H}}^2_{\hat{k}}]E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t])^2 + \binom{4}{2}(E[\tilde{\mathbf{H}}^2_{\hat{k}}\tilde{\mathbf{O}}^2_t] - E[\tilde{\mathbf{H}}^2_{\hat{k}}]E[\tilde{\mathbf{O}}^2_t] - 2E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t])^2 +\\
& \binom{4}{3}(E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}^3_t] - 3E[\tilde{\mathbf{O}}^2_t]E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t])^2\\
=& 4(\mu_{3,1} - 3\mu_{2,0}\mu_{1,1})^2 + 6(\mu_{2,2} - \mu_{2,0}\mu_{0,2} - 2\mu^2_{1,1})^2 + 4(\mu_{1,3} - 3\mu_{0,2}\mu_{1,1})^2.
\end{aligned}$$

The mutual information value can be estimated directly from the data, by using

$$E[\tilde{\mathbf{H}}^a_{\hat{k}}\tilde{\mathbf{O}}^b_t] = \mu_{a,b} \approx m_{a,b} = \frac{1}{q}\sum_{i=1}^{q} (\tilde{h}_{i,\hat{k}})^a (\tilde{o}_{i,t})^b,$$

where $m_{a,b}$ is the a,b-th joint sample moment of $\tilde{\mathbf{H}}_{\hat{k}}$ and $\tilde{\mathbf{O}}_t$. Here $\tilde{h}_{i,\hat{k}}$ is the normalized prediction value under key guess $\hat{k}$ and $\tilde{o}_{i,t}$ is the normalized observation at time t, both arising from processing plaintext d_i. Using these sample moments in Equation 4.6, the mutual information value is estimated as:

$$\begin{aligned}
I(\tilde{\mathbf{H}}_{\hat{k}}; \tilde{\mathbf{O}}_t) \approx& \frac{1}{2}m^2_{1,1} + \frac{1}{4}(m^2_{2,1} + m^2_{1,2}) + \frac{1}{12}\left((m_{3,1} - 3m_{2,0}m_{1,1})^2 + (m_{1,3} - 3m_{0,2}m_{1,1})^2\right)\\
&+ \frac{1}{8}(m_{2,2} - m_{2,0}m_{0,2} - 2m^2_{1,1})^2.
\end{aligned} \tag{4.9}$$

Evaluation

Cumulants strongly rely on the Gaussian assumption. However, in general this assumption does not completely hold. In case the data is not Gaussian, but uniform (or even constant), the estimated MI value might give unusual results. In order to prevent this, a goodness-of-fit test for

normality, like the one described by Jarque and Bera [JB80], might be inserted within the analysis. Time points where observations deviate significantly from a normal distribution might be excluded. However, most tests require that more measurements yield a better approximation of the normal distribution. This does not hold in practice, since observations might follow a distribution that will never adequately approximate the normal distribution. Therefore, no goodness-of-fit test is implemented in the mutual information analysis described in this thesis.

However, it might be interesting for further research. Note that to improve MIA, a goodness of fit test must be fast and applicable on large data sets. By these criteria, tests like the Shapiro-Wilk test, requiring order statistics, are not very feasible on large trace sets.

In order to grasp how the cumulant model distinguishes key candidates, the expression from Equation 4.6 is evaluated. The first summation in Equation 4.6 can equivalently be written as:

$$\frac{1}{4}\sum_{i,j\in I_2}(R_{i,j})^2 = \frac{1}{2}E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t]^2 = \frac{1}{2}E\left[\frac{\mathbf{H}_{\hat{k}}-m_{\mathbf{H}_{\hat{k}}}}{s_{\mathbf{H}_{\hat{k}}}}\cdot\frac{\mathbf{O}_t-m_{\mathbf{O}_t}}{s_{\mathbf{O}_t}}\right]^2 = \frac{E[(\mathbf{H}_{\hat{k}}-m_{\mathbf{H}_{\hat{k}}})(\mathbf{O}_t-m_{\mathbf{O}_t})]^2}{2s^2_{\mathbf{H}_{\hat{k}}}s^2_{\mathbf{O}_t}}.$$

This equals the square of the correlation coefficient (cf. Definition 2.3.1) of $\tilde{\mathbf{H}}_{\hat{k}}$ and $\tilde{\mathbf{O}}_t$, divided by two. However, squaring and halving is a monotonous operation on the absolute value of the argument. Recall that correlation power analysis is also based on the absolute correlation factor. Hence, this first term gives a contribution related to the correlation coefficient of $\tilde{\mathbf{H}}_{\hat{k}}$ and $\tilde{\mathbf{O}}_t$.

Mutual information analysis using cumulants apparently evaluates correlation, and more factors. These third and fourth order cumulants can be viewed as representing higher order dependencies of prediction and observation.

Due to the parametric nature of this mutual information estimation method, it is more noise-resistant than histogram-based mutual information estimation, as pointed out by Le and Berthier [LB10]. Summarizing, one can conclude that cumulant-based mutual information is especially powerful when prediction and observation follow a (near-) normal distribution. However, this information might be a priori unknown.

4.4.3 Other

Density estimation is a well-studied topic in probability theory. Therefore, many methods are described, each having typical benefits and drawbacks. In this section, three methods are discussed. Each of these methods is used in combination with mutual information analysis in at least one technical report. Besides the three methods explained here, many other are applicable to MIA. For example, mutual information analysis using Kologorov-Smirnov tests and Cramer-von-Mises tests is explained by Veyrat-Charvillon and Standaert [SVC09].

Whereas many density estimation tools are named *nonparametric*, the majority of these methods still depends on one or more parameters. However, no assumption about the underlying distribution is made, which is the case in parametric estimation.

Kernel Density Estimation

A kernel K, sometimes referred to as Parzen window, is a mass function, having the following two properties:

$$K(x) = K(-x) \quad \forall x; \tag{4.10}$$

$$\int_{-\infty}^{\infty} K(x)dx = 1. \tag{4.11}$$

Every function satisfying these properties can be used as a kernel function. Some examples are listed here; in, e.g., Batina et al. [BGP+11] a wider treatment is given. The function $i(t)$ represents the indicator function, being equal to one if statement t is true and zero otherwise.

- Uniform: $K(x) = \frac{1}{2}i(|x| \leq 1)$;
- Gaussian: $K(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$;
- Epanechnikov: $K(x) = \frac{3}{4}(1 - x^2)i(|x| \leq 1)$.

Kernel density estimation is a generalization of histogram density estimation. A histogram is constructed by increasing mass uniformly over a fixed interval (the bin), corresponding to the value of an observation. A kernel-based density is constructed by adding a kernel centered at the value of an observation. The overall density is then the sum of all the kernels. In essence, the density at a point t is estimated by a weighted sum of observations within a certain range around t. The weight is determined by the kernel function, the range by the bandwidth of the kernel.

A short overview of benefits and drawbacks of kernel density estimation in MIA is given below.

Benefits

- A more smooth distribution is obtained, generally converging faster than a histogram-based distribution. However, the distribution is defined in terms of a finite (but possibly large) sum of kernels. Hence, it might be hard to obtain a simple expression covering the density function.
- Observations close to histogram bin bounds might affect the overall drastically when the bin bounds are shifted a little to left or right. Kernel density estimation is not vulnerable to such effects from partitioning intervals. Hence, this estimation method show more noise-resistance than a histogram-based estimation method.

Drawbacks

- As mentioned before, describing the final probability density functions concisely can be difficult when many observations must be taken into account. Moreover, to compute mutual information values, entropies must be estimated from this densities. Estimating entropy of a continuous function requires integration, hence an adequate discretization of the probability density has to be distilled. This process, as well as estimating entropy from the discretization, might be prone to numerical errors. Moreover, a balance between preciseness and computational complexity must be found to guarantee decent performance on these both aspects.
- A kernel shape and scale (the bandwidth) must be selected, so this method still has some degrees of freedom. Performance depends on this selection, and can be vulnerable to errors due to improper decisions. A too large bandwidth can result in low resolution, whereas a too small bandwidth might yield estimates that are vulnerable to statistical variation.

Kernel density estimation is applied on MIA by, among others, Veyrat-Charvillon and Standaert [SVC09], Prouff and Rivain [PR09], Venelli [Ven10] and Batina et al. [BGP+11]. Kernel MIA is claimed to have advantages over histogram MIA, in terms of efficiency and noise resistance. However, the results presented in these collaboration are generally on small (i.e., $\leq$ 1000 traces) trace sets, to limit computation time. In contrast, the number of traces required to correctly compromise a subkey might easily be in the order of tons for high-end cryptographic devices.

B-Splines

Mutual information analysis using B-splines for density estimation is introduced by Venelli [Ven10] at WISTP 2010. A B-spline is a generalized Bézier curve; a density approximation consists of a collection of consecutive B-splines. Continuity is demanded, and optional for the first derivative. An extensive treatment of entropy estimation through B-splines is given by Venelli, the reader is referred to this paper for more information. Here, a brief summary of benefits and drawback on this method is provided.

Benefits

- In comparison with histogram-based density estimation, B-spline density estimation particularly is more noise-resistant. Since noise is inevitable in side channel analysis on practical implementations, this is strongly favorable.
- Entropy estimation via B-splines is less naive than using histograms; however, it is less complex than a kernel density estimation. Hence, it might serve as a good compromise between these two methods.

Drawbacks

- A continuous, piecewise smooth probability distribution is obtained. However, from this distribution entropy must be estimated, requiring discretization. The extent to which smoothness contributes in providing more information, without heavily increase computational complexity, is partly assessed in the paper. However, results in Venelli [Ven10] show that density estimation via B-splines does increase computation time significantly.
- Although this method does not assume an underlying distribution, certain parameters have to be set. For example, the number of curves used to estimate an empirical distribution, and the degree of the curves.

Parametric Models

When an attacker has knowledge about underlying probability distributions of observation and prediction, a parametric model might be a good choice. In that case, an attacker assumes a certain probability distribution and optimizes the associated parameters by examining data. In side channel analysis, many parametric models focus on estimating parameters of Gaussian distributions. Among others, Prouff and Rivain [PR09] and Flament et al. [FGD+10] describe this technique.

Benefits

- This method is generally simple. No complex calculations need to be performed to estimate parameters. Moreover, most parameters can be estimated incrementally, yielding a memory-efficient and fast method.
- Mutual information can be estimated directly from parameters. Entropy of a distribution can be expressed in terms of distribution parameters (see Example 2.1.15), hence only these parameters are required to adequately estimate entropy. Key guesses might wrongly be indicated as correct when deviating much from the assumed – wrong – density.

Drawbacks

- In general, underlying distributions might not be known. Whereas the underlying distribution of predictions can be easily determined for the power model, observations do not follow a simple distribution in general. Hence, selecting an underlying model can already be a challenging step for an attacker.
- If the underlying distribution differs from the one assumed, performance reduces drastically since this attack focuses on behavior of samples according to the assumed distribution. Key guesses might wrongly be indicated as correct when deviating much from the assumed – wrong – density.

In mutual information analysis using cumulants, densities are assumed to be nearly Gaussian. Therefore, the cumulant method might be viewed as a parametric model, where underlying distribution parameters (mixed cumulants) are estimated as adequately as possible.

An evaluation of density estimation tools for side channel analysis is performed by Flament et al. [FGD+10]. Here, parametric density estimation is compared to histogram estimation in side channel context. Moreover, an overview of density estimation techniques applied in mutual information attacks is given by Le and Berthier [LB10].

4.5 Discussion

Besides practical benefits and drawbacks compared to other side channel distinguishers, mutual information can be used in a broader scope as information-theoretic measure. In this section, mutual information analysis is assessed in a more contemplative scope. Mutual information is assessed as measure for success of different side channel distinguishers. Furthermore, correlation analysis and mutual information analysis are compared on multiple aspects.

4.5.1 Reflections on Mutual Information

Mutual information represents the amount of information shared by two random variables. In other words, it quantifies the relation between them. This includes both linear and nonlinear relations. Therefore, mutual information exploits more information than a correlation coefficient does. However, mutual information does not indicate *how* random variables are related, in contrast with Pearson's correlation.

Since no information about the type of relation emerges from a mutual information value, an attacker might only be interested in the highest values (i.e., the best guesses), no matter what these values actually are. Three properties related to mutual information values are discussed below.

Estimation method. First of all, mutual information values depend on the probability density estimation method; some methods (or instances of a method) provide more information than other ones. When comparing results from different mutual information attacks, an attacker must note that an increase in mutual information might not evolve from better key guesses, but from different mutual information estimations. To avoid this problem, an attacker might consider usage of normalized mutual information, as introduced in Definition 2.2.8. Normalized mutual information might however not be adequate for analysis, as argued in Section 6.3.3.

Resolution. Second, probability density estimations must provide enough information to distinguish key guesses, but are also required to converge towards the real underlying density method. Consider that a histogram with many bins can reflect more information than a histogram with a few bins, but a higher resolution can induce slower convergence. When observations from a continuously distributed random variable are examined, the theoretical probability of two identical observations equals zero. Hence, when the number of bins tends to infinity, entropy of n observations theoretically approaches $\log n$. This value is independent of the actual values of the observations. Consequently, when mutual information is used as distinguisher, no distinction can be made, as indicated by Prouff and Rivain [PR09]. Therefore, the number of bins should not be greater than the number of measurements. Alternatively one might demand that assignment of observations to bins should not be injective. This is a weak constraint, which can easily be fulfilled.

Comparative Metric. Third, since mutual information indicates the strength of a dependency, efficiency of other distinguishers (such as correlation) can be expressed relative to mutual information. A more extensive comparison of correlation coefficients and mutual information as distinguisher is given in the next section. For an arbitrary distinguisher, one can estimate the theoretical amount of information on which key guesses are compared. This amount can be related to mutual information of corresponding key guesses to express the discriminating power of a distinguisher.

In Veyrat-Charvillon and Standaert [SVC09], a brief description of mutual information analysis as an evaluation metric is given. Assumptions on noiseless environments and perfect density estimation are made; this does not apply in practice. When these assumptions do not hold, mutual information estimated by a MIA tends to underestimate the actual amount of information leaked from an implementation. Clearly, an estimation method can not find more information than actually present.

4.5.2 Mutual Information Analysis versus Correlation Analysis

Where mutual information of predictions and observations is estimated in mutual information analysis, correlation coefficients are estimated in correlation power analysis. CPA is explained in Section 3.3. Mutual information-based distinguishers and correlation-based distinguishers are compared in this section.

Mutual information analysis aims at finding a general relation, instead of only linear relations as targeted by correlation attacks. This strength in generality might as well turn into a weakness. Moradi et al. [MMPS09] (among others) observe the following property.

On the one hand, suppose that there is a key guess $\hat{k} \neq \kappa$, where κ is the correct key, such that $|\rho(\mathbf{H}_{\hat{k}}, \mathbf{H}_\kappa)| = 1$. Then a correlation attack will not be able to distinguish the correct key guess κ from the wrong key guess $\hat{k}$. In this case, a perfectly linear relation between $\mathbf{H}_{\hat{k}}$ and $\mathbf{H}_\kappa$ exists. Then, also a mutual information attack is unable to distinguish these key guesses.

On the other hand, suppose there is a $\hat{k} \neq \kappa$ such that $|\rho(\mathbf{H}_{\hat{k}}, \mathbf{H}_\kappa)| < 1$. If predictions using $\hat{k}$ have a stronger arbitrary relation with observations than a linear relation between predictions using κ and observations, wrong key guesses are given a higher mutual information score than the correct key guess. Hence, a MIA does not yield the desired result, whereas correlation attacks do not suffer from this problem.

It follows that mutual information analysis might be more sensitive to produce ghost peaks. An unintentional dependence or relation between predictions and observation can occur, e.g., through statistical fluctuations. This unintentional relation should have a linear nature to be noticed by a correlation-based distinguisher. However, since mutual information analysis aims at any relation in general, the likelihood that an arbitrary relation is noticed by MIA is larger. Hence, a ghost peak is more likely to occur in MIA than in CPA.

Moradi et al. therefore conclude this discussion stating that when a linear relation between predictions and observations holds, MIA is able to distinguish in fewer cases than CPA. However, arbitrary dependence through statistical fluctuation might vanish due to averaging, when increasing the number of observations. Arbitrary relations due to physical properties of implementation can as well exist; however, this has not been encountered in practice. Hence, although mutual information attacks may require more traces to correctly compromise subkeys, it is theoretically sensitive enough to find linear relations, when no other intentional dependence exists.

To avoid ghost peaks, the number of measurements needs to be increased. This reduces the efficiency of a mutual information attack and is reflected in the results on MIA and comparative results on CPA, presented in Chapter 7. One can see that a MIA generally requires more power traces to correctly compromise cipher keys, especially when dependence between power consumption and prediction is nearly linear. As indicated by Gierlichs et al. [GBTP08] and later papers on MIA, mutual information attacks will not be more efficient in this linear case, so a correlation attack is preferred in scenarios of linear dependence between prediction and observation. MIA's generality is a drawback in that case.

However, mutual information analysis would not be considered a high-potential side channel distinguisher if there were no cases where MIA is stronger than CPA. The assumption of a linear dependence between prediction and observation holds in numerous cases, but implementations showing nonlinear behavior increasingly appear on the market. One of these implementations is examined in Section 7.3.3. Special countermeasures can be implemented to prevent linear relations between power consumption and intermediate values. In that case, correlation attacks are no longer useful. A mutual information attack, however, can still exploit nonlinear leakage of information.

4.5.3 CPA as Instance of MIA

One thought experiment involving mutual information analysis is to search for relation between CPA and MIA. More specifically, in this section the question to what extent correlation analysis is an instance of mutual information analysis is evaluated.

When a random variable is fully determined by another random variable, their mutual information equals the minimum of both entropies, as indicated in Definition 2.2.8. As described in the same consequence, dividing mutual information by this minimal entropy yields the normalized mutual information (I_N): a measure in domain $D_{I_N} = [0, 1]$. A correlation coefficient of (minus) one indicates a bijective relation, this is a case where one variable is fully determined by another. Hence, the absolute value of a correlation coefficient (AC) scales in the same domain $D_{AC} = [0, 1]$. Whereas a one in D_{AC} corresponds to a one in D_{I_N}, the converse is not true. Indeed, a random variable can fully determine another random variable in a nonlinear way, thus yielding correlation coefficients unequal to one.

Moreover, a zero in D_{I_N} yields a zero in D_{AC}, since zero mutual information implies independence of random variables (see Lemma 2.2.5). The converse is not true, since two uncorrelated random variables can have a nonlinear dependence which is reflected in their mutual information value.

Correlation coefficients and normalized mutual information are investigated as follows. A set of 10000 paired observations from a bivariate normal distribution with correlation coefficient ρ are simulated. From this set, normalized mutual information is estimated. This is done for $\rho = 0, \ldots, 1-\alpha$, in steps of $\alpha = 1/10000$. The results are depicted in Figure 4.9.

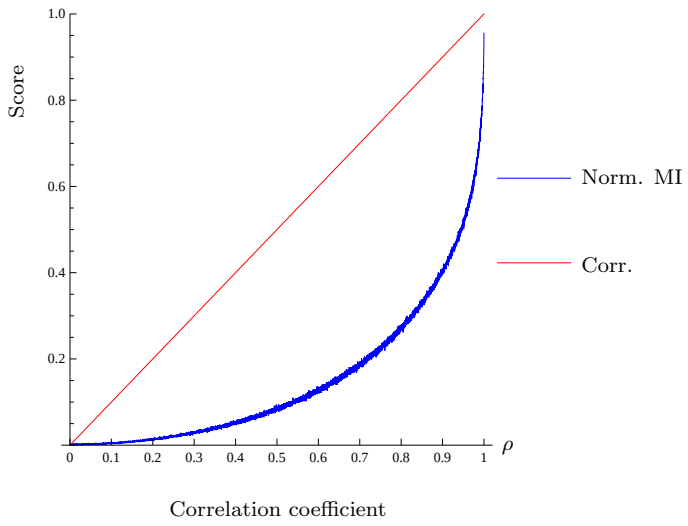


Figure 4.9: Normalized mutual information of bivariate normally distributed samples with correlation coefficient ρ.

An analytical expression for this graph can be obtained by evaluating entropies of (bivariate) standard normal random variables (cf. Example 2.1.15) and by applying $I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) + H(\mathbf{Y}) - H(\mathbf{X}, \mathbf{Y})$. The normalized mutual information equals:

$$I_N(\mathbf{X}; \mathbf{Y}) = \frac{1/2 \log(2e\pi) + 1/2 \log(2e\pi) - \log((2e\pi)\sqrt{1-\rho^2})}{1/2 \log(2e\pi)} = \frac{-2 \log(\sqrt{1-\rho^2})}{\log(2e\pi)}.$$

Thus, normalized mutual information of a bivariate normal distribution scales as $c \cdot \log(1-\rho^2)$

to correlation, for a constant c. As can be seen, $I_N(\mathbf{X};\mathbf{Y})$ is close to one for high correlation values. However, a low correlation value yields an even lower normalized mutual information value. Normalized mutual information might however be less practical on real embeddings. An evaluation of regular mutual information and normalized mutual information as practical distinguisher is presented in Section 6.3.3.

One important property of the bivariate normal distribution is that when $\rho = 0$, the two random variables actually are independent (see, e.g., Bain and Engelhardt [BE92]). This does not hold for arbitrary bivariate distributions. However, this property is reflected in the fact that normalized mutual information vanishes when correlation vanishes: random variables show less dependent behavior.

As explained in the evaluation of cumulant-based MIA in Section 4.4.2, cumulants examine multiple types of dependency between random variables. One of these types is correlation-related. Thus, correlation analysis is included explicitly in first order cumulant-based MIA. In histogram-based mutual information analysis, this relation is not that explicit. However, it is included in mutual information estimation, since it is one of the many type of relations two random variables can share.

Alternative approach

Another approach to assess whether CPA is an instance of MIA is the following one. Suppose an adversary has performed a correlation attack that successfully recovered the cipher key. The adversary is now curious whether or not this guarantees the existence of a mutual information attack able to correctly compromise the cipher key. A successful CPA implies that for every incorrect key guess $\hat{k} \neq \kappa$: $\rho(\mathbf{H}_{\hat{k}}, \mathbf{O}_t) < \rho(\mathbf{H}_\kappa, \mathbf{O}_t)$. This does not imply $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) < I(\mathbf{H}_\kappa; \mathbf{O}_t)$ for all $\hat{k} \neq \kappa$. The following arguments are analogous to those presented in Section 4.5.2.

Nonlinear dependence might appear due to, e.g., statistical fluctuations or physical properties of the implementation. When physical properties cause nonlinear dependence for some subkey, then mutual information analysis is sensitive to this. When this nonlinearity has a significant contribution, that is, the incorrect subkey is appointed as the most likely key guess, instead of the correct one, mutual information analysis fails. This failure will then appear in every type of mutual information analysis on any number of traces coming from the device.

However, wrong subkeys are expected to yield a random sample from the observation distribution, as indicated by Veyrat-Charvillon and Standaert [SVC09]. Hence, it is not likely that predictions using a wrong key guess show nonlinear dependence with observations, other than due to statistical fluctuation. By increasing the number of measurements, this fluctuation can be overcome. Prouff and Rivain [PR09] indicate (perhaps trivially) that a mutual information attack only succeeds when

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t) < I(\mathbf{H}_\kappa; \mathbf{O}_t) \tag{4.12}$$

theoretically holds for all $\hat{k} \neq \kappa$. They assume this to be true; moreover, Veyrat-Charvillon and Standaert [SVC09] and Batina et al. [BGP+11] do.

When the inequality in (4.12) holds, a successful correlation attack proves that a mutual information attack should as well work: enough information leaks through power consumption. Results presented in Chapter 7 support this statement: in every scenario where a correlation attack is successful, also a mutual information attack is successful. However, no indication about the number of traces required to succeed can be derived. This means that when a correlation attack reveals the cipher key based on a certain number of traces, MIA might require significantly more traces to obtain similar results. This increase can be due to nonlinear statistical fluctuations, or to the fact that many samples are required to obtain adequate density estimations.

Summary

Mutual information analysis aims at testing correlation of two variables, plus multiple other types of relations that might not even be described explicitly. Not only cumulant-MIA, but also mutual information analysis using histogram density estimation can be seen as a broader type of distinguisher than correlation analysis. When a correlation attack succeeds and Inequality 4.12 holds, MIA should as well succeed; however, this success might require significantly more power traces.

Correlation analysis can be seen as an restricted instance of mutual information analysis in that the relations targeted by CPA (linear ones) are also targeted by a MIA. Furthermore, a successful CPA implies the existence of a successful MIA. However, to extend a CPA to a MIA, much more measurements are generally required.

Chapter 5

Higher Order Mutual Information Analysis

Particularly on protected implementations, a first order attack might not lead to a successful cipher key recovery. Higher order attacks use multivariate statistics to correctly recover the cipher key, especially on implementations protected by countermeasures. In this chapter, higher order mutual information analysis (HO-MIA) is introduced and explained. Higher order MIA is a true generalization of first order mutual information analysis, introduced in the previous chapter.

First, an introduction to a specific countermeasure, boolean masking, is given. Then, the extension of first order mutual information analysis to higher order MIA is treated. After discussing the state of the art, theoretical properties are explained, including multiple higher-order generalizations of mutual information. Two methods for estimating mutual information are explained and evaluated. Finally, higher order MIA is compared to higher order correlation analysis.

5.1 Masking as Countermeasure

To improve the security of a cryptographic device, different countermeasures can be implemented. Countermeasures aim at concealing dependence between values processed in the device and power consumption of the device. Masking is a countermeasure that randomizes intermediate values. Cryptographic algorithms have to be adapted in order to unmask intermediates after being processed. By randomizing intermediate values, independence of processed data and power consumption is achieved. In this section, the concept of masking schemes is explained, as well as the security properties of a masking scheme.

In masking, an intermediate value x is concealed by a mask m. The mask m is typically chosen uniformly random from the same space where the intermediate value it aims to conceal belongs to. In their definition of masking schemes, Mangard et al. [MOP07] distinguish two different types of masking, being Boolean masking and arithmetic masking. Arithmetic masking consists of applying an arithmetic function to the intermediate and the mask, usually modular addition or modular multiplication. This means that the masked value x_m is constructed by $x_m = x \cdot m \mod n$, where n usually is algorithm dependent. In contrast, Boolean masking uses the XOR as function, i.e. $x_m = x \oplus m$. Since Boolean masking is used in almost every masking scheme on smart cards, only this type of masking is in the scope of this thesis.

Masking attempts to disconnect power consumption from sensitive intermediate values by randomizing these values. Hence, power consumption only depends on masked (i.e. random) values, and not on the unmasked intermediate. If the mask m is drawn uniformly random from the same space as the intermediate x, the distribution of $x \oplus m$ is identical for each x. Consequently, it is independent from x. However, the mask must ultimately be removed, which forces the algorithm to process the mask until the intermediate has to be unmasked.

When a linear function f operates on the masked intermediate value, the effect of the mask in the output is clearly visible since

$$f(x \oplus m) = f(x) \oplus f(m).$$

In order to obtain the desired output, only $f(m)$ has to be computed separately. However, most cryptographic systems consist of both linear and nonlinear functions. For a nonlinear function f, in general

$$f(x \oplus m) \neq f(x) \oplus f(m).$$

Consequently, the mask must be processed in a specific way through the masking scheme.

For both DES and AES, multiple masking schemes are designed. For each algorithm, a common masking scheme is explained in Section 5.1.2 and Section 5.1.1, respectively. Results of attacks on these specific masking schemes are provided in Chapter 7. For other cryptographic algorithms such as RSA, also many masking schemes are designed. These might also use arithmetic masking. However, these methods are not evaluated in this thesis, since almost all contemporary smart cards are equipped with an implementation of DES or AES.

The concept of masking can be seen as an instance of secret sharing. When an intermediate x is concealed by m, the intermediate is represented by (x_m, m), i.e. a secret with two shares. Both shares are required to obtain x; however, knowing only one share does not give any information about x. As pointed out by Mangard et al. [MOP07], this property implies that n masks can be resistant up to an n-th order DPA attack. Hence, higher order attacks are necessary in case of masked implementations.

5.1.1 DES Masking Scheme

A common DES masking scheme uses two masks, one for each half of the round input (see Appendix A.1). Prior to the first DES f-function, the right half is masked with m_1 and the left half is masked with m_2, both of size 32 bits. Consequently, the f-input is masked by m_1. In every round, the f-function is altered such that

$$f'(r \oplus m_1) = f(r) \oplus m_1 \oplus m_2.$$

After the XOR of left half and f-output, the left output (which is equal to the right half of the next round input) is masked by m_1. The right half is XOR'ed with $m_1 \oplus m_2$, hence the left input of the next round is then masked by m_2. Thus, every round input is masked identically. In the end, the masks are removed. Figure 5.1 shows a graphical representation of this masking scheme.

Since every right half is masked by m_1, the XOR of two round inputs is unmasked. Hence, this might be an interesting intermediate variable to attack. Moreover, the second round right half input equals the first round left output, hence an attack on the distance between round input and output might be adequate to attack. In Section 7.6, results for this attack are given.

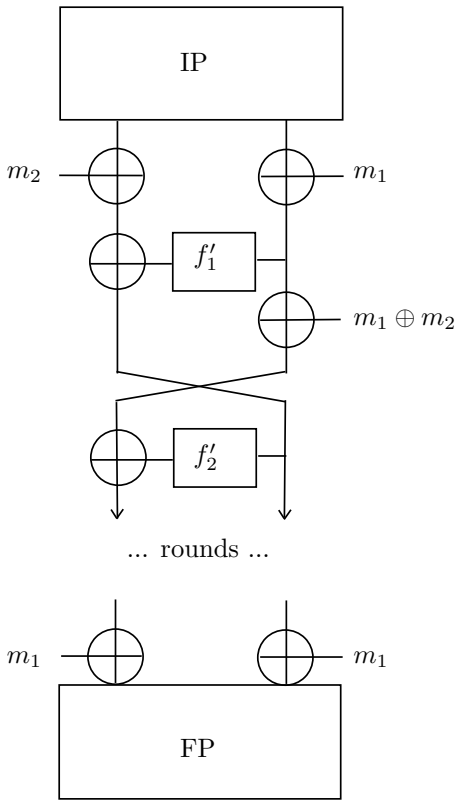


Figure 5.1: DES masking scheme using two 32-bit masks.

5.1.2 AES Masking Scheme

A common AES masking scheme, described in Mangard et al. [MOP07], masks AES states (see Appendix A.2) in each round with a mask m. Moreover, the AES `MixColumns()` operation can be masked, but no attacks on this intermediate value are presented in this thesis. The S-box input and output are assumed to be masked with the same mask. This means that the S-box function must be accordingly adapted. Let m be an 8-bit value, drawn uniformly random. The state is masked as follows:

$$\begin{array}{|c|c|c|c|}\hline m & m & m & m \\ \hline m & m & m & m \\ \hline m & m & m & m \\ \hline m & m & m & m \\ \hline \end{array} \rightarrow S'(\cdot) \rightarrow \begin{array}{|c|c|c|c|}\hline m & m & m & m \\ \hline m & m & m & m \\ \hline m & m & m & m \\ \hline m & m & m & m \\ \hline \end{array}$$

The masked S-box S' is different from the original AES S-box S: it must satisfy:

$$S'(x \oplus m) = S(x) \oplus m.$$

Hence, the masked S-box (table) must be computed prior to the first round.

Often, the S-box function literally substitutes a value. That is, S-box output $S(x)$ is written to the register as where x came from. Then, instead of leakage from the value x, information about the distance of x and $S(x)$ might leak. When both x and $S(x)$ are masked with m, their distance is unmasked. This might be a crucial observation when selecting a target to attack. In Section 7.5, results for this attack are given.

5.2 Generalizing MIA to Higher Order Scenarios

As explained in the previous section, intermediate variables are masked by random values. However, when two intermediate values are masked with the same mask, their XOR is unmasked since masks cancel out. An attacker can take advantage of this relation, by predicting the unmasked XOR and relating this to observations at the two distinct points in time, where information about the masked values leaks. In second order differential power analysis, one prediction and two observations are compared.

Comparing three variables requires a distinguisher that can deal with three input variables. Since correlation coefficients are only defined for two random variables, many higher order correlation attacks use a preprocessing step to map the multivariate problem to a univariate problem, as explained in Section 3.3.2. Specifically, two power observations are mapped to one value, which is in turn correlated to one prediction value. Using first order techniques, this problem can be solved; however, the preprocessing step might reduce performance when no linear relation exists between prediction and the combined observations.

In contrast, mutual information can easily be generalized to more dimensions. Moreover, multiple n-dimensional generalizations of mutual information exist. Four of these generalizations are discussed in Section 5.4.2. Two of them are already investigated in papers on second order mutual information. The two others are, to the best of our knowledge, not widely investigated yet on suitability for mutual information attacks where density estimation is a separate step. However, one is used in cumulant-based direct approximation of mutual information by Le and Berthier [LB10]. Higher order mutual information analysis directly works with multivariate statistics and hence does not require any preprocessing steps. Therefore, no information is lost during any transformation of the problem.

Masking schemes using 2 or more distinct masks for one intermediate value exist as well. In general, to break a masked implementation using $r-1$ masks, an r-th order attack is required, as

pointed out by Mangard et al. [MOP07]. In the following description of higher order mutual information analysis, a masked implementation with $r-1$ masks is assumed for the sake of generality. In this thesis, however, only practical attacks on unprotected and single-masked implementations are presented.

In case no information about the position of the leakages is known, $\binom{l}{r}$ samples in time interval of length l have to be examined for a masked implementation using $r-1$ masks. This strongly affects the number of calculations, and hence the running time. In practice, however, an attacker generally knows the minimum and maximum distance between two leakages in a single-masked implementation. Consequently, observations in two certain (not necessarily disjoint) ranges are compared to predicted values.

5.3 State of the Art

The introductory paper about MIA [GBTP08] posed several open questions and suggestions for further research, including extension of mutual information analysis to higher order analysis. Prouff and Rivain provide an extensive analysis of MIA in their paper "Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis" [PR09]. One of the major investigations in the paper is concerned with generalizing MIA to a higher order attack, using one specific generalization of higher order mutual information, ambiguously called generalized mutual information. For simplicity, it is assumed that an attacker knows the points in time where the selected intermediate values are processed; an assumption which does not hold in general.

Instead of results on real measurements, only simulation results on higher order MIA are given by Prouff and Rivain. However, these results indicate an advantage of second order MIA over second order CPA. For second order MIA, histogram density estimation is used, as well as identity leakage prediction. To perform higher order CPA, two different preprocessing functions (see Section 3.3.2) are used, being the absolute-difference combining and the normalized product combining. The paper concludes by stating that, according to their simulation results, a higher order extension of MIA is much more efficient than higher order CPA. However, no results on real cryptographic implementations are given, hence their claim is not validated in practice.

When introducing mutual information-based side channel analysis [GBTP08], Gierlichs et al. suggested that a mutual information-based distinguisher would be specifically interesting when applied to protected implementations. In "Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis" [GBPV10], Gierlichs et al. further develop this concept using a different choice of higher order generalization in comparison with Prouff and Rivain: interaction information. A comparison of both generalizations is therefore included in their paper.

Both simulation results and results on a practical implementation for second order and third order MIA are presented. In each case, power consumption has a nearly linear relation with the Hamming weight of processed values, hence few (less than 500) traces are required for successful attacks. The results for second order attacks indicate that second order CPA is more efficient, merely due to the linear relation and suitability of combining functions. However, in the third order attacks HO-MIA is strongly in favor. Due to relations between power consumption and Hamming weight of intermediate values, a Hamming weight power model scores better than an identity model (see Section 3.2.2 for a treatment of power models).

The two higher order generalizations are compared for both Hamming weight prediction models and identity prediction models. Performance on a Hamming weight model shows no large differences; this model is most adequate due to Hamming-weight dependent power consumption. Generalized MI is in small favor on identity prediction models, probably due to the fact that more information is examined from the low-noise embedding.

Combining the results of Prouff and Rivain [PR09], Standaert and Veyrat-Charvillon [SVC09] and Gierlichs et al. [GBPV10], a comprehensive study on mutual information analysis was performed by Batina et al. [BGP+11]. The two generalizations of mutual information are again

compared, using both a low-noise and a high-noise implementation. Again, a nearly linear relation between power consumption and Hamming weight of intermediate values is assumed. As a consequence, the Hamming weight model scores better than the identity model.

Differences between higher order generalizations resemble those indicated by Gierlichs et al. [GBPV10], for low-noise scenarios. However, interaction information tends to be more noise-resistant than generalized mutual information, since the first method scores better on noise measurements. Due to the linearity of power consumption compared to Hamming weight of intermediate values, a second order correlation attack still is more efficient.

Cumulant-based second order mutual information analysis is discussed by Le and Berthier [LB10], under a Gaussian assumption. Correlation analysis is more efficient than MIA in their results; however, cumulant-based MIA should be preferred over histogram MIA.

5.4 Theoretical Foundation

Higher order mutual information analysis is an extension of the mutual information analysis introduced in the previous chapter. Hence, both concepts share a significant part in theoretical background. In this section, the specific theory behind higher order MIA is explained. First, a treatment of prediction and observation is given. Observations include the r shares of the masked implementation. Then, estimation of mutual information values is explained. Multiple higher order generalizations of mutual information exist in literature and are explained in this section. Evidently, for a single mask, read $r = 2$.

5.4.1 Prediction and Observation

During a cryptographic run, multiple intermediate values are updated and stored over time. Since each operation requires an amount of energy, power consumption on different moments in time depends on different intermediate values. Clearly, the actual relation between power consumption and intermediate values depends on the physical properties of the cryptographic device and needs to be modeled.

In a masked implementation, intermediate values are randomized to prevent first-order analysis. However, combining information from two values, masked with identical masks, might reveal the unmasked value. Hence, an attacker aims at finding two (or more, for more than one mask) intermediate values that are masked with the same mask. The XOR of these values is unmasked, and when the unmasked intermediate variables can be predicted, based on a certain key guess $\hat{k}$, their XOR can also be predicted. This prediction value is, analogous to early notation, denoted by $\mathbf{H}_{\hat{k}}$.

Like in Chapter 4, observation at time t is represented by a random variable $\mathbf{O}_t$. In an attack on a masked implementation, power consumption from at least r distinct points in time has to be measured in order to obtain information about the cipher key. Hence, calculation of mutual information involves $\mathbf{O}_{t_1}, \ldots, \mathbf{O}_{t_r}$; the r distinct measurements at times $t_1, t_2, \ldots, t_r$.

5.4.2 Higher Order Mutual Information Generalizations

Several proposals to extend the mutual information to multivariate cases exist. One should bear in mind that different notions of 'shared information' can be distinguished. For example, the mutual information between three random variables can represent the information shared by all three variables, or can also consider the information shared by any pair. In e.g. Jakulin and Bratko [JB04], calculation of information arising from multiple random variables is discussed. In general, a set of n random variables can have $2^n - 1$ information relations. To measure these relations, at most the same number of degrees of freedom are required. However, not all relations are interesting to examine in side channel analysis. In this section, four methods are evaluated as mutual information-based side channel distinguisher.

When describing the generalizations, it is assumed that all random variables are defined over the same space, for simplicity. Hence, $x \in \mathcal{X}^n$ is an n-dimensional realization of $\mathbf{X}_1, \ldots, \mathbf{X}_n$ and

$$P(\mathbf{X} = x) = P(\mathbf{X}_1 = x_1, \ldots, \mathbf{X}_n = x_n).$$

Besides a description of mutual information generalizations in n dimensions, an explicit expression for the case $n = 3$ is provided, because implementations and results in the next chapters are focused on second order analysis.

'Generalized' Mutual Information

One generalization considers mutual information of a single random variable and the joint behavior of multiple random variables. It can be seen as the information shared by the input and outputs of an n-way channel. The following notation is used by Prouff and Rivain [PR09]:

$$I(\mathbf{X}_1; (\mathbf{X}_2, \ldots, \mathbf{X}_n)). \tag{5.1}$$

In terms of probability distributions, one computes:

$$\sum_{x_1 \in \mathcal{X}} f_{\mathbf{X}_1}(x_1) \sum_{x_2, \ldots, x_n \in \mathcal{X}^{n-1}} f_{\mathbf{X}_2, \ldots, \mathbf{X}_n | \mathbf{X}_1}(x_2, \ldots, x_n | x_1) \log \left(\frac{f_{\mathbf{X}_2, \ldots, \mathbf{X}_n | \mathbf{X}_1}(x_2, \ldots, x_n | x_1)}{f_{\mathbf{X}_2, \ldots, \mathbf{X}_n}(x_2, \ldots, x_n)} \right). \tag{5.2}$$

In context of side channel analysis on a masked implementation, the grouping of random variables $\mathbf{X}_1$ and $(\mathbf{X}_2, \ldots, \mathbf{X}_n)$ is not arbitrary: one might distinguish one prediction value and $n - 1$ observations. The generalized mutual information of three random variables is given by

$$I(\mathbf{X}; (\mathbf{Y}, \mathbf{Z})) = H(\mathbf{Y}, \mathbf{Z}) - H(\mathbf{Y}, \mathbf{Z} | \mathbf{X}) = H(\mathbf{X}) + H(\mathbf{Y}, \mathbf{Z}) - H(\mathbf{X}, \mathbf{Y}, \mathbf{Z}).$$

Since $H(\mathbf{Y}, \mathbf{Z}) \geq H(\mathbf{Y}, \mathbf{Z} | \mathbf{X})$ (Corollary 2.2.6), this generalization is nonnegative.

Gierlichs et al. [GBPV10] named this type *generalized mutual information* (GM). Note that since multiple generalizations exist, this name is ambiguous. However, for comparative reasons, the name is used in this thesis. This does not mean that generalized mutual information represents the only true generalization of mutual information.

Interaction Information

A generalization coming from the set-theoretic notion of mutual information is based on inclusion-exclusion and is sometimes referred to as multivariate mutual information. To avoid another ambiguity in naming, this generalization is called *interaction information* (II); a name used for this concept by McGill [McG54], among others.

$$I(\mathbf{X}_1; \ldots; \mathbf{X}_n) = - \sum_{T \subseteq \{1, \ldots, n\}} (-1)^{n - |T|} H(\mathbf{X}_T). \tag{5.3}$$

Here, $\mathbf{X}_T = \{\mathbf{X}_i : i \in T\}$. In terms of probability distributions, one computes:

$$I(\mathbf{X}_1; \ldots; \mathbf{X}_n) = - \sum_{T \subseteq \{1, \ldots, n\}} (-1)^{n - |T|} \sum_{x_T \in \mathcal{X}^{|T|}} f_{\mathbf{X}_T}(x_T) \log f_{\mathbf{X}_T}(x_T).$$

This expression involves computation of $2^n - 1$ entropies. Interaction information of three random variables is given by

$$I(\mathbf{X}; \mathbf{Y}; \mathbf{Z}) = I(\mathbf{X}; \mathbf{Y} | \mathbf{Z}) - I(\mathbf{X}; \mathbf{Y}). \tag{5.4}$$

$$= H(\mathbf{X}, \mathbf{Y}) + H(\mathbf{Y}, \mathbf{Z}) + H(\mathbf{X}, \mathbf{Z}) - (H(\mathbf{X}) + H(\mathbf{Y}) + H(\mathbf{Z}) + H(\mathbf{X}, \mathbf{Y}, \mathbf{Z})) \tag{5.5}$$

Gierlichs et al. [GBPV10] present the negative value of this measure. It is possible that this generalization attains negative values. The exact meaning of negative mutual information among multiple random variables is explained in Section 5.4.3.

Total Correlation

The *total correlation* (TC) (see, e.g., Watanabe [Wat60] or Han [Han78]) of a set of random variables is given by:

$$C(\mathbf{X}_1, \ldots, \mathbf{X}_n) = \sum_{i=1}^{n} H(\mathbf{X}_i) - H(\mathbf{X}_1, \ldots, \mathbf{X}_n). \tag{5.6}$$

Total correlation indicates the amount of information shared by *all* random variables and is equal to an instance of the Kullback-Leibler divergence:

$$C(\mathbf{X}_1, \ldots, \mathbf{X}_n) = D_{\mathrm{KL}}(f_{\mathbf{X}_1,\ldots,\mathbf{X}_n} || f_{\mathbf{X}_1} \cdot \ldots \cdot f_{\mathbf{X}_n}).$$

Note that total correlation is not restricted to measuring linear dependence among multiple variables. This is in contrast with (Pearon's) correlation coefficient of two variables.

Total correlation can easily be estimated by evaluating the joint entropy and the sum of the individual entropies, or the Kullback-Leibler divergence:

$$D_{\mathrm{KL}}(f_{\mathbf{X}_1,\ldots,\mathbf{X}_n} || f_{\mathbf{X}_1} \cdot \ldots \cdot f_{\mathbf{X}_n}) = \sum_{x_1,\ldots,x_n \in \mathcal{X}^n} f_{\mathbf{X}_1,\ldots,\mathbf{X}_n}(x_1, \ldots, x_n) \log \left(\frac{f_{\mathbf{X}_1,\ldots,\mathbf{X}_n}(x_1, \ldots, x_n)}{f_{\mathbf{X}_1}(x_1) \cdot \ldots \cdot f_{\mathbf{X}_n}(x_n)} \right).$$

Total correlation of three random variables is given by

$$C(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = H(\mathbf{X}) + H(\mathbf{Y}) + H(\mathbf{Z}) - H(\mathbf{X}, \mathbf{Y}, \mathbf{Z}).$$

This generalizations is nonnegative, as it is an instance of the Kullback-Leibler divergence.

Dual Total Correlation

The *dual total correlation* (DT) (see, e.g., Han [Han78]) of a set of random variables is given by:

$$D(\mathbf{X}_1, \ldots, \mathbf{X}_n) = H(\mathbf{X}_1, \ldots, \mathbf{X}_n) - \sum_{i=1}^{n} H(\mathbf{X}_i | \mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n), \tag{5.7}$$

or equivalently by

$$D(\mathbf{X}_1, \ldots, \mathbf{X}_n) = \sum_{i=1}^{n} H(\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n) - (n-1) H(\mathbf{X}_1, \ldots, \mathbf{X}_n). \tag{5.8}$$

Dual total correlation of three random variables is given by

$$D(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = H(\mathbf{X}, \mathbf{Y}) + H(\mathbf{Y}, \mathbf{Z}) + H(\mathbf{X}, \mathbf{Z}) - 2H(\mathbf{X}, \mathbf{Y}, \mathbf{Z}).$$

This generalization is a sound dual of total correlation. If a general measure of information L, using collections $\mathbf{S}_i \subseteq \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$, $(i = 1, \ldots, n)$, is defined as

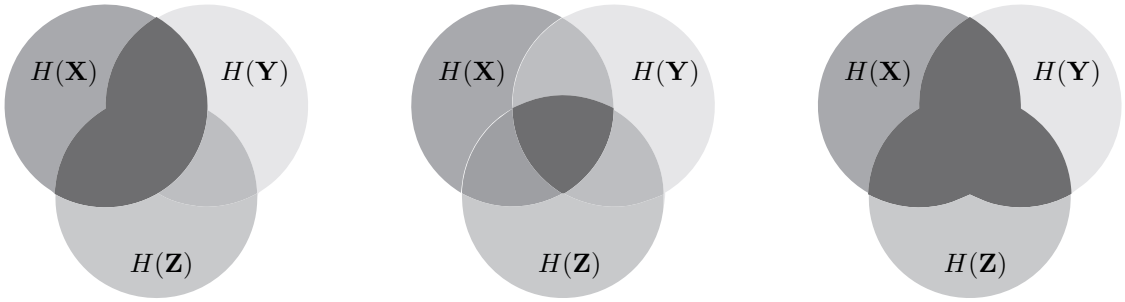
$$L(\mathbf{S}_i) = \sum_{i=1}^{n} H(\mathbf{S}_i) - |\mathbf{S}_i| \cdot H(\mathbf{X}_1, \ldots, \mathbf{X}_n),$$

then Equation 5.6 appears for $\mathbf{S}_i = \mathbf{X}_i$. Dually, define $W(\mathbf{S}_i) = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\} \backslash \mathbf{S}_i$. Now Equation 5.8 appears when evaluating

$$L(W(\mathbf{S}_i)) = L(\{\mathbf{X}_1, \ldots, \mathbf{X}_n\} \backslash \mathbf{X}_i) = L(\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n).$$

The dual relation follows since

$$W(W(\mathbf{S}_i)) = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\} \backslash \{\mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_n\} = \mathbf{X}_i = \mathbf{S}_i.$$



$I(\mathbf{X};(\mathbf{Y},\mathbf{Z}))$ $\quad -I(\mathbf{X};\mathbf{Y};\mathbf{Z})$ $\quad D(\mathbf{X},\mathbf{Y},\mathbf{Z})$

Figure 5.2: Information diagrams for different generalizations of mutual information.

Note that, analogous to total correlation, dual total correlation is not limited to linear dependence among variables. In relation to total correlation holds

$$\frac{C(\mathbf{X}_1,\ldots,\mathbf{X}_n)}{n-1} \leq D(\mathbf{X}_1,\ldots,\mathbf{X}_n) \leq (n-1)C(\mathbf{X}_1,\ldots,\mathbf{X}_n).$$

However, this domain can be particularly large for large n. As a consequence of the nonnegativity of total correlation, dual total correlation is also nonnegative.

Higher Order Cumulants

Le and Berthier [LB10] introduce a mutual information approximation based on cumulants up to the fourth order to use on a masked implementation. As indicated in Section 4.4.2, cumulants are used to estimate mutual information via Kullback-Leibler divergence. Hence, higher order mutual information estimated by this method resembles estimation of total correlation. Therefore, an investigation in the performance of total correlation estimated by histograms compared to higher order cumulant-MIA is especially interesting.

Besides that, Le and Berthier [LB10] indicate that mutual information of three random variables might be estimated using the recursive formula for interaction information:

$$I(\mathbf{X};\mathbf{Y};\mathbf{Z}) = I(\mathbf{X};\mathbf{Y}|\mathbf{Z}) - I(\mathbf{X};\mathbf{Y}),$$

where the regular mutual information values are estimated as in Section 4.4.2. Higher order mutual information analysis using cumulants is assessed in Section 5.5.2.

5.4.3 Evaluation of Generalizations

To grasp the differences between the four generalizations from Section 5.4.2, Figure 5.2 (inspired by [BGP+11]) depicts their information diagrams. The dark grey area represents the respective information measure. Note that $I(\mathbf{X};\mathbf{Y};\mathbf{Z})$ represents the negative value of the triangle in the center of the figure. As can be confirmed visually, the generalizations are related as follows:

$$I(\mathbf{X};(\mathbf{Y},\mathbf{Z})) = I(\mathbf{X};\mathbf{Y}) + I(\mathbf{X};\mathbf{Z}) + I(\mathbf{X};\mathbf{Y};\mathbf{Z}); \tag{5.9}$$

$$C(\mathbf{X},\mathbf{Y},\mathbf{Z}) = I(\mathbf{X};(\mathbf{Y},\mathbf{Z})) + I(\mathbf{Y};\mathbf{Z}); \tag{5.10}$$

$$D(\mathbf{X},\mathbf{Y},\mathbf{Z}) = C(\mathbf{X},\mathbf{Y},\mathbf{Z}) + I(\mathbf{X};\mathbf{Y};\mathbf{Z}). \tag{5.11}$$

Distinguishing Properties

In practice, the random variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ represent $\mathbf{H}_{\hat{k}}$, $\mathbf{O}_{t_1}$ and $\mathbf{O}_{t_2}$ respectively. This particular realization is chosen to obtain symmetry in t_1 and t_2 in every generalization.

From Equation 5.10, one observes

$$C(\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}, \mathbf{O}_{t_2}) = I(\mathbf{H}_{\hat{k}}; (\mathbf{O}_{t_1}, \mathbf{O}_{t_2})) + I(\mathbf{O}_{t_1}; \mathbf{O}_{t_2}).$$

However, $I(\mathbf{O}_{t_1}; \mathbf{O}_{t_2})$, the mutual information of the two observations, is independent from the key guess in a masked implementation. Hence, for fixed t_1 and t_2:

$$C(\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}, \mathbf{O}_{t_2}) = I(\mathbf{H}_{\hat{k}}; (\mathbf{O}_{t_1}, \mathbf{O}_{t_2})) + c, \tag{5.12}$$

where c is a nonnegative constant. Consequently, the ranking of keys produced with $C(\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}, \mathbf{O}_{t_2})$ as distinguisher is theoretically equal to the ranking produced with $I(\mathbf{H}_{\hat{k}}; (\mathbf{O}_{t_1}, \mathbf{O}_{t_2}))$.

Moreover, in case of a totally random generation of masks from a perfectly uniform distribution, $\mathbf{O}_{t_1}$ and $\mathbf{O}_{t_2}$ are unrelated and the constant $c = I(\mathbf{O}_{t_1}; \mathbf{O}_{t_2})$ is equal to zero and $C(\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}, \mathbf{O}_{t_2}) = I(\mathbf{H}_{\hat{k}}; (\mathbf{O}_{t_1}, \mathbf{O}_{t_2}))$. The key candidate ranking of both methods are theoretically the same; moreover, the mutual information values are equal for every candidate.

With the assumption of a perfectly random masking scheme, one can derive more interesting properties. For example, $\mathbf{H}_{\hat{k}}$ and $\mathbf{O}_{t_1}$, as well as $\mathbf{H}_{\hat{k}}$ and $\mathbf{O}_{t_2}$ are independent. Hence, mutual information values $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_{t_1})$ and $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_{t_1})$ have no contribution and are theoretically equal to zero. This implies that all generalizations reduce to one type: interaction information. However, some generalizations do take mutual information of two variables into account. This is beneficial when information leaks via mutual information of two variables due to nonuniform masking generation. It might also be a drawback when mutual information of two random variables is merely caused by noise. The distinguishing properties of the methods are assessed on actual implementations in Chapter 7.

To name one final observation, Equation 5.11 indicates that the dual total correlation is the sum of two other models. Hence, each key candidate gets a score that is equal to the sum of the scores given in the other two models. That is, the scoring represents the average score of those two models (multiplied by two). Hence, dual total correlation can be seen as the average of total correlation and interaction information. This means that candidates that are ranked highly in both total correlation and interaction information, will be ranked highly in dual total correlation.

Negative Mutual Information

When examining the formulae for higher order mutual information, one can observe that interaction information can attain both positive and negative values. This happens because the effect of fixing one of the variables may decrease or increase dependence between the other variables. For both cases, an example is presented here. Recall that $I(\mathbf{X}; \mathbf{Y}; \mathbf{Z}) = I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) - I(\mathbf{X}; \mathbf{Y})$.

On the one hand, regard the situation where two factors, $\mathbf{X}$ and $\mathbf{Y}$, have a common cause $\mathbf{Z}$. Hence, $\mathbf{Z}$ might partially account for the relation between $\mathbf{X}$ and $\mathbf{Y}$. Consequently $I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) \leq I(\mathbf{X}; \mathbf{Y})$ and hence $I(\mathbf{X}; \mathbf{Y}; \mathbf{Z}) \leq 0$. In words: if one knows $\mathbf{Z}$, this reduces the mutual information $\mathbf{X}$ and $\mathbf{Y}$ share due to their respective dependence on $\mathbf{Z}$.

On the other hand, consider the situation where a factor $\mathbf{Z}$ has (at least) two causes, $\mathbf{X}$ and $\mathbf{Y}$. Hence, $\mathbf{Z}$ might partially depend on both $\mathbf{X}$ and $\mathbf{Y}$. A typical example is $\mathbf{Z} = \mathbf{X} \oplus \mathbf{Y}$. If $\mathbf{X}$ and $\mathbf{Y}$ are independent, their mutual information equals zero. However, conditioning on $\mathbf{Z}$, information on $\mathbf{X}$ and $\mathbf{Y}$ appears. In the XOR example: conditioning on the output $\mathbf{Z}$, the value $\mathbf{X}$ is even completely determined by $\mathbf{Y}$. Hence, the mutual information of all three is positive.

Differential power analysis suits the latter case. Two variables masked with the same mask have a boolean relation, their sum equals the unmasked predicted value under correct subkey hypothesis. Consequently, interaction information yields positive values for correct key hypotheses.

Method	Expression	Passes
Interaction information	$H(\mathbf{X},\mathbf{Y}) + H(\mathbf{Y},\mathbf{Z}) + H(\mathbf{X},\mathbf{Z}) -$ $(H(\mathbf{X}) + H(\mathbf{Y}) + H(\mathbf{Z}) + H(\mathbf{X},\mathbf{Y},\mathbf{Z}))$	7
Generalized MI	$H(\mathbf{X}) + H(\mathbf{Y},\mathbf{Z}) - H(\mathbf{X},\mathbf{Y},\mathbf{Z})$	3
Total correlation	$H(\mathbf{X}) + H(\mathbf{Y}) + H(\mathbf{Z}) - H(\mathbf{X},\mathbf{Y},\mathbf{Z})$	4
Dual total correlation	$H(\mathbf{X}) + H(\mathbf{Y}) + H(\mathbf{Z}) - H(\mathbf{X},\mathbf{Y},\mathbf{Z})$	4

Table 5.1: Number of passes through a three-dimensional histogram for each higher order mutual information generalization.

Computational Complexity

Higher order generalizations are used to estimate the mutual information of a collection of variables, based on multiple probability densities. As in practice densities might be estimated by means of multidimensional histograms, one might wonder which generalization requires the least number of computations in a histogram.

Here, this question is answered for the case of three random variables: $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. The number of bins equals u, v and w respectively. An overview of the required entropies for each generalization is given in Table 5.1.

Computing $H(\mathbf{X})$ consists of u times $v \cdot w$ additions, hence one must run through the complete histogram. The same holds for $H(\mathbf{X},\mathbf{Y})$ and $H(\mathbf{X},\mathbf{Y},\mathbf{X})$ and by symmetry for every entropy. Hence, the number of passes only depends on the number of distinct entropies in the expression. Table 5.1 indicates the number of passes for every MI generalization.

5.5 Multivariate Mutual Information Estimation Models

One crucial step in estimating mutual information is estimation of probability distributions. In Chapter 4, histogram density estimation is explained for the bivariate case. Moreover, a direct mutual estimation method using cumulants is explained. For both of these methods, a generalization to higher orders is described in this section. Furthermore, a brief overview of other possible density estimation methods used in literature is given.

Most of the information on higher order histograms and cumulants extends the concepts introduced in Section 4.4.1 and Section 4.4.2, respectively.

5.5.1 Histogram Model

The histogram model can be extended to higher orders easily. Where $I(\mathbf{X};\mathbf{Y})$ is estimated using a two-dimensional histogram, $I(\mathbf{X}_1;\ldots;\mathbf{X}_n)$ is estimated by means of a joint probability distribution arising from an n-dimensional histogram.

Theoretical Foundation

Let $\mathbf{X}_i$ be a random variable over $\mathcal{X}_i$ for $i = 1,\ldots,n$. In order to obtain a probability density function based on an n-dimensional histogram, the following steps have to be taken.

- Analogous to the two-dimensional case, one has to decide on the number of bins u_i for each of the n random variables. For all $i = 1,\ldots,n$, the bins of $\mathcal{X}_i$ are again assumed to be equally sized, a bin has size $|\mathcal{X}_i|/u_i$.
- Having decided these numbers, one needs to divide $\mathcal{X}_1$ into u_1 bins, $\mathcal{X}_2$ into u_2 bins, et cetera. An n-dimensional bin $B(i_1,\ldots,i_n)$ can be represented as follows:

$$B(i_1,\ldots,i_n) = B(i_1) \times \cdots \times B(i_n) \subset \mathcal{X}_1 \times \cdots \times \mathcal{X}_n,$$

  for each $(i_1,\ldots,i_n) \in \{1,\ldots,u_1\} \times \cdots \times \{1,\ldots,u_n\}$.

- Next, perform q observations $\theta_1, \ldots, \theta_q$ from $\mathbf{X}_1$, q observations $\zeta_1, \ldots, \zeta_q$ from $\mathbf{X}_2$, et cetera. As described in Section 4.4.1, for each j there is exactly one bin $B(i_1) \subseteq \mathcal{X}_1$ such that $\theta_j \in B(i_1)$. Analogously, for all j there is exactly one bin $B(i_2) \subseteq \mathcal{X}_2$ such that $\zeta_j \in B(i_2)$, et cetera. Again a function F can be defined as:

$$F(i_1, \ldots, i_n) = |\{j : \theta_j \in B(i_1), \zeta_j \in B(i_2), \ldots\}|.$$

  In words, F assigns an integer to an n-dimensional bin $B(i_1, \ldots, i_n)$ representing the number of observations $(\theta_j, \zeta_j, \ldots) \in B(i_1) \times B(i_2) \times \cdots \times B(i_n)$.

- For $(i_1, \ldots, i_n) \in \{1, \ldots, u_1\} \times \cdots \times \{1, \ldots, u_n\}$, define the empirical probability $p(i_1, \ldots, i_n)$ of an observation ending up in bin $B(i_1, \ldots, i_n)$ as

$$p(i_1, \ldots, i_n) = \frac{F(i_1, \ldots, i_n)}{q}.$$

The empirical joint probability function $p = f_{\mathbf{X}_1, \ldots, \mathbf{X}_n}$ can be used to calculate every marginal or joint distribution of (a subset of) the $\mathbf{X}_i$'s. From these distributions, single and joint entropies can be estimated. Depending on the higher order mutual information generalization, multiple entropy expressions can be combined to yield the desired higher order mutual information estimate.

Model

When a masked implementation with $r - 1$ masks is assumed, the general model consists of $r + 1$ random variables, being one prediction and r observations. This chapter focuses at second order attacks, so from now on a masked implementation with one mask is assumed, i.e. $r = 2$.

As before, prediction under key guess $\hat{k}$ is denoted by $\mathbf{H}_{\hat{k}}$, observations at times t_1 and t_2 are denoted by $\mathbf{O}_{t_1}$ and $\mathbf{O}_{t_2}$ respectively. For explanation of the model, $\hat{k}$, t_1 and t_2 are assumed to be fixed. However, an attack involves every possible key guess and probably multiple time combinations. Hence, the procedure described here needs to be performed multiple times, for different $\hat{k}$, t_1 and t_2.

First, the attacker decides on u, v_1 and v_2, the number of bins for prediction, first and second observation respectively. One likely choice is to set $v_1 = v_2$, this corresponds to the assumption in first order MIA that the number of bins for observation is equal for all t. Since an attacker has no information about the probability distribution of power consumptions at any time point t, there is no reason to choose different numbers of bins for observation.

After determining the bounds of each bin $B(i, j, k)$ $(i = 1, \ldots, u; j = 1, \ldots, v_1; k = 1, \ldots, v_2)$, the measurements can be placed in the appropriate bins. When all traces are analyzed and all measurements are placed in the corresponding three-dimensional bin, one can estimate the empirical joint probability function by dividing every integer by q, the number of traces.

The histogram model equips an attacker with a histogram-based estimation of the joint distribution function $f_{\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}, \mathbf{O}_{t_2}}$, from which any joint or marginal distribution can be derived. Univariate, bivariate and trivariate entropies can be calculated by means of the appropriate marginal or joint distribution, obtained from the histogram method. Depending on the selected higher order mutual information expression, an attacker computes the following values.

- In case of the generalized mutual information, one estimates:

$$I(\mathbf{H}_{\hat{k}}; (\mathbf{O}_{t_1}, \mathbf{O}_{t_2})) = I(\mathbf{O}_{t_1}; \mathbf{H}_{\hat{k}}) + I(\mathbf{O}_{t_2}; \mathbf{H}_{\hat{k}}) - I(\mathbf{H}_{\hat{k}}; \mathbf{O}_{t_1}; \mathbf{O}_{t_2}) \tag{5.13}$$
$$= H(\mathbf{O}_{t_1}, \mathbf{O}_{t_2}) + H(\mathbf{H}_{\hat{k}}) - H(\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}, \mathbf{O}_{t_2}). \tag{5.14}$$

  One single entropy, one joint entropy of two variables and one joint entropy of three variables have to be estimated.

- In case of the information interaction, one estimates:

$$I(\mathbf{H}_{\hat{k}};\mathbf{O}_{t_1};\mathbf{O}_{t_2}) = I(\mathbf{O}_{t_1};\mathbf{O}_{t_2}|\mathbf{H}_{\hat{k}}) - I(\mathbf{O}_{t_1};\mathbf{O}_{t_2}) \tag{5.15}$$

$$\begin{aligned} &= H(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_1}) + H(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_2}) + H(\mathbf{O}_{t_1},\mathbf{O}_{t_2}) \\ &- H(\mathbf{H}_{\hat{k}}) - H(\mathbf{O}_{t_1}) - H(\mathbf{O}_{t_2}) - H(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_1},\mathbf{O}_{t_2}). \end{aligned} \tag{5.16}$$

  Entropies of the $2^3 - 1$ nonzero combinations of the three random variables have to be estimated.

- In case of the total correlation, one estimates:

$$C(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_1},\mathbf{O}_{t_2}) = H(\mathbf{H}_{\hat{k}}) + H(\mathbf{O}_{t_1}) + H(\mathbf{O}_{t_2}) - H(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_1},\mathbf{O}_{t_2}). \tag{5.17}$$

  Only the marginal entropies and the entropy of the complete set is required here.

- In case of the dual total correlation, one estimates:

$$D(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_1},\mathbf{O}_{t_2}) = H(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_1}) + H(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_2}) + H(\mathbf{O}_{t_1},\mathbf{O}_{t_2}) - 2H(\mathbf{H}_{\hat{k}},\mathbf{O}_{t_1},\mathbf{O}_{t_2}). \tag{5.18}$$

  Only joint entropies are required here, no single entropies.

Evaluation

Histograms provide an estimate for a trivariate empirical probability density. Advantages and disadvantages of histograms are summarized in this section.

Like a two-dimensional histogram, no underlying distribution is assumed in three-dimensional histograms. Hence, for second order analysis, histograms provide an assumption-free estimate of probability distributions. However, the number of bins in each dimension needs to be determined in advance. This decision is of crucial interest; using many bins might increase the vulnerability for noise, since a noisy observation might easily be assigned to a wrong bin. In contrast, using too few bins might average measurements too much, yielding no significant distinction properties. Since no complex calculations (such as integration) are involved in three-dimensional entropy estimation using histograms, it is a fast method to estimate a three-dimensional joint probability distributions.

Assumptions made on histograms in first order analysis also hold here. For example, since one needs to known the absolute bounds for observation, filtering for outliers must be executed before a histogram is constructed.

In many papers on second order mutual information analysis (including Prouff and Rivain [PR09] and Gierlichs et al. [GBPV10]) histograms are used for density estimation. A disadvantage of histograms in side channel analysis context is that, like in first order attacks, no overall best bin decision can be made. Depending on the underlying distribution of observations, other density methods might be more appropriate; a histogram could show a relatively slow convergence. However, this requires knowledge of the underlying distribution. Moreover, to adequately model a distribution via a histogram, more measurements are required than, for instance, by the calculation of a correlation coefficient. However, in case of nonlinear or unpredictable behavior, a histogram is a nonparametric and fast way to adequately model an empirical density.

5.5.2 Cumulant Model

Cumulant-based mutual information estimation is easily generalized to higher orders. Mixed cumulants of multiple random variables exist and can be combined in a natural way, as indicated by Le and Berthier [LB10]. However, a near-Gaussian underlying distribution for prediction and observation is assumed.

Theoretical Foundation

In Theorem 4.4.3, the following result is proven. Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$ be an n-dimensional random variable, where each component follows a nearly standard normal distribution. Write $f_{\mathbf{X}}(x_1, \ldots, x_n)$ as a shorter notation for the joint distribution $f_{\mathbf{X}_1,\ldots,\mathbf{X}_n}(x_1, \ldots, x_n)$ and denote $\mathbf{Y} = \mathbf{X}_1 \cdot \ldots \cdot \mathbf{X}_n$, so $f_{\mathbf{Y}}(x_1, \ldots, x_n) = f_{\mathbf{X}_1}(x_1) \cdot \ldots \cdot f_{\mathbf{X}_n}(x_n)$. Denote $W = \{1, \ldots, n\}$ and define the following index sets:

$$I_2 = W^2 \backslash \{(i,i) : i \in W\};$$
$$I_3 = W^3 \backslash \{(i,i,i) : i \in W\};$$
$$I_4 = W^4 \backslash \{(i,i,i,i) : i \in W\}.$$

The mutual information between the components of $\mathbf{X}$ can be estimated as follows.

$$I(\mathbf{X}_1; \ldots; \mathbf{X}_n) \approx \frac{1}{4} \sum_{i,j \in I_2} (R_{i,j}(\mathbf{X}))^2 + \frac{1}{12} \sum_{i,j,k \in I_3} (T_{i,j,k}(\mathbf{X}))^2 + \frac{1}{48} \sum_{i,j,k,l \in I_4} (Q_{i,j,k,l}(\mathbf{X}))^2. \quad (5.19)$$

In case of second order analysis, mutual information of the three random variables $\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}$ and $\mathbf{O}_{t_2}$ is estimated. As showed in Lemma 4.4.1, cumulant-based mutual information approximation is based on estimation of the Kullback-Leibler divergence. Consequently, cumulant-based higher order mutual information resembles total correlation, see Definition 2.2.9.

Alternatively, one can estimate mutual information using Equation 5.4:

$$I(\mathbf{H}_{\hat{k}}; \mathbf{O}_{t_1}; \mathbf{O}_{t_2}) = I(\mathbf{O}_{t_1}; \mathbf{O}_{t_2} | \mathbf{H}_{\hat{k}}) - I(\mathbf{O}_{t_1}; \mathbf{O}_{t_2}),$$

where first order (conditional) mutual information values are estimated via cumulants, as explained in Section 4.4.2. This expression resembles interaction information. However, Le and Berthier [LB10] indicate that this method is less efficient, especially when noise is low or moderate. Moreover, as compared to the expression in 5.19, this expression includes much more terms. This makes estimation slow and memory-consuming. Therefore, this method is not further investigated in this thesis. However, in the future it might be interesting to compare this method to total correlation using cumulants, as well as to interaction information using histograms.

Model

A collection of q traces yields q predictions $h_{1,\hat{k}}, \ldots, h_{q,\hat{k}}$ under key guess $\hat{k}$, as well as q observations $o_{1,t}, \ldots, o_{q,t}$ at time t. Denote the normalized prediction under key guess $\hat{k}$ by $\tilde{\mathbf{H}}_{\hat{k}}$ and the normalized observation at time t by $\tilde{\mathbf{O}}_t$, i.e.

$$\tilde{\mathbf{H}}_{\hat{k}} = \frac{\mathbf{H}_{\hat{k}} - m_{\mathbf{H}_{\hat{k}}}}{s_{\mathbf{H}_{\hat{k}}}}, \tilde{\mathbf{O}}_t = \frac{\mathbf{O}_t - m_{\mathbf{O}_t}}{s_{\mathbf{O}_t}}.$$

Here, $m_{\mathbf{X}}$ represents the mean of $\mathbf{X}$ and s_X represents the standard deviation of $\mathbf{X}$.

In order to approximate $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_{t_1}; \mathbf{O}_{t_2})$ using Theorem 4.4.3, the summands are taken over $(\mathbf{X}_1, \ldots, \mathbf{X}_n) = (\mathbf{H}_{\hat{k}}, \mathbf{O}_{t_1}, \mathbf{O}_{t_2})$. The cumulants R, T and Q can be expressed in terms of trivariate joint moments $\mu_{a,b,c}$ to obtain:

$$\sum_{i,j\in I_2}(R_{i,j})^2 = \binom{2}{1}(E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_{t_1}]^2 + E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_{t_2}]^2 + E[\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}_{t_2}]^2) = 2(\mu^2_{1,1,0} + \mu^2_{1,0,1} + \mu^2_{0,1,1}); \tag{5.20}$$

$$\begin{aligned}\sum_{i,j,k\in I_3}(T_{i,j,k})^2 =& \binom{3}{1}(E[\tilde{\mathbf{H}}^2_{\hat{k}}\tilde{\mathbf{O}}_{t_1}]^2 + E[\tilde{\mathbf{H}}^2_{\hat{k}}\tilde{\mathbf{O}}_{t_2}]^2 + E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}^2_{t_1}]^2 + E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}^2_{t_2}]^2 + E[\tilde{\mathbf{O}}^2_{t_1}\tilde{\mathbf{O}}_{t_2}]^2 + \\ & E[\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}^2_{t_2}]^2 + E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}_{t_2}]^2 \\ =& 3(\mu^2_{2,1,0} + \mu^2_{2,0,1} + \mu^2_{1,2,0} + \mu^2_{1,0,2} + \mu^2_{0,2,1} + \mu^2_{0,1,2} + \mu^2_{1,1,1});\end{aligned} \tag{5.21}$$

$$\sum_{i,j,k,l\in I_4}(Q_{i,j,k,l})^2 = \binom{4}{1}Q_1 + \binom{4}{2}Q_2 + 2\cdot\binom{4}{3}Q_3, \tag{5.22}$$

where

$$\begin{aligned}Q_1 =& (-3\mu_{2,0,0}\mu_{1,1,0} + \mu_{3,1,0})^2 + (-3\mu_{1,1,0}\mu_{0,2,0} + \mu_{1,3,0})^2 + (-3\mu_{2,0,0}\mu_{1,0,1} + \mu_{3,0,1})^2 + \\ & (-3\mu_{0,2,0}\mu_{0,1,1} + \mu_{0,3,1})^2 + (-3\mu_{1,0,1}\mu_{0,0,2} + \mu_{1,0,3})^2 + (-3\mu_{0,1,1}\mu_{0,0,2} + \mu_{0,1,3})^2;\end{aligned} \tag{5.23}$$

$$\begin{aligned}Q_2 =& (-2(\mu_{1,1,0})^2 - \mu_{2,0,0}\mu_{0,2,0} + \mu_{2,2,0})^2 + (-2(\mu_{1,0,1})^2 - \mu_{2,0,0}\mu_{0,0,2} + \mu_{2,0,2})^2 + \\ & (-2(\mu_{0,1,1})^2 - \mu_{0,2,0}\mu_{0,0,2} + \mu_{0,2,2})^2;\end{aligned} \tag{5.24}$$

$$\begin{aligned}Q_3 =& (-2\mu_{1,1,0}\mu_{1,0,1} - \mu_{2,0,0}\mu_{0,1,1} + \mu_{2,1,1})^2 + (-\mu_{0,2,0}\mu_{1,0,1} - 2\mu_{1,1,0}\mu_{0,1,1} + \mu_{1,2,1})^2 + \\ & (-2\mu_{1,0,1}\mu_{0,1,1} - \mu_{1,1,0}\mu_{0,0,2} + \mu_{1,1,2})^2.\end{aligned} \tag{5.25}$$

The mutual information value can be estimated directly from the data, by using

$$E[\tilde{\mathbf{H}}^a_{\hat{k}}\tilde{\mathbf{O}}^b_{t_1}\tilde{\mathbf{O}}^c_{t_2}] = \mu_{a,b,c} \approx m_{a,b,c} = \frac{1}{q}\sum_{i=1}^{q}(\tilde{h}_{i,\hat{k}})^a(\tilde{o}_{i,t_1})^b, (\tilde{o}_{i,t_2})^c,$$

where $m_{a,b,c}$ is the a,b,c-th joint sample moment of $\tilde{\mathbf{H}}_{\hat{k}}$, $\tilde{\mathbf{O}}_{t_1}$ and $\tilde{\mathbf{O}}_{t_2}$. Here $\tilde{h}_{i,\hat{k}}$ is the normalized prediction value under key guess $\hat{k}$ and $\tilde{o}_{i,t}$ is the normalized observation at time t, both arising from processing plaintext d_i. Using these sample moments in Equation 5.19, one finds the estimated mutual information value $I(\mathbf{H}_{\hat{k}};\mathbf{O}_{t_1};\mathbf{O}_{t_2})$. All other steps resemble those described in Section 4.4.2.

Evaluation

The same criteria for first order MIA using cumulants apply here. Thus, the same benefits and drawbacks are encountered as in Section 4.4.2.

Cumulants strongly rely on the Gaussian assumption. However, in general this assumption does not completely hold. A goodness of fit test can be incorporated in the analysis; however, no universal test seems to be applicable. For example, most tests require that more measurements yield a better approximation of the normal distribution. This does not hold in practice, since observations might follow a distribution that will never adequately approximate the normal distribution. Furthermore, the expression of trivariate mutual information consists of many terms. This strongly impacts the speed of calculation.

Note that $\sum_{i,j,k\in I_3}(T_{i,j,k})^2$ contains the term $\mu^2_{1,1,1}$. This term represents the squared product of three normalized random variables. This term is closely related to the correlation in combination with the normalized product combining function (Section 3.3.2). Le and Berthier [LB10] indicate that this value resembles absolute difference, in case the attack aims at one bit. Hence, this term gives a contribution related to a preprocessed correlation coefficient.

The parametric nature of this mutual information estimation method induces that it is more noise-resistant than histogram-based mutual information estimation, as indicated by Le and Berthier. As in the first order case, cumulant-based mutual information analysis is especially adequate when prediction and observation follow a near-normal distribution. However, an attacker might no know the distributions a priori.

5.5.3 Other

As explained for first order MIA, many other ways to estimate a probability density exist. A short overview of density estimation methods used on second order mutual information analysis is presented here. For a more broad treatments of density estimation methods, see Section 4.4.3.

Kernel density estimation is used in second order mutual information analysis by Batina et al. [BGP+11]. They use a Gaussian kernel with bandwidth chosen according to a specific rule of thumb. The results on simulated traces with different noise levels indicate the following properties.

In case of an adequate leakage model, histogram-based MIA and kernel-based MIA score equally in number of traces. For a less appropriate prediction model, or more noise, kernel-based MIA is more efficient. This might be due to the fact that a kernel estimation is more smooth than a noisy histogram, and hence some of the noise is averaged. However, the authors mention the importance of an adequate leakage model and density estimation method, but conclude that no general best decision can be made to apply for generic DPA attempts. Furthermore, no comparison of computation time is given.

In an early work of Prouff and Rivain [PR09] is described how to extend a parametric estimation to more than two variables. A Gaussian mixture is assumed as underlying distribution, having multidimensional parameters that must be estimated. However, only simulation results based on histograms are presented. Consequently, no conclusion about the benefits of parametric estimation in higher order MIA on practical implementations can be drawn.

The papers on higher order mutual information analysis are not numerous. Evidently, the probability density estimation models for first order analysis in Section 4.4.3 can be generalized to higher orders, to be used for higher order MIA. However, like in first order analysis, no universal best density estimation model can be appointed for HO-MIA.

5.6 Discussion

Mutual information analysis is generalized to higher orders more straightforwardly than correlation analysis. In this section, parallels between higher order MIA and higher order CPA are drawn, as well as differences are indicated. Furthermore, a thought experiment on higher order correlation analysis and higher order MIA is presented.

5.6.1 Second Order MIA versus Preprocessing in CPA

The preprocessing step is a limitation in generality of higher order CPA. The strength of the attack strongly relies on the accuracy of preprocessing, as well as on an accurate power model. The existence of multiple suggestions for and usages of preprocessing functions indicate that no universal best preprocessor is appointed yet.

As indicated in Section 4.5.2, mutual information attacks generally are less efficient than correlation attacks when a linearity assumption is fulfilled. However, as pointed out by two papers on HO-MIA, this is not the case in second order analysis.

Gierlichs et al. [GBPV10] present results on second order MIA and second order CPA, based on measurements from a masked implementation leaking information about Hamming weight. Using the normalized product preprocessor (see Section 3.3.2), second order CPA scores equivalent to second order MIA for a Hamming weight-based power model. No clear advantage of second order correlation analysis rises from these results.

Furthermore, both CPA using the absolute difference preprocessor and CPA using the normalized product preprocessor perform worse than a second order mutual information attack on a simulations leaking information about the identity of intermediate values, as presented by Prouff and Rivain [PR09]. Hence, preprocessing strongly affects performance. One might conclude that using correlation as distinguisher is only feasible in specific cases, where preprocessed variables do show linear relations with predictions. For example, Magnard et al. [MOP07] indicate that the absolute difference preprocessor is adequate on devices that leak information about Hamming weight. However, in practice an attacker has limited knowledge about information leakage through power consumption. An a priori best preprocessing function can henceforth not be appointed. This is in favor of general attacks like HO-MIA, lacking the need of preprocessing.

5.6.2 Higher Order CPA as Instance of Higher Order MIA

A successful first order correlation attack guarantees the existence of a successful mutual information attack, as argued in Section 4.5.3. Furthermore, mutual information examines linear and nonlinear relations between random variables, hence incorporates targets from CPA. In this section, a similar thought experiment is performed by investigating to what extent higher order correlation analysis might be viewed as an instance of higher order mutual information analysis.

One property of mutual information analysis is not shared by correlation analysis: the ease and straightforwardness to be generalized to higher orders. Whereas mutual information of multiple variables is defined in multiple ways, no straightforward extension of correlation coefficients exists. The need for a preprocessing function further reduces generality; where first order CPA assumes a linear relation between prediction and observation, a linear relation between prediction an a certain combination of observations is assumed in second order CPA. Since the truly multivariate nature of HO-MIA is not shared by HO-CPA, this obstructs the view of higher order correlation analysis as an instance of higher order mutual information analysis. However, information about a correlation attack can provide an attacker with information about a mutual information attack.

Alternative approach. Suppose a successful second order correlation attack exists, using absolute-difference as preprocessing function. One might wonder if this is a sufficient condition for a successful second order mutual information attack to exist.

In a successful absolute-difference correlation attack, predictions based on the correct key guess are correlated with the absolute difference of two masked variables. This implies that, given a prediction, few uncertainty exists about the absolute difference of two corresponding observations. When few uncertainty about the absolute difference exists, fixing one of the two values yields few uncertainty about the other value. Hence, given the prediction value, the mutual information of two observations is high. However, in a proper masking scheme, the mutual information of two observations (not conditioned on prediction) theoretically equals zero. Thus, a high conditioned mutual information value translates to a high interaction information value for the correct key guess.

As explained in Section 5.4.3, the four mutual information generalizations are theoretically equivalent in a perfectly masked implementation, where masks are drawn uniformly at random. Hence, a high score in interaction information theoretically implies a high score in every other method. However, other methods might be more vulnerable to noise under influence of mutual information values of two variables, or from highly dependent observations at incorrect time instances. This might be reflected in a need for more traces to yield success.

As conclusion, a successful second order correlation attack is a sufficient condition for a second order mutual information attack to exist. This conclusion is supported by results as presented in Chapter 7. There is no implementation, or simulation, that is successfully broken by a second order correlation attack, but not by a second order mutual information attack. However, the number of traces needed for a correct key retrieval might be different for different methods. Therefore, a successful second order correlation attack does not provide a lower bound, nor an upper bound, on the number of traces required for a successful second order mutual information attack.

Chapter 6

Implementation and Practical Aspects

Besides the highly interesting mathematical properties of mutual information analysis, it also yields practical benefits over existing methods. To obtain results on first order and higher order mutual information analysis as introduced in the previous chapters, an implementation has been written in Java. This module fits into Riscure's side channel analysis tool Inspector and benefits from other supporting methods for, e.g., user interface or score calculation.

In this chapter, the implementation of first order mutual information analysis is explained, followed by an explanation of the implementation of higher order mutual information analysis. As can be seen there, this is merely an extension of the first order implementation. The chapter is concluded by presenting design considerations.

6.1 Implementation of First Order MIA

In order to obtain practical results on genuine cryptographic embeddings, mutual information analysis is implemented in Java. In this section the implementation of first order mutual information analysis is explained.

6.1.1 Assumptions and Restrictions

Some assumptions on the input data are made, which are of crucial importance for a mutual information analysis to succeed.

- Power traces are assumed to be sufficiently aligned. That is, the action performed at time t corresponds to the same action in every trace.
- Traces are filtered for outliers by Riscure's filtering methods. In this way, no outliers can disturb the analysis. A filter for outliers consists of deleting every measurement that exceeds a certain threshold, which might depend on the standard deviation of the measurements at that specific point in time.
- The attacker has knowledge about the points in time where leakage occurs. This might be due to previous analysis of the traces, to a pattern in the trace or to knowledge from the implementation. This reduces the number of time points that have to be examined, and hence the computation time.

6.1.2 Starting the Analysis

One of the steps in differential power analysis, described in Section 3.1.3, consists of obtaining power traces; each trace corresponding to power consumption during the processing of one plaintext. Depending on whether the plaintext or the ciphertext is available, one can choose to attack

the first round or the last round of the cryptographic algorithm. From now on, it is assumed that the first round is attacked.

First, an attacker needs to select an intermediate value which is likely to have a relation with power consumption. This selection can, for example, be based on prior knowledge of cryptographic devices. Then, an attacker must select an interval in the power trace where information leakage about intermediate values is expected to occur, denote by l its length. Analysis is performed on this interval. An attacker might also decide to run an attack on a user specified part of the trace set, instead of attacking all available traces.

A power trace typically is an array of values, with a time-scale equal to the measurement resolution of the oscilloscope. An attack is performed on l time values, so trace values arising from plaintext d can be seen as the array $(o_{d,1}, \ldots, o_{d,l})$.

For the same plaintext d, one must predict the power consumption of a specific intermediate value, based on subkey guess $\hat{k}$. Denote by n the number of number of subkey guesses. Since the attack recovers the key part by part, the number n is the product of the number of round key parts and the number of candidates per part. For example, in case of an attack on an AES implementation, the number of key parts equals 16 and the number of candidates equals 256, hence $n = 4096$. The number n is determined by the algorithm and the selected intermediate value.

Besides selecting an appropriate interval to attack, an attacker needs to specify a power model according to which the predictions are made. The power model (see Section 3.2.2) translates predictions of the actual intermediate value to power consumption estimations.

Predictions can be seen as an array of n values $(h_{d,1}, \ldots, h_{d,n})$, for each plaintext d. Now, as described in Chapter 4, for each pair $(\hat{k}, t) \in \{1, \ldots, n\} \times \{1, \ldots, l\}$ one estimates $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t)$ using an estimation method.

General Input

In short, the following steps must be taken before to start a general mutual information analysis.

- Select an intermediate value and a power model.
- Select part of the power trace where information leakage is expected.
- Enter the number of traces on which analysis is performed.
- Select a mutual information estimation model.
- Select whether or not to normalize mutual information values. However, the use of normalized mutual information as distinguisher is discouraged, see Section 6.3.3.

In the following sections, method-specific information is given for histogram-based MIA and cumulant-based MIA respectively.

6.1.3 Histogram Method

Besides the general input, some algorithm-specific input is required. Moreover, specific computations are performed when estimating mutual information using histograms. This section summarizes the input that has to be provided, and the computations that need to be performed in order to execute a mutual information analysis using the histogram density estimation method.

Parameters

As explained in Section 4.4.1, only equally spaced bins are considered in this thesis. Furthermore, the number of bins for observation is assumed to be equal for all time points. However, one has to decide on this number of bins v for the observation random variable $\mathbf{O}_t$, sometimes called

resolution. This can be done in many ways, some of which are explained here. Recall that q is the number of traces.

Fixed decision. One makes an educated guess for the number of bins and uses that guess for every time instant, i.e. for each i and j in $\{1, \ldots, l\}$: $v_i = v_j$.

Scott's rule [Sco79]. $v_t = 3.49 s_t q^{-1/3}$ with s_t the sample standard deviation of $\mathbf{O}_t$. Note that the number of bins is not necessarily equal for all t, since the standard deviation of $\mathbf{O}_t$ is not necessarily equal for every t.

Sturges' rule [Stu26]. $v_t = 1 + \log q$. This rule is based on the idea that, given a histogram of a nearly normal distribution with v bins, the i'th bin should contain around $\binom{v-1}{i}$ observations. For large q, this histogram converges to a normal distribution. This rule yields the same number of bins for each t.

The first option is preferred for multiple reasons. One of the main reasons is that the number of traces is not fixed a priori. Suppose one performs an attack on q traces and bases the number of bins on (the standard deviation of) q observations. In case the attack is not successful and the attacker wishes to proceed with another q traces, he is forced to continue estimating a density using a number of bins based on (the standard deviation of) only the first q observations. For example, Sturges' rule indicates that one extra bin should be used when doubling the number of traces. Besides this argument, it might be useful to examine several choices for v to assess its influence.

The same procedure must be performed for the predictions. The educated guess is considered to be a valid decision rule here, since the number of distinct prediction values is known (and small) and the bins are equally spaced. Hence, the number of prediction bins u is chosen to be equal to the number of distinct prediction outcomes. This decision maximizes the amount of information coming from a histogram. Indeed, merging bins yields indifferentiability of distinct predictions, while adding bins has no contribution. This implies, for example, that since a Hamming weight model of a DES S-box output ranges from 0 to 4, the number of bins is chosen to be equal to 5. Analogously, since an identity model of an S-box output has 16 outcome possibilities, the number of bins is then chosen to be equal to 16.

The histogram method uses a multi-pass algorithm. During the first pass, the absolute bounds on $o_{d,t}$ are obtained, for each t. This can be done in three different ways.

User-specified. The upper and lower bounds are assigned manually. This requires knowledge of the output values and for each trace set, different values have to be initialized. However it is the easiest method in terms of calculation power; it makes the first pass actually obsolete. A naive solution is to assign the same bounds for each t. Clearly, this reduces the amount of information coming from a histogram, for observations at different time points are not likely to populate the exact same interval.

Sample search. During the first phase a random sample of the trace set, e.g. 10%, is inspected to determine the empirical bounds. This is the give-and-take solution. One has to bear in mind that there can exist measurements that exceed this empirical domain. These measurements have to be placed in the lowest or highest bin, depending on the nature of the outlier. Alternatively, they can be ignored if one can be sure that this does not heavily affect the density estimation.

Exhaustive search. During the first pass, all traces are scanned to determine the maximum and minimum value for each t. This requires all traces to pass and is hence the most intensive, but also most accurate method.

Since the computational complexity of bound determination is low, this pass does not consume much time. Hence, exhaustive search can be used. However, there are cases where the trace set is not completely available in advance. As a theoretical example, suppose one performs an attack on q traces. If the cipher key has not yet been compromised, one continues on another q traces.

This is repeated until the cipher key is recovered. In general one does not want to rerun the first q traces when deciding to continue the attack. Therefore, no new bounds are determined from the complete set of $2q$ traces. Hence, the bound determination method reduces from exhaustive to sample search, where the sample consists of the first q traces. However, this has not happened in practice.

Histograms can be saved to disk after analysis. Later, these stored histograms can be used as input for a new analysis, instead of a trace set. When analysis is performed on stored histograms, parameters of these stored histograms (such as number of bins) overwrite the user-specified parameters. Evidently, the number of bins is already fixed in a saved histogram.

Method-specific input

- An attacker must select the number of bins for observation. The number of bins for prediction is determined by the intermediate value and the power model; an attacker is not able to change this decision.
- The attacker can specify whether or not to store histograms.
- When histograms from a previous analysis have been stored, an attacker can choose to analyze these histograms again, instead of analyzing the current trace set. Whether histograms have been stored previously is not checked by the module; a user is responsible for this knowledge.

Computation

First, analysis of traces and construction of histograms is explained. Thereafter, analysis of previously stored histograms is treated.

Histogram construction. Construction of histograms involves the following computational steps. First, all traces are analyzed to determine upper bounds and lower bounds for observations at each point in time (Pass 1). Then, the bounds for each histogram bin are determined, as well as the number of phases in which histograms are constructed. Finally, the actual construction of the histograms takes place.

Pass 1

For each t, the module keeps track of the minimum and the maximum value of $o_{d,t}$ over all plaintexts. This gives the lower bounds $\alpha_1, \ldots, \alpha_l$ and the upper bounds $\omega_1, \ldots, \omega_l$ such that for all d, t: $\alpha_t \leq o_{d,t} \leq \omega_t$. The traces are assumed to be already filtered for outliers, so bound-determinations does not suffer from malicious traces.

Intermediate step 1

After the first pass, the upper bounds ω_t and lower bounds α_t on the observations are known for $t = 1, \ldots, l$, as well as the number of bins for observation. Hence, the bounds for each bin can be determined. The module assumes equally sized bins, so the bounds $b_t(i)$ are calculated using

$$b_t(i) = \alpha_t + i\frac{\omega_t - \alpha_t}{v}, i = 0, \ldots, v.$$

Hence, $b_t(0) = \alpha_t$ and $b_t(v) = \omega_t$. The closed observation interval $[a_t, \omega_t]$ is divided into bins according to:

$$[b_t(0), b_t(1)),\ [b_t(1), b_t(2)),\ \ldots,\ [b_t(v-1), b_t(v)].$$

The number of bins u arises from the prediction model and selected intermediate; this is equal to the number of possible outcomes. Since prediction values are integer, the prediction bins are of the form

$$\{0\}, \{1\}, \ldots, \{u-1\}.$$

Intermediate step 2

Since histograms need to be stored until all traces have been analyzed, this can require a large amount of memory. Specifically, a total of $n \times l \times u \times v$, (representing the number of key guesses, the number of time points, the number of bins for prediction and observation respectively) entries have to be stored jointly. Especially on trace sets involving many time points, this might overflow the available memory. Therefore, construction of histograms is split into phases, where each phase focuses on a subinterval of the l time points. Each phase corresponds to one pass of the traces; a part of the traces is examined in each phase.

In particular, in phase s ($s = 2, \ldots, w + 1$) the observations from time $t = (s - 2)T + 1$ to $t = (s - 1)T$ are examined. The variable T is chosen to be equal to 150, this decision balances the number of passes and the memory usage per phase. The total number of histogram construction phases w is determined by simply computing $\lceil l/T \rceil$.

Pass 2 to w+1

For each of the T time points and n key guesses, a histogram of size $u \times v$ is created. This (yet empty) histogram corresponds to frequency matrix F, as introduced in Section 4.4.1. Now for every trace, increase frequency matrix entry $F_{\hat{k},t}(i, j)$ by one if a prediction under key guess $\hat{k}$ goes to prediction bin i and observation at time t to observation bin j. Bin determination is implemented by both binary and linear search, an attacker can select either one. In the remainder of the analysis, linear search is used.

If the attacker has selected the option to save histograms to disk, the histograms from a phase are written to disk at this moment. When the last phase has ended, a separate file is written to disk, containing information about the saved histograms.

When all traces are processed, the matrix $F_{\hat{k},t}$ of each pair $(\hat{k}, t)$ is a two-dimensional frequency table. For each pair $(\hat{k}, t)$, the values of the empirical joint probability function are stored in the matrix $J_{\hat{k},t}$; they are obtained by dividing each entry of $F_{\hat{k},t}$ by the number of traces q. Next, the marginal probability density functions of prediction and observation are calculated from the joint density by adding up the rows, resp. columns. The model described in Section 4.4.1 is used to perform mutual information estimation.

A phase yields $T \times n$ mutual information estimations. These values are stored in the mutual information matrix M (see Definition 4.3.1), an array of size $n \times l$.

Stored histograms. In case an attacker has chosen to analyze previously stored histograms, a slightly different analysis is performed. First, the module reads information about the parameters of the stored histograms from a specific file. This provides the attacker with information about the number of bins, the number of traces analyzed and the specific range in time used for analysis. Most of the parameters of stored histograms (such as number of bins or number of traces) are already determined; parameters set by the user are ignored.

Next, the histograms are read piece by piece. After reading a histogram for a time-key guess combination, the corresponding mutual information value $I(\mathbf{H}_{\hat{k}}; \mathbf{O}_t)$ is stored in the mutual information matrix M, from which later a candidate ranking is constructed.

Method-specific Output

As described above, an attacker can choose to save all histograms to disk. This allows him to further analyze the histograms on specific properties, like shape. No further method-specific output is generated by the histogram method.

6.1.4 Cumulant Method

The cumulant method is more than just a different density type. It is actually a complete other way of computation. This section summarizes the computations that need to be performed in order to perform a mutual information analysis using the cumulant computation method.

Parameters and Method-specific Input

As explained in Section 4.4.2, the cumulant method is a parametric estimation model where near-Gaussian distributions of prediction and observation are assumed. The parameters for these near-Gaussian distributions must be estimated from the traces. No specific further input is required for the cumulant method.

Computation

The cumulant method works with normalized values, so a normalization is included. Therefore, a two-pass algorithm is used, where the mean and standard deviation are computed during the first pass, and the calculation with normalized values can take place during the second pass. However, computing the mean and standard deviation might not be the only two objectives of the first phase.

First pass

During the first pass, the sample means $m_{\mathbf{H}_{\hat{k}}}$ and sample standard deviations $s_{\mathbf{H}_{\hat{k}}}$ of the predictions are calculated for each $\hat{k}$. Besides that, the sample means $m_{\mathbf{O}_t}$ and sample standard deviations $s_{\mathbf{O}_t}$, as well as the third and fourth raw moment of the observations are calculated. During the second pass, the normalization is computed using the mean and standard deviation. The third and fourth moment can be useful to test for normality.

In order to achieve numerical stability, the moments are determined incrementally according to Pébay [Péb08]. If X is a set of size n with mean μ_1 and x is added to the set, yielding $X' = X \cup \{x\}$, then

$$\mu_1' = \mu_1 + \frac{x - \mu_1}{n+1}.$$

The second central moment can be estimated by:

$$\mu_2' = \mu_2 + \frac{(x - \mu_1)(x - \mu_1')}{n+1}.$$

The computation of the third and fourth moment can be found in Pébay [Péb08].

Second pass

During the second pass, observations are treated in their normalized form, i.e.

$$\tilde{h}_{d,\hat{k}} = \frac{h_{d,\hat{k}} - m_{\mathbf{H}_{\hat{k}}}}{s_{\mathbf{H}_{\hat{k}}}} \text{ and } \tilde{o}_{d,t} = \frac{o_{d,t} - m_{\mathbf{O}_t}}{s_{\mathbf{O}_t}}.$$

The cumulant method assumes samples to have a probability distribution close to the standard normal distribution. Hence, two routines are executed during the second pass: feasibility testing and incremental moment updating.

- Testing for feasibility
  At certain moments in time, the power consumption might be constant. Trivially, power consumption at these points in time is independent from the data processed. Since the information arising from such samples is minimal, these samples can be omitted for the sake of a reduction in computation time noise-minimization. In the implementation, the points in time t where $s_{\mathbf{O}_t} = 0$ are neglected.

  Besides observations with zero standard deviation, a test for normality can be included. For example, a test introduces by Jarque and Bera [JB80] is based on the kurtosis of the distribution, which can easily be obtained in the first pass. However, many normality tests assume convergence towards a proper Gaussian form, when the number of observations increase. This assumption is not feasible in side channel analysis, since the underlying distribution of observation might not be equal to a Gaussian function. Consequently, convergence to a normal distribution is never reached adequately. For this reason, no test for normality is implemented.

- Update incremental moments
  For each $\hat{k}$ and t, several (mixed) sample moments of $\tilde{\mathbf{H}}_{\hat{k}}$ and $\tilde{\mathbf{O}}_t$ are computed incrementally using the pairs $(\tilde{h}_{d,\hat{k}}, \tilde{o}_{d,t})$, $d = 1, \ldots, q$. The moments required for estimation of mutual information are derived from Equation 4.9:

$$\begin{array}{lllll} E[\tilde{\mathbf{H}}_{\hat{k}}] & E[\tilde{\mathbf{H}}_{\hat{k}}^2] & E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t] & E[\tilde{\mathbf{H}}_{\hat{k}}^2\tilde{\mathbf{O}}_t] & E[\tilde{\mathbf{H}}_{\hat{k}}^3\tilde{\mathbf{O}}_t] \\ E[\tilde{\mathbf{O}}_t] & E[\tilde{\mathbf{O}}_t^2] & E[\tilde{\mathbf{H}}_{\hat{k}}^2\tilde{\mathbf{O}}_t^2] & E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t^2] & E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t^3] \end{array}$$

  Since the means of both $\tilde{\mathbf{H}}_{\hat{k}}$ and $\tilde{\mathbf{O}}_t$ are equal to zero, the incremental update of the a, b'th mixed first moment $\mu_{a,b}$ of $d-1$ samples to $\mu'_{a,b}$, where the pair $(h_{d,\hat{k}}, o_{d,t})$ is added to the set, is performed as follows:

$$\mu'_{a,b} = \mu_{a,b} + \frac{(\tilde{h}_{d,\hat{k}})^a(\tilde{o}_{d,t})^b - \mu_{a,b}}{d}.$$

  The value of $\mu_{a,b}$ is an estimation of $E[\tilde{\mathbf{H}}_{\hat{k}}^a\tilde{\mathbf{O}}_t^b]$. These mixed first moments are estimated for each $\hat{k}$ and t.

Score

For each $\hat{k}$ and t, the mutual information value is estimated in the following way, and stored in the mutual information matrix M.

First, the auxiliary intermediates R, T, Q_1 and Q_2, based on Equation 4.9, are computed.

$$\begin{aligned} R &= E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t]^2 = \mu_{1,1}^2; \\ T &= E[\tilde{\mathbf{H}}_{\hat{k}}^2\tilde{\mathbf{O}}_t]^2 + E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t^2]^2 = \mu_{2,1}^2 + \mu_{1,2}^2; \\ Q_1 &= (E[\tilde{\mathbf{H}}_{\hat{k}}^3\tilde{\mathbf{O}}_t] - 3E[\tilde{\mathbf{H}}_{\hat{k}}^2]E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t])^2 + (E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t^3] - 3E[\tilde{\mathbf{O}}_t^2]E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t])^2 \\ &= (\mu_{3,1} - 3\mu_{2,0}\mu_{1,1})^2 + (\mu_{1,3} - 3\mu_{0,2}\mu_{1,1})^2; \\ Q_2 &= (E[\tilde{\mathbf{H}}_{\hat{k}}^2\tilde{\mathbf{O}}_t^2] - E[\tilde{\mathbf{H}}_{\hat{k}}^2]E[\tilde{\mathbf{O}}_t^2] - 2(E[\tilde{\mathbf{H}}_{\hat{k}}\tilde{\mathbf{O}}_t])^2)^2 = (\mu_{2,2} - \mu_{2,0}\mu_{0,2} - 2\mu_{1,1}^2)^2. \end{aligned}$$

Then the mutual information value is estimated using the identity from Equation 4.9:

$$I(\tilde{\mathbf{H}}_{\hat{k}}, \tilde{\mathbf{O}}_t) = c\left(\frac{R}{2} + \frac{T}{4} + \frac{Q_1}{12} + \frac{Q_2}{8}\right).$$

The constant c equals $\frac{1}{\log 2}$, to convert the logarithms from base e to base 2. Note that, in case one is only interested in the highest mutual information occurrence of a set (and not the particular value), taking $c = 1$ suffices.

6.1.5 General output

The mutual information matrix M is constructed at the end of the analysis, a subkey candidate ranking is made based on this matrix. The key guesses for each subkey are ranked, where the highest mutual information value corresponds to the highest rank. An attacker can specify how many top-ranked candidates per subkey are displayed at the end of the analysis. The most likely cipher key guess is then obtained by reverse-engineering the cipher key from the subkeys in the first round.

To provide an example, consider an attack on the AES S-box output in the first round. For each of the 16 state bytes, 256 candidates exist. These 256 candidates are ranked based on the corresponding mutual information value, for each of the 16 subkeys. The most likely cipher key guess then equals the concatenation of the 16 top-ranked candidates.

6.2 Implementation of Higher Order MIA

Besides functionality to perform first order mutual information analysis, the module developed for Riscure's Inspector is also able to perform second order MIA, jointly evaluating observations from $\upsilon = 2$ time points. Higher order attacks ($\upsilon > 2$) are commonly believed not practical due to heavy computational costs. In smart cards security testing standards, only second order attacks are required to be tested for high order side channel analysis. Hence, implementations in this section focus at mutual information attacks on one prediction value and two observation values.

6.2.1 Assumptions and Restrictions

Some assumptions on the input data are made, which are of crucial importance for a second order mutual information analysis to succeed. Most of these assumptions resemble those presented for first order analysis.

- Power traces are assumed to be sufficiently aligned. That is, the action performed at time t corresponds to the same action in every trace.
- Traces are filtered for outliers by Riscure's filtering methods. In this way, no outliers can disturb the analysis. A filter for outliers consists of deleting every measurement that exceeds a certain threshold, which might depend on the standard deviation of the measurements at that specific point in time.
- The attacker has knowledge about the points in time where leakage of masked values occurs. This might be due to previous analysis of the traces, to a pattern in the trace or to knowledge from the implementation. This reduces the number of time points that have to be examined, and hence the computation time. More specific, an attacker knows the minimum distance and the maximum distance between the points in time where the two masked variables are processed.

6.2.2 Starting the Analysis

One of the steps in second order differential power analysis, described in Section 3.1.3, consists of obtaining power traces, each trace corresponding to power consumption during the processing of one plaintext. As in first order analysis, it is assumed that the first round is attacked.

First, an attacker needs to select an intermediate value which is likely to have a relation with two power consumption values. This selection can, for example, be based on prior knowledge of cryptographic devices. Then, an attacker must select an interval in the power trace where information leakage about intermediate values is expected to occur. The length of this interval is called l. For a masked implementation with $\upsilon - 1$ masks, as many as $\binom{l}{\upsilon}$ possible combinations of υ observations can be examined. Narrowing down the interval to only useful time instances is highly recommended.

An attacker has to specify a mask distance δ_m and a mask range δ_r. This represents the minimum distance between two leakages, and the range within which this second leakage is expected to occur. This reduces the number of time-pairs from $\binom{l}{2}$ to $\delta_r(l - \delta_m - \delta_r)$. An attacker might also decide to run an attack on a user specified part of the trace set, instead of attacking all available traces.

Besides selecting an appropriate interval to attack, an attacker needs to specify a power model according to which the predictions are made. The power model translates predictions of actual intermediate value to power consumption estimations.

Finally, an attacker has to specify the type of mutual information estimation model: via histograms or cumulants. Depending on this decision, further input might be required. This is explained in the corresponding section.

A power trace arising from processing plaintext d is again denoted by an array $(o_{d,1}, \ldots, o_{d,l})$. Predictions can be seen as an array of n values $(h_{d,1}, \ldots, h_{d,n})$, for each plaintext d.

General Input

Summarizing the previous paragraph, the following steps must be taken before starting a general second order mutual information analysis.

- Select an intermediate value and a power model.
- Select part of the power trace where information leakage is expected.
- Enter the mask distance and the mask range.
- Enter the number of traces on which analysis is performed.
- Select a mutual information estimation model.
- Select whether or not to normalize mutual information values. However, the use of normalized mutual information as distinguisher is discouraged, see Section 6.3.3.

In the following sections, method-specific information is given for histogram-based MIA and cumulant-based MIA respectively.

6.2.3 Histogram Method

The following requirements apply to second order mutual information analysis using histograms.

Parameters

Once again, an attacker has to decide on the number of bins v for the observation. This can again be done by Sturges' rule, Scott's rule or by a fixed decision, as explained in Section 6.1.3. Following the same arguments, a fixed decision is made prior to the analysis. A user can henceforth specify this number of bins for observation. The decision to have the same number of bins for each t still holds. When considering two time instances, both probability density functions are estimated using the same number of bins. The number of prediction bins follows from the selected prediction method and is equal to the number of possible outcomes.

An attacker can select whether or not to save the histograms computed during the analysis. This reduces the computation time of a second attack on the same histogram, since the histogram only has to be read from disk, instead of constructed from the traces. Stored histograms are especially useful when evaluating different scoring models (that is, higher order mutual information generalizations) on the same set of traces.

Consequently, when a histogram has been stored during a previous attack, an attacker may also select to analyze this histogram, instead of a trace set. In this case, only the desired scoring model has to be selected. All other parameters are fixed by the stored histogram. Evidently, no new histograms can be saved to disk, when analyzing saved ones.

Method-specific input

- An attacker must select the number of bins for observation. The number of bins for prediction is determined by the intermediate value and the power model; an attacker is not able to change this decision.
- Four mutual information generalizations are implemented as second order distinguisher. The attacker must select one.
- The attacker can specify whether or not to store histograms.
- When histograms from a previous analysis have been stored, an attacker can choose to analyze these histograms again, instead of analyzing the current trace set. Whether histograms have been stored previously is not checked by the module; a user is responsible for this knowledge.

Computation

First, analysis of traces and construction of histograms is explained. Thereafter, analysis of previously stored histograms is treated. The following description of computation steps is strongly related to the description of first order histogram computation in Section 6.1.3.

Histogram construction. Estimation of mutual information using histograms involves the following computational steps. First, all traces are analyzed to determine upper bounds and lower bounds on observations (Pass 1). Then, the bounds for each histogram bin are determined, as well as the number of phases in which three-dimensional histograms are constructed. Finally, the actual construction of the histograms takes place.

Pass 1
For each t, the lower bounds $(\alpha_1, \ldots, \alpha_l)$ and the upper bounds $\omega_1, \ldots, \omega_l$ are determined.

Intermediate step 1
Now the upper bounds and lower bounds are determined, the bin bounds are computed. The module assumes equally sized bins, hence the bounds $b_t(i)$ are calculated using

$$b_t(i) = \alpha_t + i\frac{\omega_t - \alpha_t}{v}, i = 0, \ldots, v.$$

The closed observation interval $[a_t, \omega_t]$ is divided into bins according to:

$$[b_t(0), b_t(1)),\ [b_t(1), b_t(2)),\ \ldots,\ [b_t(v-1), b_t(v)].$$

The number of bins u arises from the prediction model and selected intermediate; this is equal to the number of possible outcomes. Since prediction values are integer, the prediction bins are of the form

$$\{0\}, \{1\}, \ldots, \{u-1\}.$$

Intermediate step 2
To avoid memory overflows, analysis is performed in phases. The number of phases w is determined by size of the intervals in which the first and second observation are expected: respectively $(l - \delta_m - \delta_r)$ and δ_r. If the second range is too large (w.r.t. a chosen parameter R) to be analyzed in one pass, this range is split. If it is small enough, multiple time points from the first range are combined.

- If threshold $U = \lceil \delta_r / R \rceil > 1$ then $w = \lceil \delta_r / R \rceil \cdot (l - \delta_m - \delta_r)$.
- Else, $w = \lceil (l - \delta_m - \delta_r) / \lfloor R/\delta_r \rfloor \rceil$.

In the current implementation, $R = 100$. This value balances the number of passes and memory usage. However, no wide investigation is performed into optimizing this number, neither in optimizing the way to decide on the number of phases.

If $U > 1$, then phase $s (s = 2, \ldots, w+1)$ consists of mutual information estimation for time point $\lceil (s-1)/U \rceil$, paired with the first $(s \mod U) \cdot \delta_r$ time points from the second range. If $U = 1$, then time points $(s-2)\lfloor R/\delta_r \rfloor + 1$ to $(s-1)\lfloor R/\delta_r \rfloor$ are paired with all δ_r time points from the second range.

Pass 2 to w+1
For each time combination in a phase and each of the n key guesses, a histogram of size $u \times v \times v$ is created. This (yet empty) histogram corresponds to a three dimensional frequency matrix F. Now for every trace, increase frequency matrix entry $F_{\hat{k},t_1,1_2}(i,j,k)$ by one if a prediction under key guess $\hat{k}$ goes to prediction bin i, observation at time t_1 to observation bin j and observation at time t_2 to observation bin k.

If the attacker has selected the option to save histograms to disk, the histograms from a phase are written to disk at this moment. When the last phase has ended, a separate file is written to disk, containing information about the saved histograms.

When all traces are processed, the matrix $F_{\hat{k},t_1,t_2}$ of each triple $(\hat{k}, t_1, t_2)$ is a three-dimensional frequency table. For each triple $(\hat{k}, t_1, t_2)$, the values of the empirical joint probability function are stored in the matrix $J_{\hat{k},t_1,t_2}$; they are obtained by dividing each entry of $F_{\hat{k},t_1,t_2}$ by the number of traces q.

The marginal and joint probability density functions of prediction and observations required by the selected distinguisher are calculated from the joint density by adding up corresponding sections. From these distributions, entropy and subsequently mutual information is estimated as described in Section 5.5.1. These values are stored in the mutual information matrix M, an array of size $n \times (l - \delta_m - \delta_r) \times \delta_n$.

Stored histograms. In case an attacker has chosen to analyze previously stored histograms, a slightly different analysis is performed. First, the module reads information about the parameters of the stored histograms from a specific file. This provides the attacker with information about the number of bins, the number of traces analyzed and the specific ranges in time used for analysis. An attacker only has to specify the desired distinguisher, i.e., mutual information generalization.

Next, the histograms are read piece by piece. After reading a histogram for a time pair - key guess combination, the corresponding mutual information value is estimated using the selected distinguisher and stored in the mutual information matrix M, from which later a candidate ranking is constructed.

Specific output

The attacker can choose to store all the histograms to disk. This allows on to further analyze the histograms on specific properties, like shape. Moreover, each higher order generalization of mutual information is estimated from the same histogram. Hence, storing a histogram avoids reconstructing the same histogram for a different distinguisher. In order to perform scoring by multiple models, the histogram has to be constructed only once. This strongly reduces computation time.

Using multiple distinguishers increases success probability of attacks. For example, subkey candidates that are top-ranked by multiple distinguishers are more likely to be correct than candidates that are only top-ranked by one distinguisher.

6.2.4 Cumulant Method

This section summarizes the computations that need to be performed in order to perform a mutual information analysis using the cumulant computation method. The reader is advised to read Section 6.1.4 in advance, since implementation of second order cumulant MIA strongly resembles implementation of first order cumulant MIA.

Parameters and Method-specific Input

As explained in Section 5.5.2, the cumulant method is a parametric estimation model where near-Gaussian distributions of prediction and observation are assumed. The parameters for these near-Gaussian distributions must be estimated from the traces. No specific further input is required for the cumulant method.

Computation

The cumulant method works with normalized values, so a normalization is included. Therefore, a two-pass algorithm is used, where the mean and standard deviation are computed during the first pass, and the calculation with normalized values can take place during the second pass.

First pass

During the first pass, the sample means $m_{\mathbf{H}_{\hat{k}}}$ and sample standard deviations $s_{\mathbf{H}_{\hat{k}}}$ of the predictions $h_{\hat{k}}$ are calculated for each $\hat{k}$. Besides that, the sample means $m_{\mathbf{O}_t}$ and sample standard deviations $s_{\mathbf{O}_t}$, as well as the third and fourth raw moment of the observations o_t are calculated. The third and fourth moment can be useful to test for normality. In order to achieve numerical stability, the moments are again determined incrementally according to Pébay [Péb08].

Second pass

During the second pass, observations are treated in their normalized form, i.e.

$$\tilde{h}_{d,\hat{k}} = \frac{h_{d,\hat{k}} - m_{\mathbf{H}_{\hat{k}}}}{s_{\mathbf{H}_{\hat{k}}}} \text{ and } \tilde{o}_{d,t} = \frac{o_{d,t} - m_{\mathbf{O}_t}}{s_{\mathbf{O}_t}}.$$

The cumulant method assumes samples to have a probability distribution close to the standard normal distribution. Hence, two routines are executed during the second pass: feasibility testing and incremental moment updating.

- Testing for feasibility
  In the implementation, the points in time t where $s_{\mathbf{O}_t} = 0$ are neglected, since no information can be obtained from constant data. As argued in Section 6.1.4, no test for normality is included. This might, however, be injected at this point in the analysis.
- Update incremental moments
  For each $\hat{k}$ and pair of time points t_1 and t_2, several (mixed) sample moments of $\tilde{\mathbf{H}}_{\hat{k}}$, $\tilde{\mathbf{O}}_{t_1}$ and $\tilde{\mathbf{O}}_{t_2}$ are computed incrementally using the triples $(\tilde{h}_{d,\hat{k}}, \tilde{o}_{d,t_1}, \tilde{o}_{d,t_2})$, $d = 1, \ldots, q$. The moments required for estimation of mutual information are derived from Equation 5.19:

$$\begin{array}{lllll}
E[\tilde{\mathbf{H}}^2_{\hat{K}}] & E[\tilde{\mathbf{H}}_{\hat{K}}\tilde{\mathbf{O}}_t] & E[\tilde{\mathbf{H}}^2_{\hat{K}}\tilde{\mathbf{O}}^2_t] & E[\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}_{t_2}] & E[\tilde{\mathbf{O}}^3_{t_1}\tilde{\mathbf{O}}_{t_2}] \\
E[\tilde{\mathbf{O}}^2_t] & E[\tilde{\mathbf{H}}^2_{\hat{K}}\tilde{\mathbf{O}}_t] & E[\tilde{\mathbf{H}}^3_{\hat{K}}\tilde{\mathbf{O}}_t] & E[\tilde{\mathbf{O}}^2_{t_1}\tilde{\mathbf{O}}_{t_2}] & E[\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}^3_{t_2}] \\
E[\tilde{\mathbf{O}}^2_t] & E[\tilde{\mathbf{H}}_{\hat{K}}\tilde{\mathbf{O}}^2_t] & E[\tilde{\mathbf{H}}_{\hat{K}}\tilde{\mathbf{O}}^3_t] & E[\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}^2_{t_2}] & E[\tilde{\mathbf{O}}^2_{t_1}\tilde{\mathbf{O}}^2_{t_2}]
\end{array}$$

$$E[\tilde{\mathbf{H}}_{\hat{K}}\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}_{t_2}] \qquad E[\tilde{\mathbf{H}}^2_{\hat{K}}\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}_{t_2}] \qquad E[\tilde{\mathbf{H}}_{\hat{K}}\tilde{\mathbf{O}}^2_{t_1}\tilde{\mathbf{O}}_{t_2}] \qquad E[\tilde{\mathbf{H}}_{\hat{K}}\tilde{\mathbf{O}}_{t_1}\tilde{\mathbf{O}}^2_{t_2}]$$

  An index t without subindex means that this moment has to be estimated for both $t = t_1$ and $t = t_2$.
  Since the means of $\tilde{\mathbf{H}}_{\hat{k}}$, $\tilde{\mathbf{O}}_{t_1}$ and $\tilde{\mathbf{O}}_{t_2}$ are equal to zero, the incremental update of the a, b, c'th mixed first moment $\mu_{a,b,c}$ of $d-1$ samples to $\mu'_{a,b,c}$, where the triple $(h_{d,\hat{k}}, o_{d,t_1}, o_{d,t_2})$ is added to the set, is performed as follows:

$$\mu'_{a,b,c} = \mu_{a,b,c} + \frac{(\tilde{h}_{d,\hat{k}})^a(\tilde{o}_{d,t_1})^b(\tilde{o}_{d,t_2})^c - \mu_{a,b,c}}{d}.$$

  The value of $\mu_{a,b,c}$ is an estimation of $E[\tilde{\mathbf{H}}^a_{\hat{k}}\tilde{\mathbf{O}}^b_{t_1}\tilde{\mathbf{O}}^c_{t_2}]$. These mixed first moments are estimated for each $\hat{k}$ and time pair t_1, t_2.

Score

For each $\hat{k}$, t_1 and t_2, a mutual information value is estimated in the following way, and stored in the mutual information matrix M.

First, the auxiliary intermediates R, T, Q_1, Q_2 and Q_3, based on Equations 5.20 to 5.25, are computed.

$$\begin{aligned}
R =& \mu_{1,1,0}^2 + \mu_{1,0,1}^2 + \mu_{0,1,1}^2; \\
T =& \mu_{2,1,0}^2 + \mu_{2,0,1}^2 + \mu_{1,2,0}^2 + \mu_{1,0,2}^2 + \mu_{0,2,1}^2 + \mu_{0,1,2}^2 + \mu_{1,1,1}^2; \\
Q_1 =& (-3\mu_{2,0,0}\mu_{1,1,0} + \mu_{3,1,0})^2 + (-3\mu_{1,1,0}\mu_{0,2,0} + \mu_{1,3,0})^2 + (-3\mu_{2,0,0}\mu_{1,0,1} + \mu_{3,0,1})^2 + \\
& (-3\mu_{0,2,0}\mu_{0,1,1} + \mu_{0,3,1})^2 + (-3\mu_{1,0,1}\mu_{0,0,2} + \mu_{1,0,3})^2 + (-3\mu_{0,1,1}\mu_{0,0,2} + \mu_{0,1,3})^2; \\
Q_2 =& (-2(\mu_{1,1,0})^2 - \mu_{2,0,0}\mu_{0,2,0} + \mu_{2,2,0})^2 + (-2(\mu_{1,0,1})^2 - \mu_{2,0,0}\mu_{0,0,2} + \mu_{2,0,2})^2 + \\
& (-2(\mu_{0,1,1})^2 - \mu_{0,2,0}\mu_{0,0,2} + \mu_{0,2,2})^2; \\
Q_3 =& (-2\mu_{1,1,0}\mu_{1,0,1} - \mu_{2,0,0}\mu_{0,1,1} + \mu_{2,1,1})^2 + (-\mu_{0,2,0}\mu_{1,0,1} - 2\mu_{1,1,0}\mu_{0,1,1} + \mu_{1,2,1})^2 + \\
& (-2\mu_{1,0,1}\mu_{0,1,1} - \mu_{1,1,0}\mu_{0,0,2} + \mu_{1,1,2})^2.
\end{aligned}$$

Then the approximation of the mutual information value is computed as:

$$I(\tilde{\mathbf{H}}_{\hat{k}}, \tilde{\mathbf{O}}_{t_1}, \tilde{\mathbf{O}}_{t_2}) = c\left(\frac{R}{2} + \frac{T}{4} + \frac{Q_1}{12} + \frac{Q_2}{8} + \frac{Q_3}{4}\right).$$

The constant c equals $\frac{1}{\log 2}$, to convert the logarithms from base e to base 2.

6.2.5 General Output

The mutual information matrix M is constructed during the analysis, a subkey candidate ranking is made based on this matrix. Along with the ranking, information about the two time points where high mutual information is observed is presented. More specific, the offset of the first observation is given, together with the distance to the second observation. The most likely cipher key guess is then obtained by reverse-engineering the cipher key from the subkeys in the first round.

6.3 Considerations on Design

Designing the MIA Module involves making design decisions on many moments. Most of the design decisions have already been explained in this chapter. However, small consideration on the design of the module are presented here. First, multiple suggestions to reduce memory usage are presented, one of which is actually implemented. Furthermore, a brief explanation of unit testing is provided. Finally, considerations on the usage of normalized mutual information are shared.

6.3.1 Memory Usage Reduction

Histograms in a second order analysis involve saving a total of (`number of key guesses * number of time pairs * number of prediction bins * number of observation bins * number of observation bins`) values. In case of a masked DES attack with resolution 16 and identity model on one sample pair, this is $512 \times 1 \times 16 \times 16 \times 16$ values for only one sample pair, which is already 8 megabytes. In case of e.g. 100 sample pairs, one needs 100 times this amount of memory. Consider the following options to reduce this amount.

- Write (parts) to memory. That means, transform from working memory (ram) to disk memory. However, no easy way of random reading from disk memory exists within Inspector. Hence, this method yields no benefit, since a complete file must be read and converted to e.g. a float array, before the desired value can be processed. No gains in memory usage is achieved, and this method would imply a significant loss in time consumption.
- Divide attack per sample. That means, do the attack per sample or per sample pair (or a small number of pairs) and only store the score. Then, reread the traces and attack next pair(s). This reduces memory usage, but increases time consumption, since traces pass multiple times. However, when memory usage exceeds available memory, this might be a

good trace off. Usage of memory is compensated by increasing the amount of runs. Actually, this method is implemented in the MIA Module.

- Divide the attack per key. That means, read all samples, but only predict for one subkey and store the result. Again, traces must be read multiple times, which is slower but reduces memory usage.
- The attack subdivisions can be combined, to that for each pass only the MI of a small number of samples (or pairs) and a small number of key guesses is computed and stored. However, multiple passes of the traces must occur, again transforming use of memory to use of time. It is not very likely that this solution needs to be used in practice, since generally one of the two solutions above (split either observations or predictions) suffices.

6.3.2 Unit Testing for Method Validation

One of the features of Java is the possibility to test parts of the code, so-called unit testing. From independent parts of the code, so-called units, such as calculation methods or comparison methods, should be tested whether they do what they are expected to do. A small error in calculation can easily be overlooked when it induces no major consequences in the program's output. However, the calculation is wrong; this must be avoided. The implementation of mutual information analysis involves numerous computations of, e.g., logarithms or entropies. These methods must be numerically stable and consistent. To provide a code reviewer or software engineer with evidence that the computations are performed correctly, each unit is tested.

All computational methods are tested, sometimes in multiple ways. The test cases deal with realistic values, and sometimes with unrealistic values, in order to test (asymptotic) behavior of the method. This proves and guarantees that the computations in the MIA Module are correctly implemented.

6.3.3 Normalized Mutual Information

To transform mutual information into a score between zero and one, Definition 2.2.8 presents the normalized mutual information:

$$I_N(\mathbf{X};\mathbf{Y}) = \frac{I(\mathbf{X};\mathbf{Y})}{\min\{H(\mathbf{X}), H(\mathbf{Y})\}}$$

This expression can easily be generated to more than two dimensions. However, the use of normalized mutual information as distinguisher is discouraged for the following two reasons.

- If observations at a certain point t are nearly constant, mutual information of predictions and observation at time t are approximately equal to the entropy of the observations. The normalized mutual information would be approximately equal to one in this case. However, near-constant observations give very few information about correctness of key guesses. This henceforth results in ghost peaks.
- In high security implementations, information leakage via power consumption might be marginal. Hence, mutual information values can be rather small. On the one hand, if also the minimum of prediction entropy and observation entropy is small, dividing two small numbers might show unexpected behavior. On the other hand, if the minimal entropy is relatively large, the division might yield a number close to the numerical precision of the virtual machine.

Both situations occur regularly when analyzing trace sets. On trace sets analyzed in the next chapter, normalized mutual information never performed more efficient than regular mutual information. Therefore, no results on normalized MI are presented and the use of normalized mutual information as side channel distinguisher is dissuaded.

Chapter 7

Results

In this chapter, test results of the Mutual Information Analysis Module are given. Trace sets from real embeddings of both AES and DES, as well as simulated trace sets, are tested. The Mutual Information Analysis Module is compared with Riscure's conventional first order and second order analysis methods. For simulation, one of Riscure's simulation modules is used. Simulated traces are based on uniformly random plaintexts and a given cipher key. The practical implementations are referred to as *Riscure Sample Type Card i*, where i denotes the card identifier.

The following table of contents summarizes the attacks performed using the Mutual Information Analysis module.

Contents

First order analysis is performed on three practical DES embeddings and an AES embeding, each one leaking information about a different intermediate value and all implemented on an 8-bit CPU. This means that data is processed in blocks of eight bits. Some embeddings show more nonlinear leakage than others; this behavior is not caused by attacking a different intermediate value, it rather depends on implementation properties of the embedding. Therefore, including results on all these embeddings can increase insight in the appropriateness of MIA on different DES implementations.

For DES, both Hamming weight-based prediction models and identity-based prediction models are tested. For AES, only Hamming weight prediction models are used, to reduce computational complexity. Results on both practical implementations and simulations are presented. Simulations are used to thoroughly investigate behavior of MIA under different settings, such as noise-resistance or success rate of an attack. Analyzing many traces reduces statistical noise (cf. the central limit theorem) and therefore improves results. However, more time and memory might be required.

Simulations often consider the *success rate* of a certain event. This is defined as the number of successes among a trial, divided by the number of runs in the trial. For example, one could run attacks on 100 trace sets of equal size and count the number of successful runs. Alternatively, one can do one run on, e.g., 16 AES S-boxes and count the number of correctly compromised subkeys. For this approach, one must assume that S-boxes are identical and every S-box output leaks in the same way. Evidently, in AES simulations, these assumptions are fulfilled. In DES, eight different S-boxes exist, so one of the two assumptions does not hold. Hence, for DES the success rate will be defined as the number of correctly guessed S-box subkeys, divided by eight.

The notion of success rate can be extended to the o-th order. This means that one judges an attack to be successful in case the correct subkey belongs to the top-o ranked candidates. When an attacker performs a brute force recovery among the top-o candidates after the candidate ranking, the o-th order success rate might especially be meaningful. In this case, an attack on a round key is performed by first ranking the top-o candidates for each one of the r subkeys, an then execute a brute force search among the o^r possibilities. In this thesis, however, only order-one is examined.

First order analysis results on the success rate for different numbers of traces and different noise levels are simulated, as well as the influence of the number of bins for observation in a histogram (see Section 4.4.1). Recall from Section 6.1 that for a histogram attack, the number of bins for prediction chosen to be equal to the number of distinct possible outcomes. On real embeddings, the number of traces required to compromise the cipher key is computed for multiple choices of bins for observation in a histogram. Moreover, cumulant MIA (see Section 4.4.2) is evaluated. For comparison, correlation analysis (see Section 3.3) is performed. The number of traces required to correctly compromise a key from a real embedding and the success rate of a simulation are two related measures for success, however they are not equivalent. Interpreting these two measures requires careful comparisons.

Second order analysis also focuses on different higher order generalizations of mutual information, as introduced in Section 5.4.2. Second order mutual information attacks are compared with second order correlation attack, in particular with the absolute difference combining correlation attack (see Section 3.3.2).

In some cases, results are accompanied with visualizations of histograms. These histograms represent the joint probability distributions of prediction and observation for different key guesses. They might be used to verify results on, e.g., linearity in leakage.

The vertical axis in this visualizations represents observation density. Prediction density is represented on the horizontal axis. When the histogram of the second best key guess is depicted, its score is given as ratio to the first score. For example, if the correct key guess is ranked first, based on a mutual information value of 1.2, and the second best guess descends from a mutual information value if 0.6, then the score ratio equals 50%.

Unless stated otherwise, charts are colored as depicted in Figure 7.1.

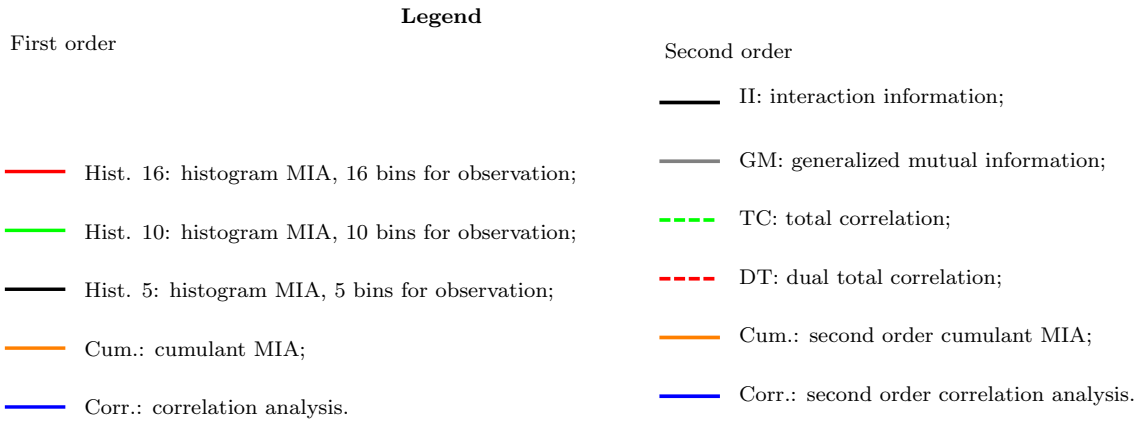


Figure 7.1: Legend of coloring convention in this chapter.

7.1 AES S-box Output

This attack aims at AES S-box output bytes in the first round. Every S-box manipulates one input byte, which in turn equals the sum of one plaintext byte and cipherkey byte. Hence, an S-box attack focuses at key bytes individually.

An AES S-box outputs one byte, which is 8 bits (see Appendix A.2). Hence, the number of distinct possible outcomes equals $2^8 = 256$. An identity model using a bin for every single possibility would henceforth use 256 bins for prediction, which might have negative influence on the performance. Not only can it be time-consuming, but many samples might be required in order to obtain a sufficient approximation of the underlying joint distribution of prediction and observation. Consequently, only Hamming weight-based attacks are performed on AES. The number of bins for prediction is then equal to 9.

Parameters of this attack:

Intermediate value: $S(x)$ where $S(\cdot)$ represents the AES S-box function and x represents a byte of the AES state after the first round key guess $\hat{k}$ (which is equal to the cipher key guess) is added. The attack is performed on each of the 16 bytes in the state.

Prediction: $\mathbf{H}_{\hat{k}} = HW(S(x))$.

Observation: $\mathbf{O}_t$ is the power consumption partly based on the AES S-box output at time t.

7.1.1 Simulation

In the following sections, the success rate for different numbers of traces and different noise levels are estimated by simulations. Moreover, the influence of the number of bins for observation on the performance is examined.

Success Rate

The success rate is determined on a noisy signal. Power consumption is modeled as having a linear relation with Hamming weight of intermediate values, which suits a Hamming weight attack.

The signal-to-noise ratio is defined as

$$SNR = \frac{\sigma_T^2}{\sigma_N^2},$$

where σ_T^2 and σ_N^2 represent variance of the traces, resp. noise. A more detailed description of SNR can be found in, e.g., Mangard et al [MOP07], Section 4.3. For an AES S-box output of 8 bits leaking Hamming weight, the variance equals $8/4 = 2$, as explained in Section 4.4.2.

To establish a success rate on many traces, a large noise component can be added to simulated traces. In this attack, Gaussian noise with mean 0 and standard deviation 40 is added to every sample, yielding a SNR of 1/800. This large standard deviation is chosen to get a grip on the behavior of mutual information analysis in noisy environments. In later sections, traces with less noise are examined as well.

Figure 7.2 depicts the success rate for different numbers of traces. Mutual information using histograms and cumulants for density estimation are used, as well as correlation analysis for comparison.

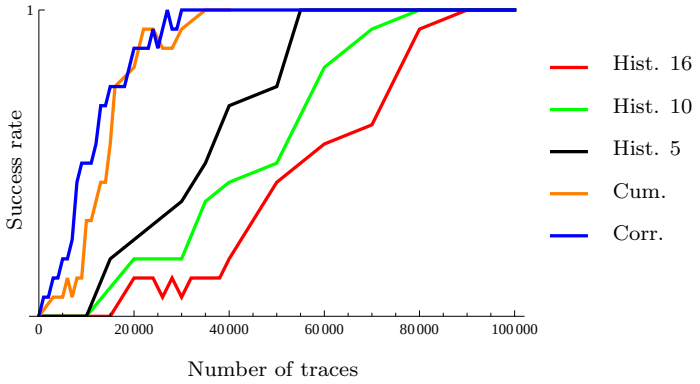


Figure 7.2: Success rate of a first order *Hamming weight* attack on an AES simulation on traces with signal to noise ratio of 1/800.

A first observation is that correlation analysis is most efficient in general; presumably due to the linear relation between leakage and Hamming weight. In comparison with CPA, more traces are required in a histogram-based mutual information attack, to adequately model density functions. In contrast, a correlation analysis can quickly gather enough evidence that supports the underlying linear relation.

Second, cumulant-based MIA performs significantly better than any of the histogram-based attacks. Cumulant-based MIA approaches the performance of correlation analysis closely. This probably is be due to the fact that the normality assumption of both prediction and observation is fulfilled. Gaussian noise does not seem to have a bad effect on cumulant MIA.

Third, more histogram bins require more traces to achieve the same success rate. Apparently, distinguishing advantages of more bins do not compensate the need for a balanced density estimation. Moreover, more bins generally are more sensitive to noise, since noise can distort an observation to several bins away. This relation is further investigated in the next section.

One of the proposals for the number of bins for observation introduced in Section 6.1.3 is Sturges' rule: $v_t = 1 + \log q$. In case of $q = 40000$ traces, Sturges' rule indicates that 16 bins should be used. However, other choices for this number already yield better results.

Figure 7.3 depicts visualizations of two histograms from an attack on an AES S-box output, leaking Hamming weight. The left image belongs to the joint density of prediction and observation in case of a correct subkey guess, the right one belongs to the joint density for the second best subkey guess. These histograms are simulated by means of 100 noiseless traces. The second best guess scores only 22% compared with the correct guess. Due to the lack of noise, this is an ideal case; one that never occurs in practice.

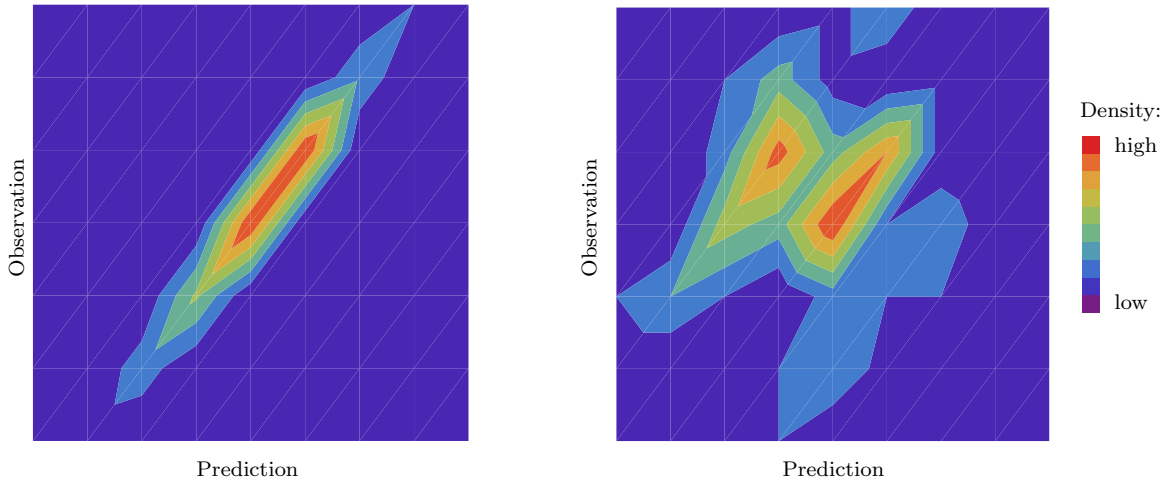


Figure 7.3: Histogram for correct key guess (left) and second best key guess (right) from an AES S-box output attack on 100 simulated noiseless traces.

Number of Bins

To investigate the influence of the number of bins in a histogram attack, another analysis is performed. Two trace sets, with noise standard deviation $\sigma_N = 10$, resp. $\sigma_N = 5$, are examined to determine the influence of the number of bins for observation on the number of traces required for a successful AES cipher key recovery on different noise levels. Figure 7.4 depicts the results.

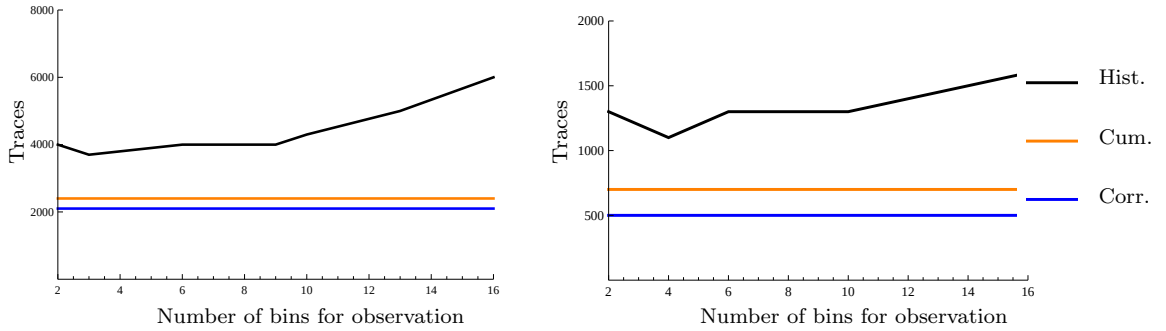


Figure 7.4: Influence of number of bins for observation on number of traces needed for a successful first order attack on the first key byte in an AES simulation. Left: $\sigma_N = 10$, right: $\sigma_N = 5$. Since both cumulant MIA and correlation analysis are independent from any bin decision, these lines are straight.

From Figure 7.4, one can see an optimum in the number of bins around 3-4. Apparently, too few bins limits distinguishing power, and too many bins requires more traces to obtain a good approximation of the underlying joint distribution. However, to quantify *too few* and *too many*: two bins and six bins score equally, the optimum is in between.

Another interesting property can be distilled from this figure. Whereas nine bins and fewer score more or less equally, a linearly increasing trend can be observed when the number of bins is greater than nine. Nine is exactly the number of bins in a histogram from a AES S-box output Hamming weight prediction model. It seems that when the histogram contains more bins for observation than for prediction, performance degrades linearly with the number of bins for observation. The moment where this trend starts is referred to as the *bin degradation point*. In Figure 7.4 (left), the bin degradation point is at nine, in Figure 7.4 (right), the bin degradation point is at ten.

The performance might as well depend on the number of traces. In case of much noise, many traces are required to compromise the key. Many bins constitute a high resolution and hence better distinction ability. Moreover, a smooth density is obtained from the large number of measurements. However, more bins can be more vulnerable to noise, since a noise observation can be placed several bins away from its noiseless destination. When, in contrast, little noise is present on a signal, using many bins may be not very adequate. Still a significantly large number of measurements is needed to obtain an appropriate estimation of the density. Using fewer bins might require fewer measurements, yielding a rough estimation of the density, instead of a scattered one.

Comparing Figure 7.4 (left) and Figure 7.4 (right), one can verify that halving the standard deviation of noise reduces the number of traces with about a factor of four. This supports a property, described by Mangard et al. [MOP07], that there exists a linear relation between noise variance and number of traces required to compromise keys.

Resistance to Noise

Noise is unavoidable in real implementations, for example due to physical properties of registers or implementations, as well as parallel execution. The extent to which an attack is resistant to noise, is an important aspect of the feasibility of the attack. Therefore, the behavior of mutual information analysis in a noisy environment is assessed.

Success rates are determined on multiple trace sets, where each trace set is generated using a different noise standard deviation. An inverse relation appears: the more noise on a signal, the less successful an attack is. In this section, noise resistance is determined on a set of 50000 traces.

Noise is modeled as being Gaussian, with zero mean and standard deviation σ. After leakage of a simulated intermediate value is modeled by means of a leakage model, noise is added to the signal. Figure 7.5 depicts the noise resistance of different mutual information analysis, compared with noise resistance of correlation analysis.

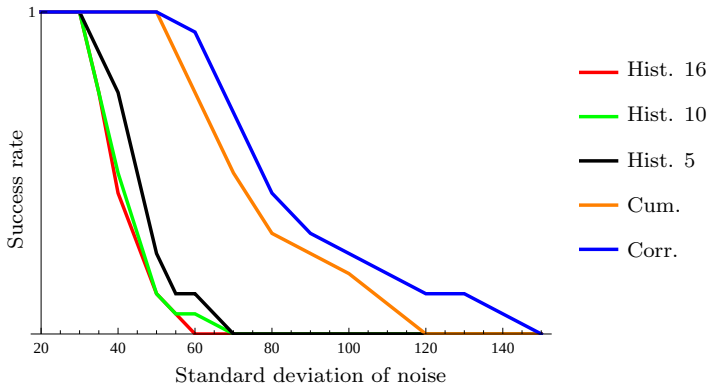


Figure 7.5: Noise resistance of a first order *Hamming weight* attack on an AES simulation.

As can be seen in Figure 7.5, correlation analysis is more resistant to noise than any mutual information analysis type. Moreover, cumulant-based mutual information analysis is far more resistant to noise than any of the histogram-based variants. This figure matches the order of performance depicted in figure 7.2.

Cumulant MIA strongly benefits from the Gaussian assumption, which holds for both leakage and noise. Hence, it can compete with correlation analysis much better than any histogram-based MIA. A preference for cumulant MIA over histogram MIA rises from this figure, as well as from previous analysis. However, correlation analysis still performs better. This might be due to the fact that the leakage is modeled perfectly linear. Correlation analysis is tailored to these types of dependence.

The figure suggests that more bins in a histogram attack are more vulnerable to noise. However, increasing the number of bins does not give a large difference in noise resistance, since the lines for

10 and 16 bins nearly overlap.

7.1.2 Low Security Software Embedding

A low security software embedding of AES is available from Riscure's Sample Type Card 3. In a software implementation, like this is, one encryption typically takes more time than in a hardware implementation, since generally no parallel processes are performed. Consequently, one trace consists of many samples and a mutual information attack might be heavily time-consuming. In this card, `SubBytes()` is implemented as 16 separate S-box table lookups. This means that the Hamming weight of each S-box output byte is strongly correlated with power consumption of the card. A Hamming weight-based prediction model is used, about 3700 consecutive time points are analyzed.

Figure 7.6 indicates the number of traces required to break this card. MIA using both histograms and cumulants are tested, as well as correlation analysis for comparison. For histogram MIA, attacks are performed for $2, \ldots, 16$ bins for observation. Recall that the number of bins for prediction is equal to the number of possible prediction outcomes, being 9.

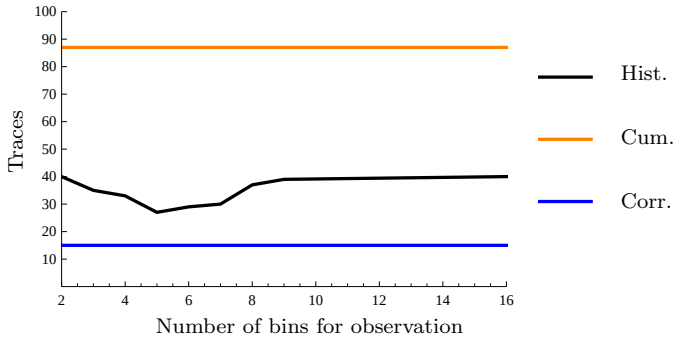


Figure 7.6: Number of traces needed for a complete successful AES *Hamming weight* attack on Sample Type Card 3, specified for different numbers of bins for observation.

As can be seen in Figure 7.6, correlation analysis is very efficient on this trace set: this method requires the fewest number of traces to correctly compromise the cipher key. Due to the low security applied on this card, leakage has a strong linear relation with Hamming weight. A correlation attack only needs 15 traces to establish a good model. Mutual information-based attacks, however, aim at estimating probability densities. It is no surprise that a mutual information attack requires more samples to adequately estimate underlying densities.

Histogram MIA scores worse than correlation analysis; however is still within a factor of four in number of traces. Comparing Figure 7.6 to Figure 7.4, no large deviations from simulation results appear. Like in simulations, using more than 9 bins for observation does not improve the result. This is the point where histograms contain more bins for observation than for prediction. The most efficient analysis uses histograms with 5 bins for observation.

Cumulant MIA scores much worse than, e.g., in simulation results. This can be due to two aspects. First, since this low security embedding has high information leakage, other methods can adequately model different probability density functions early. Cumulants, however, can require more traces to adequately estimate differences in density parameters for every subkey. Recall that the cumulant method assumes an underlying Gaussian distribution. More measurements are required to adequately approach a Gaussian density, compared to a nonparametric density estimation method on this few traces.

Second, although little noise is present on measurements from this card, this noise may not be Gaussian. This can influence the performance of cumulant MIA, since the Gaussian assumption might not be fulfilled within a small number of traces.

Correlation analysis on only 20 traces already yields correlation coefficients of around 0.98 for the correct subkeys. Clearly, the leakage has a nearly linear relation with the Hamming weight. A visualization of the histogram constructed for the correct first key byte, based on 100 measurements, is depicted in Figure 7.7 (left). This histogram clearly supports the linear relationship between prediction and observation. A histogram of the second-best subkey guess, which only scores 39% of the fist, can be seen in Figure 7.7 (right).

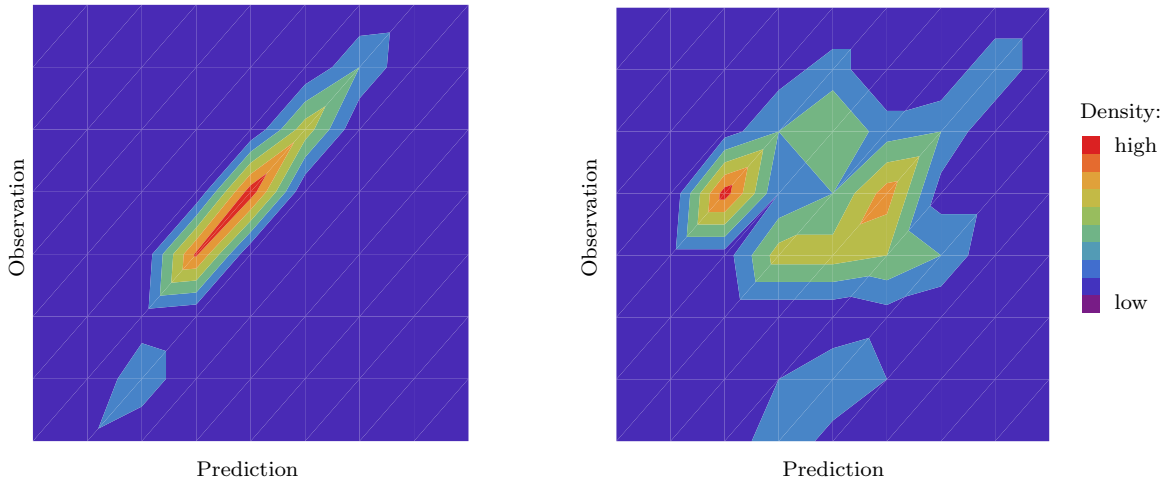


Figure 7.7: Histogram for correct key guess (left) and second best (39%) key guess (right) from the AES S-box output *Hamming weight* attack on Sample Type Card 3.

These histogram visualizations can easily be compared with the visualizations of histograms from noiseless simulations in Figure 7.3. As can be seen, the correct key yields an almost perfectly linear relation between prediction and observation, whereas the second best key guess already yields a more scattered pattern in the histogram. The discrepancy in Figure 7.7 (left) can be due to the absence of observations in a certain range.

7.2 DES S-box Output

In this section, an attack targeting the first round S-box output in a DES encryption (see Appendix A.1) is performed. This attack ultimately yields 48 bits of the DES key. The remaining 8 bits are usually compromised by brute-force attacks using plaintext-ciphertext pairs. Since the eight S-box functions in a DES round are non-identical, results are presented for each individual S-box.

A DES S-box input is a function of six (expanded) plaintext bits and six round key bits. The output, however, is four bits long. Hence, 16 different output states can occur; an identity attack is feasible on this number of possibilities. Besides an identity attack with 16 bins for prediction, a Hamming weight attack with 5 bins for prediction is performed.

Parameters of this attack:

Intermediate value: S-box output $S_i(x)$ for $i = 1, \ldots, 8$. S-box input x is the first round input, which depends on the known plaintext and a (guessed) key.

Prediction: $\mathbf{H}_{\hat{k}} = HW(S_i(x))$ and $ID(S_i(x))$.

Observation: $\mathbf{O}_t$ is the power consumption partly based on the S-box output at time t.

Theoretically, a DES round function only contains one nonlinear step: the S-boxes. Every other step in the round function, as well as in the DES scheme, is deterministic and easily invertible. Hence, predicting the left half of the round output (which equals the right half of the next round input) is as hard as predicting the round function output, since the only step in between consists of a XOR with the left half of the round input. In turn, predicting the round function output reduces to predicting the S-box output, since the only step in between is a 32-bit permutation (which is bijective). Therefore, predicting the round output is theoretically equivalent with prediction S-box output.

Prediction the difference between a round input and a round output reduces to prediction S-box output from input. When the round output is determined as written above, a XOR with the input yields the desired value. Therefore, prediction round input-output difference is theoretically equivalent with predicting S-box output.

The reason to examine attacks on these intermediate values separately is that different practical implementations might leak different intermediate values. However, in terms of simulations, these attacks can be regarded as being equivalent. Therefore, only simulation results on one type of first order DES leakage is presented, namely DES round output in Section 7.3.1. No simulation results on S-box output are presented in this section, nor are simulation results on round difference presented in Section 7.4.

7.2.1 Low Security Software Embedding

An implementation of DES, leaking information about the S-box output, is available on Riscure's Sample Type Card 2. A total of 100 traces are available for analysis; two choices for the number of bins are compared. For each S-box, the number of traces needed to compromise the correct subkey is shown in Table 7.1 for a Hamming weight-based attack and in Table 7.2 for an identity-based attack. The results are presented per S-box because attacks focus on each S-box separately, guessing the corresponding round key bits.

Method \ Box:	1	2	3	4	5	6	7	8
Hist. 5	20	30	30	40	70	40	50	20
Hist. 16	50	60	50	50	100	70	60	70
Cum.	50	> 100	60	60	70	30	60	50
Corr.	10	20	20	20	30	20	20	30

Table 7.1: Number of traces required for a successful DES S-box output *Hamming weight* attack on Sample Type Card 2.

Method \ Box:	1	2	3	4	5	6	7	8
Hist. 5	50	50	40	70	60	60	30	40
Hist. 16	100	> 100	60	>100	70	>100	80	90
Cum.	30	50	20	20	30	20	20	30
Corr.	20	30	20	20	20	20	20	20

Table 7.2: Number of traces required for a successful DES S-box output *identity*-based attack on Sample Type Card 2.

A first observation from Table 7.1 and Table 7.2 might be the overall strong performance of mutual information attacks. However, even though there are mutual information analysis attacks effective within 100 traces, a correlation attack still performs better.

The Hamming weight-based prediction, with results in Table 7.1, is evidently adequate. More bins for observation generally require more traces to compromise a subkey correctly. With an

implementation leaking this much information, few traces are already sufficient to gather enough evidence for a subkey. Dividing few measurements over many bins yields a scattered pattern that might not adequately approximate the underlying distribution. Consequently, more measurements are required to come to the correct key guess. Five bins for observation seems to be a good choice, it is equal to the optimal choice in the AES attack from the previous section. A reason for this can be that both this card (Sample Type 2) and the AES card (Sample Type 3) are equipped with an 8-bit CPU, so power consumption depends on 8 bits.

Identity-based prediction methods are less efficient in number of traces. Moreover, for some subkeys, a histogram attack with 16 bins requires more than 100 traces to be compromised correctly. In contrast, cumulant-based MIA is more efficient using an identity prediction model. This is a surprising result, since identity-based predictions do not follow a (near) Gaussian distribution, but a uniform distribution. Due to the low number of measurements, however, this underlying distribution may not be adequately approximated. Also correlation attacks do not suffer from an identity prediction method. This means that both Hamming weight-based predictions and identity-based predictions are correlated with observations. However, one has to bear in mind that Hamming weight and identity of 4-bit values are correlated with $\rho = 0.81$, which (partly) accounts for the high score. Examining the joint distribution of prediction and observation in both cases can reveal more information.

Figure 7.8 shows a graphical representation of two histograms, based on 100 traces and Hamming weight prediction. Figure 7.9 visualizes two histograms obtained from an identity attack. In both figures, the histogram of the correct key guess is placed on the left; the histogram for the second best key guess is placed right. The score of the second best equals 58% for Figure 7.8 and 93% for Figure 7.9. As can be seen from the results, a correlation attack is successful since the leakage is globally linear, whereas the second best histogram in figure 7.8 is far from linear. Furthermore, a global linear trend can be seen in Figure 7.9 (left), but the diagram on the right is completely scattered.

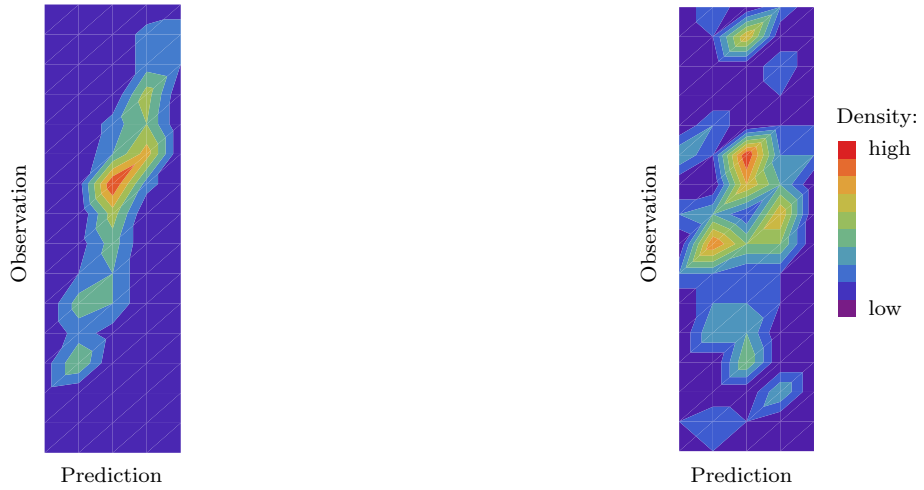


Figure 7.8: Histogram of correct key guess (left) and second best (58%) key guess (right) from the DES S-box output *Hamming weight* attack on Sample Type Card 2, using 16 bins for observation.

7.3 DES Round Output

This attack on a DES implementation aims at information leakage arising from processing the first round output. In particular, it aims at the left half of the output, which is equal to the right half of the second round input. Randomness of this intermediate value might be increased due to involvement of the left half round input, although theoretically this should not matter. Leakage

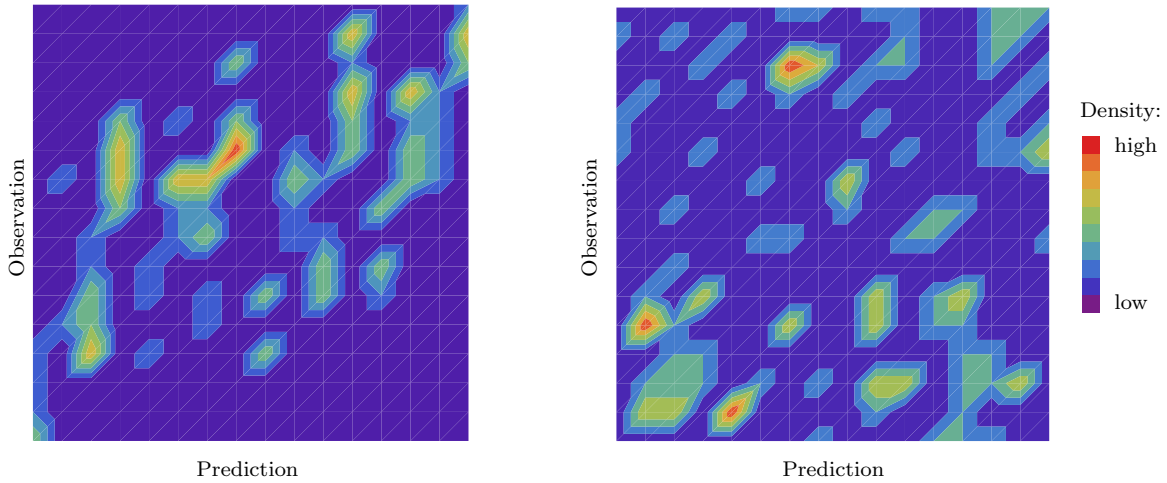


Figure 7.9: Histogram of correct key guess (left) and second best (93%) key guess (right) from the DES S-box output *identity* attack on Sample Type Card 9A, using 16 bins for observation.

depending on this intermediate is likely to occur, since it is often written to a place that is in turn used as input for a new round. When a register is reset before new results are written, leakage about the new result occurs. This happens in many hardware implementations.

Parameters of this attack:

Intermediate value: One bit ($b = (b_1)$) or four bits ($b = (b_1, \ldots, b_4)$) in the first round output l_1, coming from S-box S_i. The left half of the first round output is equal to the right half of the second round input. The attack is performed for $i = 1, \ldots, 8$, i.e.,on each of the 8 S-boxes.

Prediction: $\mathbf{H}_{\hat{k}} = HW(b)$ and $ID(b)$.

Observation: $\mathbf{O}_t$ is the power consumption partly based on the round output bits at time t.

7.3.1 Simulation

In this section, simulation results of a DES round output attack are presented. However, these results are equivalent with simulation results from a DES S-box output attack, as the only difference consists of a bijective permutation and a XOR with known data. Moreover, these results are theoretically equivalent with simulation results from a DES round input-output difference attack; only a XOR with the input is required.

Theoretically equivalent means that, under certain circumstances, no distinction can be made in behavior between two types of attacks. In practice, this does not hold, therefore practical attacks on all three intermediate values are presented in this chapter. However, in the simulations examined in this chapter, noise is modeled equally for every intermediate value, and no external influences from logical steps is taken into account. No distinction can therefore be made in simulation results; for this reason only one type of intermediate value is attack by means of simulations.

In this section, success rates depending on the number of traces, as well as depending on the number of bins for observation are estimated by means of simulations. Moreover, the noise resistance of this attack is assessed.

Success Rate

The success rate is determined on a noisy signal. Leakage is modeled as having a linear relation with Hamming weight, which suits a Hamming weight attack. Moreover, identity leakage is mod-

eled, which suits an identity attack.

Hamming weights of uniformly drawn 4-bit values are distributed binomially with variance 4/4=1 (see Section 4.4.2). The variance of uniformly drawn integers in range $1, \ldots, n$ equals $(n^2 - 1)/12$, hence the variance of identity equals 21.25.

For a fair comparison of identity-based attacks and Hamming weight-based attacks, the signal to noise ratio must be equal for both. The signal to noise ratio is chosen to be equal to 1/16, to have less noise than in the AES first order case, but enough noise to guarantee that at least 500 traces are required for any attack to have success. In order to achieve this signal to noise ratio, traces leaking Hamming weight are equipped with Gaussian noise with a variance of 16; traces leaking identity are equipped with Gaussian noise with a variance of 340.

Figure 7.10 depicts the success rate for different numbers of traces, for Hamming weight attacks (left) and identity attacks (right). Mutual information analysis, using histograms and cumulants for density estimation, is depicted, as well as correlation analysis for comparison.

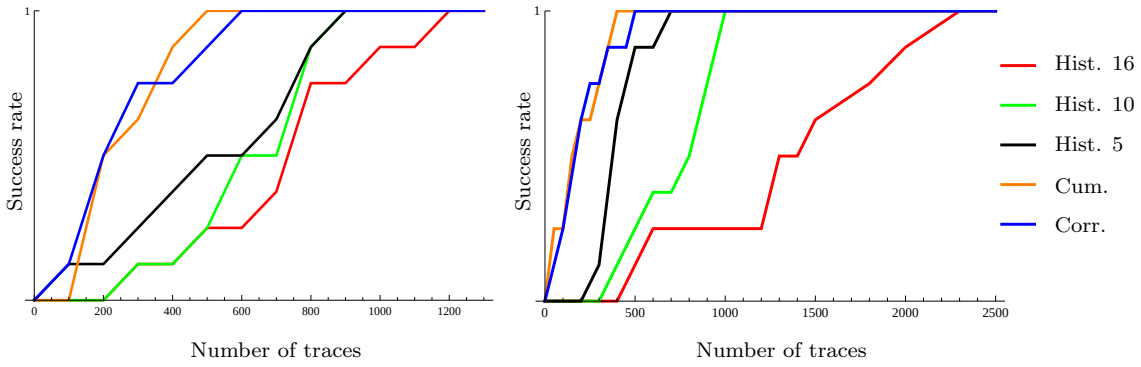


Figure 7.10: Success rate of a first order *Hamming weight* attack (left) and *identity* attack (right) on a DES simulation. Signal to noise ratio = $\frac{1}{16}$.

As can be seen in Figure 7.10 (left), cumulant-based MIA reaches full success first, shortly later followed by correlation analysis. This is good news for the mutual information lobby. However, histogram-based MIA performs much slower. Again, increasing the number of bins decreases performance. It is likely that using more bins increases noise vulnerability, which is also examined later in this section.

From Figure 7.10 (right), one can observe similar behavior for the identity-based attacks. Note that since identity leakage is also linear with identity prediction, correlation attacks are efficient. As is the case in Hamming weight models, more histogram bins require more traces to achieve the same success rate. However, reducing the number of bins for observation has much more advantages in performance of identity attacks than in Hamming weight attacks. Apparently, noise resistance might be weaker for identity attacks, something that is investigated later.

Number of Bins for Observation

To investigate the influence of the number of bins in a histogram attack, a new analysis is performed. Trace sets with signal to noise ratio of 1/16 are examined to determine the influence of the number of bins for observation on the number of traces required for a successful DES cipher key recovery. Figure 7.11 depicts the results for Hamming weight attacks (left) and identity attacks (right).

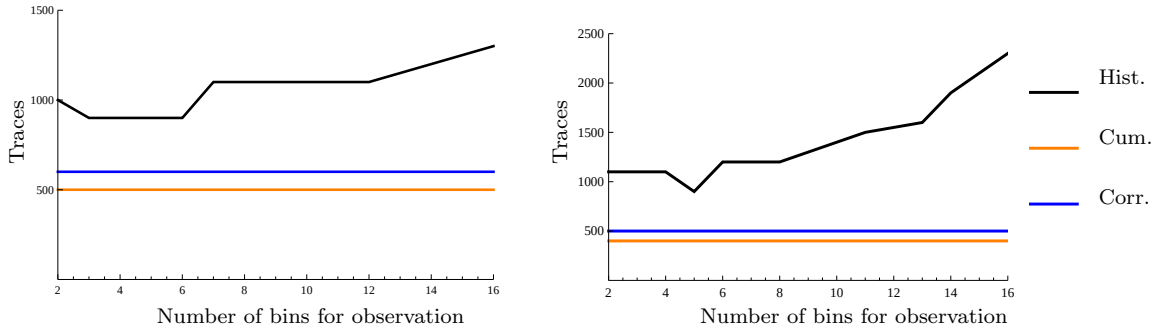


Figure 7.11: Influence of number of bins for observation on number of traces needed for a complete successful first order *Hamming weight* attack (left) and *identity* attack (right) on a DES simulation. Since both cumulant MIA and correlation analysis are independent from any bin decision, these lines are straight.

Figure 7.11 (right) shows a clear optimum at five bins for observation, whereas Figure 7.11 (left) shows a range of adequate bin choices. In both Hamming weight and identity simulations, more than seven bins for observation does not further improve the performance of the attack. The degradation point occurs at 12 bins (left), resp. 8 bins (right).

Resistance to Noise

The behavior of mutual information analysis on DES simulations in a noisy environment is assessed. Success rates are determined on multiple trace sets, where each trace set is generated using a different noise standard deviation. In this section, noise resistance is determined on a set of 1000 traces.

Figure 7.12 depicts the noise resistance of different mutual information analysis for Hamming weight attacks (left) and identity attacks (right), compared with noise resistance of correlation analysis.

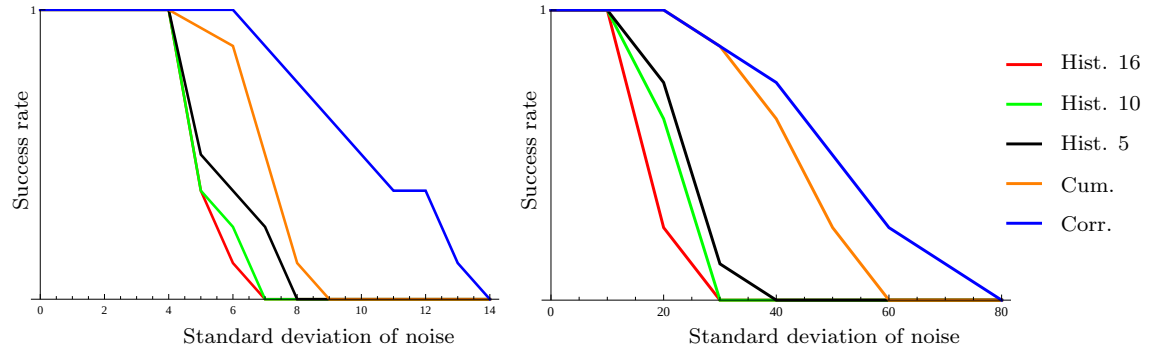


Figure 7.12: Noise resistance of a first order *Hamming weight* attack (left) and *identity* attack (right) on a DES simulation.

From Figure 7.12, one can observe that correlation analysis is more resistant to noise than any mutual information analysis. Moreover, cumulant-based mutual information analysis is far more resistant to noise than any of the histogram-based variants. The figure does not completely match the order of performance depicted in figure 7.10; cumulant MIA seems to be more vulnerable to large noise. Also, using more bins does not have such a huge effect on the noise resistance as it has

on the number of traces needed to correctly compromise subkeys, but still few bins are in favor over many bins.

That correlation attacks are the most noise-resistant attacks might be due to the scope of correlation attacks: linearity. Only linear dependence is rated, no other types of dependencies are inquired. Consequently, no noise can arise from possibly imaginary nonlinear relations. In MIA, however, 'any' dependence is exploited. Here, it can happen that a statistical fluctuation is viewed as a kind of nonlinear dependence. An improper score is awarded more likely, yielding a weaker noise resistance than a correlation attack.

7.3.2 High Security Hardware Embedding 1

A high security hardware DES embedding leaking information about the round output is available from Riscure's Sample Type Card 9. This hardware card contains a highly advanced contemporary DES embedding, equipped with an option to enable a countermeasure. Traces derived from this card when the countermeasure is turned on, are referred to as coming from Sample Type Card 9A. These traces are analyzed in this section. Round output is written to two 32-bit registers. Since the left half of the round output equals the right half of the input, the left half of the round output is written to the same register where the right half of the round input comes from.

The countermeasure is implemented as follows. Before the round output is written to the register, a random mask is injected into the register. When the round output is written to the register, information leakage about the Hamming distance of mask and round output might appear. This leakage should be theoretically independent from the round output, to prevent correlation attacks. However, the random number generator operates far from random. Viewing the non-random mask as a (approximately) fixed value, information leaking from the implementation indicates then the distance between the round output and a fixed value. Consequently, a first order attack on the round output suit this scenario better than a second order attack.

One might wonder why this type of masking is used. One of the main reasons is that in a hardware embedding, S-boxes are implemented as logical circuits. Consequently, no masked S-box can be created for every run. A masking scheme should henceforth be implemented in such a way that it suits the logic that is already embedded on the device.

The round output is attacked in sets of four bits, each set coming from one S-box. Besides correlation analysis, a second attack is used for comparison on this embedding, called summated difference-of-means (SDoM). This attack aims at each of the four bit coming from one S-box individually. The measurements are grouped into two bins according to the predicted value of one bit. The score of a subkey candidate is given by difference of the means of these two groups. This procedure is performed for each of the four bits from one S-box output and the results are added up. This defines the definitive score for every subkey candidate.

This summated difference-of-means scoring model relaxes assumptions made in Hamming weight models, for example the assumption that all bits have identical leakage. Therefore, this model can be stronger than a Hamming weight model in case of nonlinear leakage. In comparison with identity-MIA, this method is more practical when individual leakages from four bits can be added up. However, this method can be more vulnerable to noise, since also noise from four bits is added up. Identity-MIA may be more practical when some bits are more noisy than others.

Using a Hamming weight prediction model and an identity prediction model, the number of traces required to compromise each one of the eight S-box subkeys is compared with a correlation attack and summated difference-of-means attack. Table 7.3 shows the results for Hamming weight-based attacks, Table 7.4 shows results for identity-based attacks, expressed in thousands. Since DES S-boxes are not identical, leakage in round output can be different. Consequently, some S-box subkeys might be compromised more easily (i.e.,requiring fewer traces to get compromised) than others. The results are therefore presented per S-box subkey. Results for both 5 and 16 bins for observation are included, as well as on cumulant-MIA and correlation analysis. From this embedding, a total of 965392 traces are available for analysis. In case no number is given in a cell, this subkey has not been compromised within 965392 measurements.

Method, Subkey:	1	2	3	4	5	6	7	8
Hist. 5				200k			200k	400k
Hist. 16	300k			200k	250k		450k	250k
Cum.	400k			200k			600k	400k
Corr.	400k			100k			500k	400k
SDoM	60k	140k	20k	50k	130k	50k	110k	50k

Table 7.3: Number of traces required for a successful DES round output *Hamming weight* attack on Sample Type Card 9A. $k = 1000$.

Method, Subkey:	1	2	3	4	5	6	7	8
Hist. 5	300k	350k	50k	150k	250k	200k	200k	150k
Hist. 16	150k	350k	100k	450k	250k	200k	400k	200k
Cum.			400k	400k		300k	700k	
Corr.			350k	400k		150k		
SDoM	60k	140k	20k	50k	130k	50k	110k	50k

Table 7.4: Number of traces required for a successful DES round output *identity attack* on Sample Type Card 9A.

Table 7.3 indicates that a summated difference-of-means attack is the only successful Hamming weight-based attack. Hence, the countermeasure is successful in preventing correlation attacks. Moreover, Table 7.4 shows that a summated difference-of-means attack requires the fewest number of traces, compared to other methods, for each subkey. Since Hamming weight of one bit equals identity, summated difference-of-means scores equally in both attacks. The results for CPA and MIA are not in accordance with simulation results. One can conclude that assumptions which hold for simulations (such as Gaussian noise) do not hold in this practical case. Furthermore, one can see that Hamming weight prediction models to not match reality adequately, since this leakage model is not applicable on this card.

One can observe that five bins for observation is a better choice than 16 on subkey seven, but the converse is true for subkey eight. From the efficiency of CPA and SDoM, one can conclude that subkey seven comes from a more noisy S-box. Hence, the results on subkey seven confirm that increasing the number of bins decreases noise-resistance. However, when noise is lower, choosing more bins can yield more accurate approximations of distributions, and hence better distinguishing properties. This is the case at subkey eight.

When comparing genuine correlation attacks with mutual information attacks, MIA is strongly in favor. Due to the countermeasure, correlation attacks do not yield adequate results. However, since masks are not sufficiently random, enough information is leaked for a mutual information identity attack to succeed. This shows feasibility of MIA on practical implementations, where correlation analysis fails.

Figure 7.13 shows results on the number of traces required for a complete (i.e.,all 8 S-box subkeys) successful attack on Sample Type 9A, in relation to the number of bins for observation in a histogram. Note that only identity-prediction MIA and summated difference-of-means are depicted, since these are the only two successful types of attack on this trace set. As can be seen, 5 bins for observation yields the most efficient result. Furthermore, increasing the number of bins does not necessarily decrease performance. This might be due to the fact that already many measurements are required to sufficiently discriminate subkey guesses. This figure is significantly different from simulation results depicted in Figure 7.11. Clearly, power consumption of this card is not nicely related to Hamming weight or identity of intermediate values. Moreover, noise does not seem to be Gaussian.

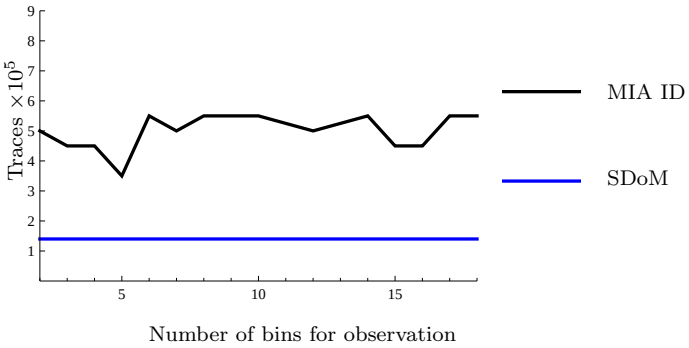


Figure 7.13: Number of traces needed for a complete successful attack on Sample Type 9A, in relation to the number of bins for observation.

Since both correlation attacks and Hamming weight mutual information attacks are not completely successful within the number of available traces, leakage might be nonlinear up to a certain extent. Moreover, an identity mutual information attack *does* compromise the complete cipher key, further confirming nonlinear leakage. Cumulant MIA is approximately as good as correlation analysis, possibly due to non-Gaussian behavior of leakage.

Especially in hardware implementations, nonlinearity of leakage often occurs. A hardware embedding of an algorithm like DES requires series of many logical circuits, switches and gates. Therefore, the path one bit travels might be significantly different from the path another bit travels. Noise can depend on the path one bit travels, and may consequently be different for every bit. A bit flip from zero to one can have different power consumption properties than a bit flip from one to zero. Furthermore, since many algorithmic step are performed in parallel on a hardware embedding, noise from other components might be present on parts of the signal.

A graphical representation of histograms for both Hamming weight and identity attacks are shown in Figure 7.14, resp. Figure 7.15. These images represent the joint density of prediction and observation for the correct subkey guess and the second best subkey guess. In this case, the subkey under attack is the first S-box key in round one. As can be seen, an identity model for prediction yields a nearly uniform distribution of predictions, whereas a Hamming weight model yields a Gaussian-like distribution. The second-best response scores 93% of the correct response in case of a Hamming weight model. In case of an identity model, the second best guess scores 94%.

As can be seen in Figure 7.14, the dependence of prediction and observations seems to resemble the sum of two bivariate Gaussian distributions. Likewise, the marginal distribution of observation appears to have two peaks in Figure 7.15. Since this behavior does not suit a normally distributed random variable, the requirements for a cumulant method to succeed are not met. This clarifies why the cumulant method does not give a correct score.

Although this card is protected by a masking countermeasure, a first order mutual information attack is able to correctly compromise the cipher key. A second order attack on this trace set has been performed, this attack does not yield the cipher key within the 965392 available traces. Furthermore, no second order correlation attack, using absolute difference preprocessing, is effective within this number of traces. The a second order attack is more involved, one can conclude that the countermeasure does prevent first order correlation analysis, but is not implemented that well to prevent any first order attack, like a MIA.

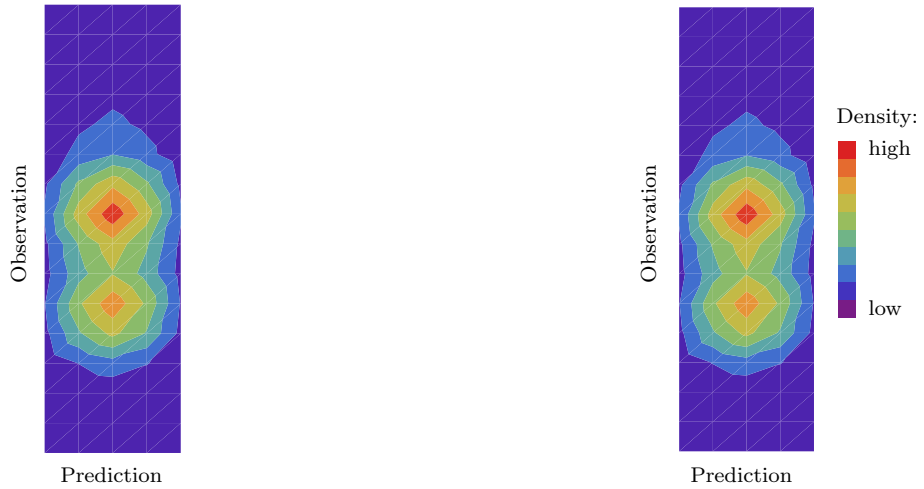


Figure 7.14: Histogram of correct key guess (left) and second best (93%) key guess (right) from a DES round output *Hamming weight* attack on Sample Type Card 9A.

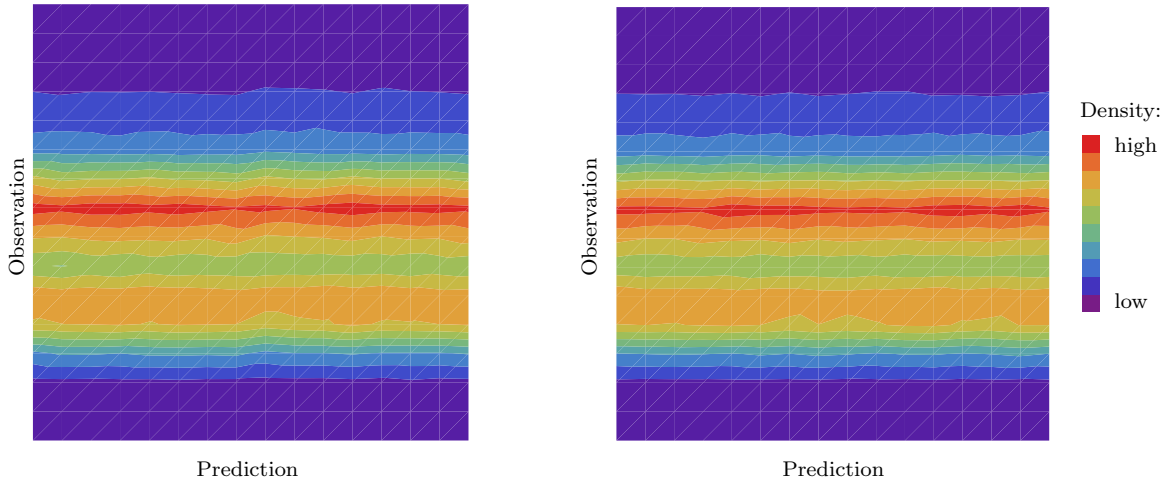


Figure 7.15: Histogram of correct key guess (left) and second best (94%) key guess (right) from a DES round output *identity attack* on Sample Type Card 9A.

7.3.3 High Security Hardware Embedding 2

The high security embedding analyzed in the previous section has the option to enable countermeasures. In this section, the same card is analyzed while countermeasures are disabled. Traces from this card are referred to as coming from Sample Type Card 9B, a total of 48791 traces is available. Still, this hardware embedding is expected to leak information in a nonlinear way; the properties of Sample Type Card 9A as described in the previous section trivially apply to Sample Type Card 9B.

Instead of injecting a mask, the register is reset to zero before a value is written to it. Hence, only absolute power models apply; no difference models. Again, the left output half is attacked. Both MIA with a Hamming weight model and MIA with an identity model are compared to correlation attacks and summated difference-of-means attacks. Table 7.5 shows results for Hamming weight-based attacks, Table 7.6 shows results for identity-based attacks. In case no number is given in a cell, this subkey has not been compromised within 48791 measurements. A k represents 1000.

From Table 7.5, one can observe that no method is able to compromise all S-box subkeys within the number of available traces when a Hamming weight-based prediction model is used. Mutual

Method, Subkey:	1	2	3	4	5	6	7	8
Hist. 5			10k	40k	10k	20k	30k	30k
Hist. 16	25k		35k		10k	20k	15k	30k
Cum.	48790			20k				
Corr.				20k				
SDoM		44k						22k

Table 7.5: Number of traces required for a successful DES round output *Hamming weight* attack on Sample Type Card 9B.

Method , Subkey:	1	2	3	4	5	6	7	8
Hist. 5	15k	20k	10k		5k		30k	25k
Hist. 16	10k	20k	30k	30k	38490	25k	30k	25k
Cum.		45k	350k	35k				
Corr.				30k				
SDoM		44k						22k

Table 7.6: Number of traces required for a successful DES round output *identity* attack on Sample Type Card 9B.

information analysis with histograms for density estimation is able to correctly find six of the eight subkeys, whereas cumulant MIA, correlation and summated difference-of-means only find one or two. The most interesting result from this table is the fact that a summated difference-of-means attack does not yield decent results, unlike in Table 7.3. Summing up scores of four bits might suffer from noise, or some bits are not suitable to use for distinguishing key guesses.

A more striking result follows from Table 7.6: a mutual information attack *does* yield a successful cipher key recovery using identity predictions. This proves the functionality of mutual information and the identity prediction model on a real embedding once more; a Hamming weight prediction model does not suit information leakage from this card. Moreover, nonparametric histogram method appear to be adequate, as compared to the parametric cumulant method. Simulated attacks in Section 7.3.1 show significantly different results. This shows that assumptions made in simulations (linear leakage) do not always hold in practice.

Furthermore, one might observe that selecting more bins for observation yields better results; specifically, the cipher key is compromised when using 16 bins for observation, but not when using five bins. A deeper investigation in the influence of bins choices can give clarification. Figure 7.16 shows the number of traces required for a successful Hamming weight, resp. identity attack on this trace set, in relation to the number of bins for observation.

From Figure 7.16, one can observe that selecting five bins for observation was no good choice. Where the complete cipher key is not recovered by Hamming weight attacks nor identity attacks, six bins for observation do the trick. The global minimum in number of traces required to break is at six bins, for both prediction methods. Using more bins, a Hamming weight attack quickly is not feasible on this trace set anymore. An identity attack, however, yields the correct cipher key, but using more traces. An interesting dip occurs at 16 bins: this is exactly equal to the number of bins for prediction. Thus, the histogram is perfectly square. When more bins for observation are used, an identity attack no longer compromises the full cipher key within the available number of traces, providing evidence that noise is dominant when more bins are used.

Curious about the reason a summated difference-of-means attack does not give desired results on the unprotected embedding, another analysis is performed. Instead of attacking four bits in the round output, coming from the same S-box, a single bit is attacked. This means that key candidates are scored based on predictions of one bit. For correlation attacks, this is equivalent

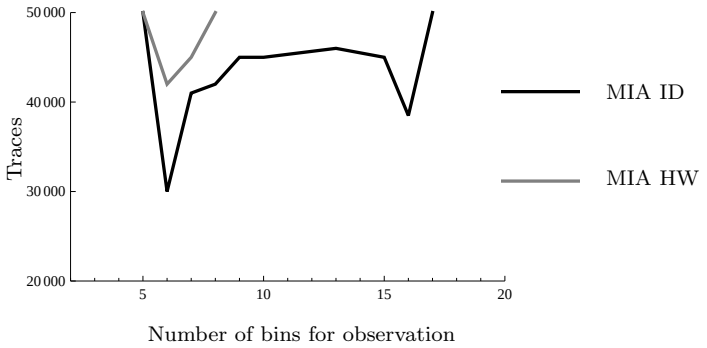


Figure 7.16: Number of traces needed for a complete successful attack on Sample Type Card 9B, in relation to the number of bins for observation.

to a difference-of-means attack on one bit, i.e., not the sum of four bits. For mutual information attacks, a density for observation is created for both possible values of the bit. Figure 7.17 shows how information about S-boxes leaks through different bits in a correlation attack. A black square for a (bit, S-box) pair (i, j) means information leakage is strong: an analysis based on this output bit i ranks the correct subkey for S-box j first. The lighter a square is, the weaker rankings based on this bit are. Figure 7.18 shows how information about S-boxes leaks through different bits in a mutual information attack.

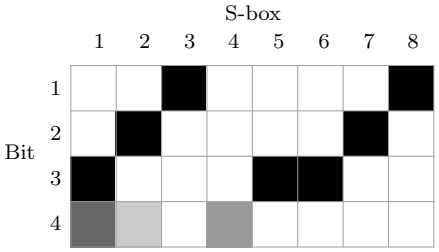


Figure 7.17: Leakage specified by S-box bits in a *correlation* attack.

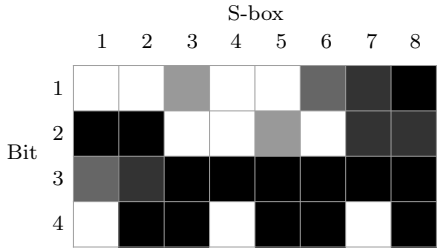


Figure 7.18: Leakage specified by S-box bits in a *mutual information* attack.

From Figure 7.17, one can distill a reason for the failure of summated difference-of-means attacks on this trace set. For every S-box, information leaks specifically through one bit. When calculating scores per box from all four bits, three of the four bits merely add noise, rather than useful information. Hence, adding up scores for all four bits, might yield no adequate scoring for every subkey. Although leakage is nonlinear in this case, no advantage can be taken from summing up results from single bits.

In Figure 7.18, much more bits are adequate as intermediate value in a MIA. Apparently, estimation of density yields more information than only the difference of means. Moreover, a regular mutual information attack is based on all four bits. The more bits give individual contribution, the more adequate an attack is. Furthermore, this figure might explain why S-box subkey 4 and 6 are not compromised by a 5-bin identity attack (see Table 7.6); especially information on the fourth S-box subkey seems to leak through only one bit.

A graphical representation of the histograms for both Hamming weight and identity attack are shown in Figure 7.19, resp. Figure 7.20. These figures resemble the histograms in Section 7.3.2. The second-best response scores 92% of the correct response in case of an identity model. In case of a Hamming weight model, the second best guess scores 90%.

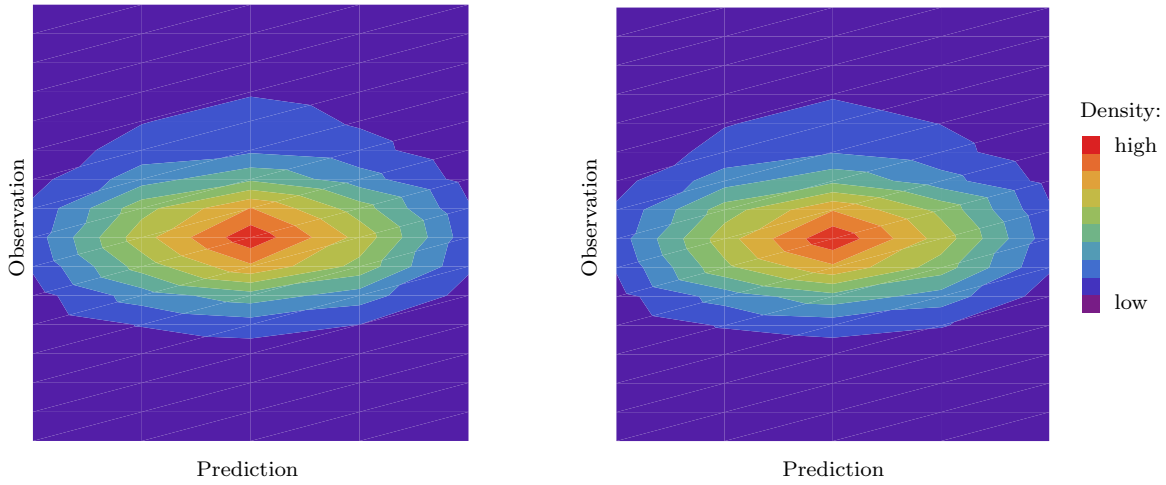


Figure 7.19: Histogram of correct key guess (left) and second best (90%) key guess (right) from a DES round output *Hamming weight* attack on Sample Type Card 9B.

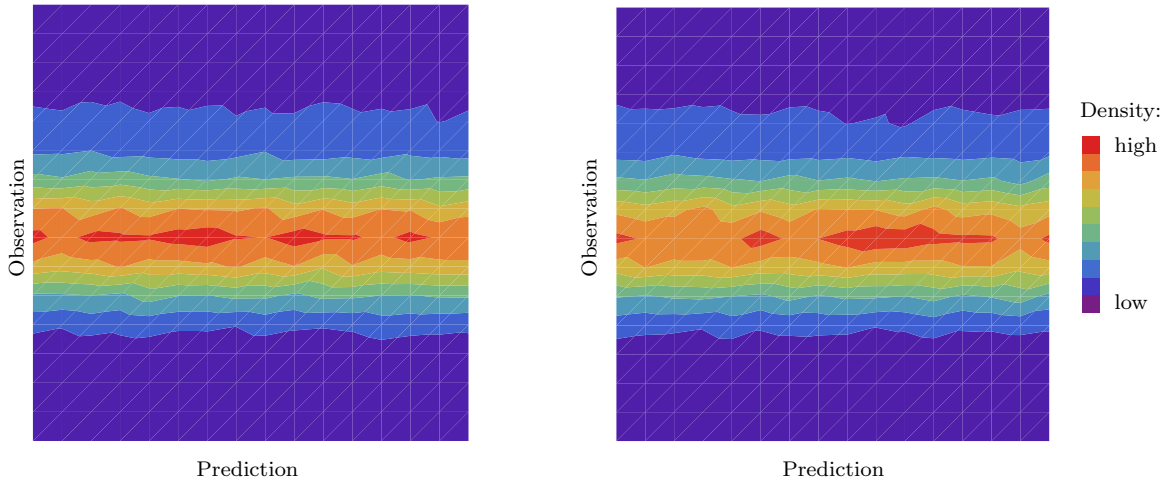


Figure 7.20: Histogram of correct key guess (left) and second best (92%) key guess (right) from a DES round output *identity* attack on Sample Type Card 9B.

Finding a linear dependence in both graphs is very difficult. Moreover, the highest correlation in a CPA attack is around 0.01. This supports the fact that correlation analysis is not very powerful on this trace set. However, although cumulant MIA is not efficient on this trace set, one Gaussian peak is visible for observations.

7.4 DES Round Difference

This attack on DES aims at the difference between input and output of the first round. In many hardware implementations, round input and output are written to the same register, since output is again equal to the input of a next round. Therefore, when writing round output to a register only bits are flipped in case the input and output are unequal at that bit. Power consumption might then be related to the difference between these two values.

Hamming distance between round input and round output can be predicted, as well as the identity of the difference.

Parameters of this attack:

Intermediate value: Four bits of round input, four bits of round output. Recall that output of round i equals round input of round $i + 1$.

Prediction: $\mathbf{H}_{\hat{k}} = HD(R_1, R_2)$ and $ID(R_1 \oplus R_2)$.

Observation: $\mathbf{O}_t$ is the power consumption partly based on distance between the round input and output bits at time t.

As stated before, no simulation results on this attack are presented. As predicting round input-output difference requires knowledge of round input and output, which is theoretically equal to a round output attack, only simulations on round output attacks are presented. For these simulations, see Section 7.3.1.

7.4.1 High Security Hardware Embedding - Power Traces

A DES hardware implementation leaking distance of round input and output is available from Riscure's Sample Type Card 8. This card is older than Sample Type Card 9, which is attacked in the previous Section. On this card, each DES round is performed during one clock cycle, and the round outputs are written to the same 32-bit register. Both power measurements and electromagnetic radiation measurements are available, the power traces are referred to as coming from Sample Type Card 8A, to distinguish from the electromagnetic traces used in the next section.

Using Hamming distance prediction model and identity distance prediction model, the number of traces required to compromise an S-box subkey are compared with a correlation attack. Table 7.7 shows the results for Hamming distance-based attacks. Moreover, the results presented in Table 7.7 are depicted in Figure 7.21. An attack with identity-based prediction models yields no single subkey success within the available 10000 traces. Therefore, no results on identity-MIA are presented. Histogram-based attacks are performed using 5, 10 and 16 bins for observation, to grasp the influence of bin-decisions.

Method Box:	1	2	3	4	5	6	7	8
Hist. 16	2000	6000	3000	3000	3000	3000	3000	8000
Hist. 10	3000	3000	3000	3000	4000	4000	2000	5000
Hist. 5	3000	2000	2000	2000	2000	3000	2000	4000
Cum.	2000	2000	1000	2000	2000	3000	2000	4000
Corr.	1000	2000	1000	1000	1000	2000	1000	1000
SDoM	2000	2000	1000	1000	2000	2000	1000	2000

Table 7.7: Number of traces required for a successful DES round output *Hamming distance* attack on Sample Type Card 8A.

As can be seen from both Table 7.7 and Figure 7.21, correlation-based attacks are most efficient. Cumulant MIA scores better than any histogram-based mutual information attack. This can indicate that predictions and observations both follow a near-Gaussian distribution; favorable for parametric density estimation models, as indicated by Flament et al. [FGD+10].

In histogram attacks, using more bins for observation does not necessarily have a negative influence on performance, but using five bins yields the overall best performance among the bin choices. Furthermore, summated difference-of-means attacks indicate that adding up scores for individual bits yields adequate values to correlate with predictions. Among others, a reason for this might be that bits have nearly equal contribution to leakage.

An identity attack does not compromise any of the subkeys within 10000 traces. Since the correlation attack is successful in an early stage, a nearly linear dependence between power consumption and Hamming weight of processed values is likely. An identity attack fails in this case, because it assumes a resolution which is too high. Too many traces are required to discriminate

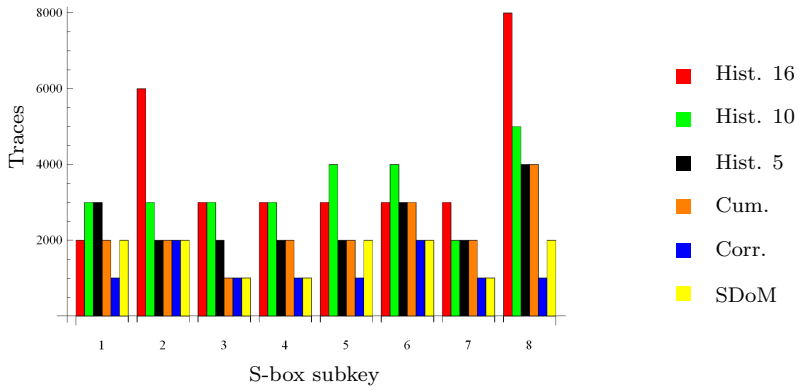


Figure 7.21: Visualization of Table 7.7 for subkey of S-box $1, \ldots, 8$.

correct and wrong subkey guesses. In order to verify linear dependence, a graphical representation of histograms can be examined. The histogram visualizations resemble those depicted in Figure 7.19 and are not included for conciseness. The figure confirms the near-Gaussian distribution, supporting the performance of cumulant MIA.

Simulated attacks as described in Section 7.3.1 show similar behavior to the results presented in this section, at least, for Hamming distance-based prediction models. Mutual information analysis using an identity prediction model does not yield adequate results on this practical embedding and can henceforth not be compared with simulations.

7.4.2 High Security Hardware Embedding - Electromagnetic Traces

Side channel analysis is typically performed on power traces. However, multiple other signals leaking from a device might be used. One example is the electromagnetic radiation a smart card emits while performing any (cryptographic) operation. Electromagnetic radiation emerging from smart cards is proportional to power consumption. A DES hardware implementation leaking distance of round input and output via electromagnetic radiation is available from Riscure's Sample Type Card 8. On this card, which is the same card as the one under attack in Section 7.4.1, each DES round is performed during one clock cycle, and the round outputs are written to the same 32-bit register. The traces are referred to as coming from Sample Type Card 8B, to distinguish from the traces used in the previous section.

For both a Hamming distance prediction model and an identity distance prediction model, the number of traces required to compromise an S-box subkey using MIA are compared with a correlation attack. Table 7.8 shows the results for Hamming distance-based attacks. Moreover, the results presented in Table 7.8 are depicted in Figure 7.22. As is the case when attacking power traces (in Section 7.4.1), an attack with identity-based predictions yields no single subkey successfully within the available 16000 traces. Therefore, no results on identity-MIA are presented. Histogram-based attacks are performed using 5, 10 and 16 bins for observation, to see the influence of bin-decisions.

The results presented in Table 7.8 and Figure 7.22 resemble the results on power traces in Table 7.7 and Figure 7.21. An identity attack does not compromise any of the subkeys within the 16000 available traces, so also electromagnetic radiation seems to have a linear relation with Hamming weight of processed values. S-box subkeys 2 and 8 are hard to compromise, while S-box subkeys 1 and 4 are compromised early. This trend also holds in the results on power traces as presented in the previous section. Clearly, electromagnetic measurements include more noise than power measurements on this card. Not only are more traces required to correctly compromise subkeys, also using more bins for observation does in some cases affect the performance quite negatively.

Mutual information analysis using Hamming distance predictions yields results which globally resemble results from simulations. However, due to higher noise (as compared to the power traces analyzed in Section 7.4.1), some subkeys are more difficult to compromise.

Method Box:	1	2	3	4	5	6	7	8
Hist. 16	5000	>16000	>16000	4000	>16000	5000	>16000	>16000
Hist. 10	5000	9000	4000	5000	9000	6000	12000	11000
Hist. 5	4000	>16000	5000	3000	9000	6000	9000	16000
Cum.	4000	7000	3000	2000	10000	6000	3000	16000
Corr.	3000	6000	4000	2000	2000	3000	3000	12000
SDoM	4000	3000	1000	2000	4000	2000	3000	5000

Table 7.8: Number of traces required for a successful DES round output *Hamming distance* attack on Sample Type Card 8B.

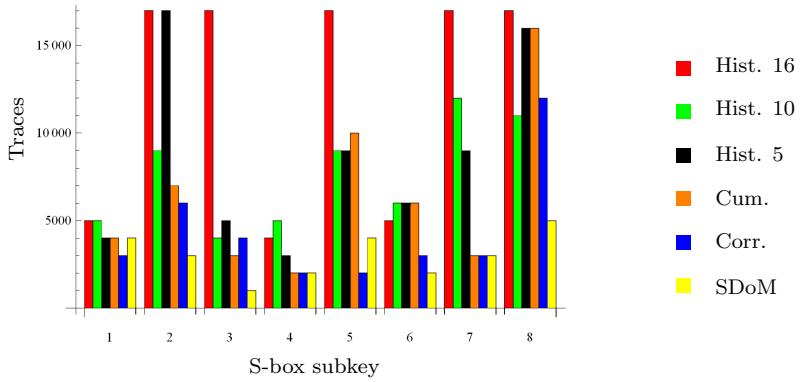


Figure 7.22: Visualization of Table 7.8 for subkey of S-box $1, \ldots, 8$.

7.5 Masked AES S-box

Masking is a countermeasure to prevent an attacker from obtaining information about intermediate values from a first order attack. However, second order attacks might be successful to a masked implementation. An AES masking scheme, where AES S-box input and output are masked with the same mask m, is explained in Section 5.1.2. In certain articles about MIA (e.g., Gierlichs et al. [GBPV10]), it is assumed that an attacker can obtain information about both $S(x) \oplus m$ and m. Whereas the masked S-box output $S(x) \oplus m$ might be leaking, the mask m is not likely to leak itself. However, the masked S-box input, $x \oplus m$, is likely to leak. Hence, this attack aims at the masked input and output of an AES S-box.

Parameters of this attack:

Intermediate values: Masked S-box input $x \oplus m$ and masked S-box output $S(x) \oplus m$.

Prediction: $\mathbf{H}_{\hat{k}} = HD(x \oplus m, S(x) \oplus m) = HW(x \oplus S(x))$.

Observation: $\mathbf{O}_{t_1}$ is the power consumption partly based on the masked S-box input $x \oplus m$ at time t_1, $\mathbf{O}_{t_2}$ is the power consumption partly based on the masked S-box output $S(x) \oplus m$ at time t_2.

Information about the mask can leak at multiple other positions, for example, while generating a masked S-box (which is generated as a lookup table in software embeddings). However, these leakages do not provide enough information to fully obtain (and hence remove) the mask. Since every run is masked with a new, randomly generated mask, only a simple power analysis on one trace may yield information about the mask. However, most contemporary implementations are sufficiently protected to prevent simple power analysis attacks.

The mask can also leak at more positions, in case other intermediate values are masked with the same mask. Actually, any two intermediate values which are masked by the same m can be

used to compromise subkeys. In this attack, S-box input $x \oplus m$ and S-box output $S(x) \oplus m$ are used, but it might as well be m and $S(x) \oplus m$, or $S(x) \oplus m$ and $S(y) \oplus m$. As long as the XOR of both intermediate values can be predicted, these attacks are equivalent, since masks are canceled out by a XOR. However, in practice attack results can be different, e.g., due to noise, or to the fact that information from one intermediate value is more vulnerable to leakage than another.

S-box masking is not widely used to protect AES hardware implementations on smart cards, since it requires much physical equipment to imitate a masked S-box. Therefore, only a software embedding is examined in this section. Second order correlation attacks are performed using the absolute difference combining function, as introduced in Definition 3.3.2.

7.5.1 Simulation

The success rate and the resistance to noise are estimated by means of simulations. Moreover, an investigation into the influence of the number of bins for observation is made. The four different higher order generalizations of mutual information introduced in Section 5.4.2 (being interaction information (II), generalized mutual information (GM), total correlation (TC) and dual total correlation (DT)) are compared to cumulant-based second order MIA and second order correlation power analysis. Figure 7.23 depicts three of the four generalizations.

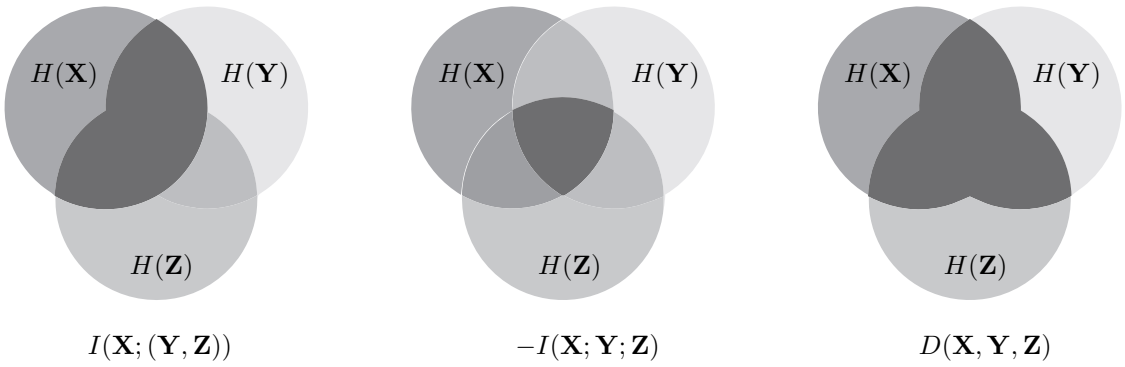


Figure 7.23: Information diagrams for different generalizations of mutual information. From left to right: generalized MI, interaction information (negative) and dual total correlation.

Success Rate

The success rate for different numbers of traces is determined on a set of simulated power traces with signal to noise ratio of 1/2. Since S-box input and output are both eight bits long, an identity attack would require 256 bins for prediction, which computationally intensive. Analogous to the results presented on unmasked AES implementations, only Hamming weight-based prediction models are used. Figure 7.24 shows the success rate for different types of attacks. Five bins for observation are used in histograms.

From Figure 7.24, one can obtain interesting results. First of all, a correlation attack is most efficient. Apparently absolute difference is a suitable combining function, and is power consumption sufficiently correlated with the outcome of this function. Furthermore, cumulant-based mutual information analysis performs significantly better than any of the histogram attacks. This order of performance resembles the results on unmasked AES simulations, as given in Figure 7.2.

Among the mutual information generalizations, interaction information dominates the other three. Total correlation seems to perform the worst; however, the same number of traces to fully compromise the AES cipher key is required as in an attack using the generalized mutual information. Gierlichs et al. [GBPV10] compare generalized mutual information and interaction information, there interaction information performs best on Hamming weight prediction models; however, their results are obtained from a practical implementation.

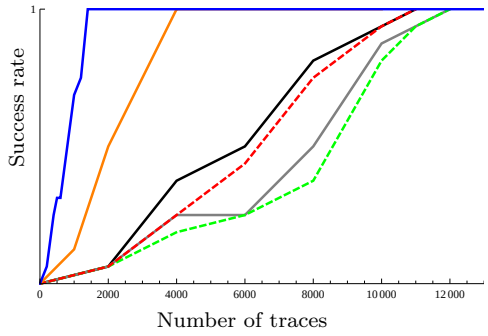


Figure 7.24: Success rate of a second order *Hamming weight* attack on a masked AES simulation.

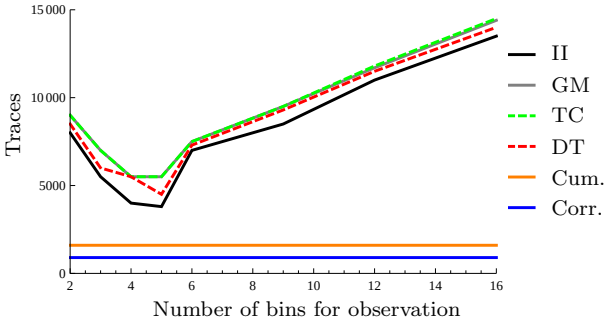


Figure 7.25: Influence of number of bins for observation on number of traces needed for a successful second order *Hamming weight* attack on the first key byte in a masked AES simulation.

Number of Bins for Observation

The number of traces required for a successful attack on the first AES key byte is determined for different numbers of bins for observation. In Figure 7.25, results on a trace set with signal to noise ratio of 1/2 are given. As can be seen, selecting five bins yields the most efficient histogram attack, for all higher order generalizations. The order in which the mutual information generalizations score resembles the order as depicted in Figure 7.24: interaction information scores best among all.

The bin degradation point can be found at six bins for observation, which is still less than the number of bins for prediction. Apparently, a five bins are sufficient to adequately estimate a Gaussian distribution, like Hamming weight of observations with Gaussian noise.

Resistance to Noise

Noise resistance on second order attacks is investigated on a set of 1000 traces. Moreover, to better distinguish the four higher order mutual information generalizations, noise resistance of these methods is further investigated using a set of 10000 traces. Figure 7.26 (left) depicts the results on 1000 traces; Figure 7.26 (right) for 10000 traces. Five bins for observation are used in histogram MIA attacks.

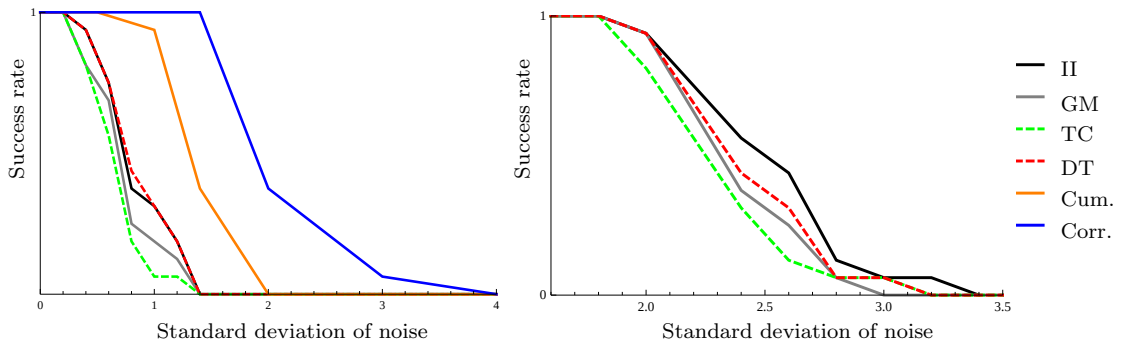


Figure 7.26: Noise resistance of a second order *Hamming weight* attack on a masked AES simulation. Left: on 1000 traces. Right: four mutual information generalizations compared on 10000 traces.

The results in Figure 7.26 (left) match the results on unmasked AES as depicted in Figure 7.5. Clearly, correlation with the absolute difference of two power consumption values yields the most noise-resistant attacks. Assumptions made in the simulated model might however not always hold in practice.

Figure 7.26 (right) indicates noise resistance of the higher order mutual information generalizations as introduced in Section 5.4.2. Interaction information seems to be most resistant to noise. This can easily be explained by looking at the information diagrams in Figure 5.2: interaction information represents only the overlap on all three circles. Therefore, no noise from 2-variable mutual information is included. Generalized mutual information does include full mutual information of prediction and observation, for two observations. This can be useful in case mask generation is not completely random, but might as well consist of noise.

Scores from dual total correlation are equal to the sum of scores from interaction information and total correlation. Therefore, it can be seen as the average of these two methods. In most of the measurements, this relation is reflected.

7.5.2 Software Embedding

A software embedding of masked AES is available on Riscure's Sample Type Card 3. This is the same card as the one analyzed in Section 7.1.2, but now countermeasures are enabled. Figure 7.28 indicates the progress in number of correctly guessed keybytes is depicted as function of the number of traces, for those methods that actually did compromise the full AES key. The number of traces required to fully compromise the AES key can be obtained from the figure. Five bins for observation are used in the histogram attacks.

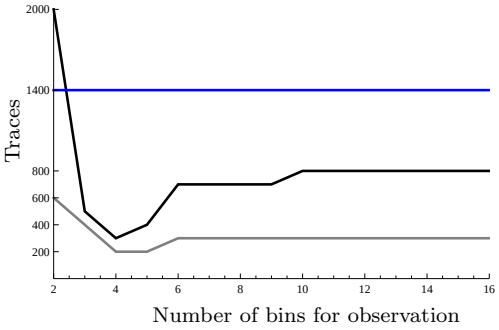


Figure 7.27: Influence of number of bins for observation on number of traces needed for a successful second order *Hamming weight* attack on the first key byte on masked Sample Type Card 3.

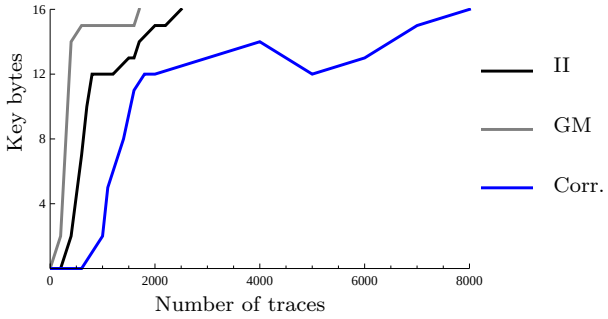


Figure 7.28: Number of correctly compromised key bytes on the number of traces in a second order *Hamming weight* attack on masked Sample Type Card 3.

A first observation from Figure 7.28 is that cumulant-based MIA does not give an adequate score, neither do total correlation and dual total correlation. After 8000 traces, no single key byte is correctly compromised by any of these methods. Furthermore, no single key byte is correctly compromised by the cumulant method, using 8000 traces. This is in contrast with simulation results. Apparently, noise from mutual information of two observations is of significant influence. An extensive analysis of this phenomenon is presented at the end of this section.

Figure 7.28 indicates a clear distinction in performance between the three successful models. Striking is the efficiency in which all three methods are able to recover twelve key bytes. However, the last four bytes are more difficult to compromise. An generalized mutual information attack is most efficient in discriminating key guesses for the last four key bytes, whereas a correlation-based attack requires three times more traces to compromise the last four bytes, compared to the first

twelve. Since it is able of correctly retrieving all key bytes, the absolute difference preprocessing function seems to be adequate on this trace set; however, it strongly reduces the amount of information coming from paired observations. The fact that generalized MI performs better as distinguisher on this trace set than interaction information, might be due to nonuniform masks. In that case, mutual information of prediction and one observation is not equal to zero. This yields more information about the likelihood for subkey guesses of being correct. However, this nonuniformity can not be verified.

Gierlichs et al. [GBPV10] and Batina et al. [BGP+11] present results about generalized mutual information and interaction information; however, applied to attack practical *DES* embeddings with 8-bit CPU. They indicate that interaction information and generalized mutual information perform nearly equivalent when using Hamming weight-based predictions, in low-noise scenarios. In high-noise scenarios, interaction information is preferred. Furthermore, on attack with identity-based predictions, using generalized MI as distinguisher is more efficient on low-noise implementations. On high noise, however, interaction information information and generalized MI tend to coincide. However, second order CPA is in favor of any MIA method in their results, unlike the results presented in this section. Apparently, the preprocessing function used on Sample Type Card 3 is less adequate than the preprocessing they used on their implementations. Although DES and AES are not completely comparable, the results in this section on different MIA distinguishers show similarities to those presented by Gierlichs et al. and Batina et al. on DES, showing advantages for generalized mutual information over interaction information.

The influence of the number of bins for observation is investigated on the real embedding. A second order attack on the input and output of the first AES S-box is performed, where the number of bins varies. Figure 7.27 shows the results for two MIA successful scoring models. For comparison, a third line is drawn according to the number of traces required in a correlation attack, being 1400 traces.

From Figure 7.27, one can observe that selecting four to five bins for observation yields the most efficient attacks. In this figure, a small increase in number of traces can be observed between nine and ten bins for an interaction information attack. Since nine bins equals the number of bins for prediction, the peculiar trend that attacks to not become more efficient when more bins are used for observation than for prediction is reflected here. However, no strict increase in number of traces can be observed when increasing the number of bins for observation; performance remains at a steady level.

Total correlation and dual total correlation. Total correlation and dual total correlation do not yield adequate scores, the same holds for second order cumulant MIA. After 8000 traces, no single key byte is correctly compromised. Consequently, no score is presented in Figure 7.28. This misbehavior can be due to the fact that both distinguishers use more overlapping areas of the information diagrams depicted in Figure 7.23. The part of information included in total correlation and dual total correlation, but not in interaction information or generalized mutual information is given by, e.g., $D(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) - I(\mathbf{X}; (\mathbf{Y}, \mathbf{Z}))$. In terms of entropy and conditional mutual information of $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = (\mathbf{H}, \mathbf{O}_1, \mathbf{O}_2)$, this part can be written as

$$H(\mathbf{H}, \mathbf{O}_1) + H(\mathbf{H}, \mathbf{O}_2) - H(\mathbf{H}) - H(\mathbf{H}, \mathbf{O}_1, \mathbf{O}_2) = I(\mathbf{O}_1; \mathbf{O}_2 | \mathbf{H}).$$

It can be interpreted as the mutual information of two observations, independent from predictions. This factor can specifically have influence on practical implementations.

Theoretically, $\mathbf{O}_1$ and $\mathbf{O}_2$ are independent when power consumption observations are related to masked variables. However, when other variables are processed, power consumption can show dependence. This dependence yields a mutual information value $I(\mathbf{O}_1; \mathbf{O}_2)$ greater than zero. Consequently, when a distinguisher takes this mutual information value into account, ghost peaks may appear. This implies that the correct key is not top-ranked. Furthermore, this means that ranking is based on power consumption measurements at wrong points in time. This problem is

not encountered in simulation results, since the points in time where information leakage occurs are known. On practical implementations, narrowing down the search space to those samples where information leaks is beneficial; however, an attacker might not know the exact location of leakage.

Based at the results in this section, it would be advisable to start an analysis using either interaction information or generalized mutual information as distinguisher. These methods have proven to be less affected by measurements at incorrect time points.

7.6 Masked DES Round Output

A masking countermeasure that suits DES is explained in Section 5.1.1. Since each S-box gives rise to four bits of the round output, four bits can be attacked simultaneously in quest for an S-box subkey.

Parameters of this attack:

Intermediate value: Four bits in the first round output l_1, coming from S-box S_i. The left half of the first round output (l_1) is equal to the right half of the second round input. The attack is performed for $i = 1, \ldots, 8$, i.e., on each of the 8 S-boxes.

Prediction: $\mathbf{H}_{\hat{k}} = HW(b)$ and $ID(b)$.

Observation: $\mathbf{O}_{t_1}$ is the power consumption partly based on the masked S-box input $x \oplus m'$ at time t_1, $\mathbf{O}_{t_2}$ is the power consumption partly based on the masked S-box output $S(x) \oplus m'$ at time t_2. The mask m' depends on the unknown mask m_1.

second order correlation analysis is performed using the absolute difference combining function, introduced in Definition 3.3.2.

7.6.1 Simulation

The success rate and the resistance to noise are estimated by means of simulations. Moreover, an investigation on the influence of the number of bins for observation is made. The four different higher order generalizations of mutual information, as introduced in Section 5.4.2 are compared to cumulant-based second order MIA and second order correlation power analysis. Attacks using identity predictions are compared with attacks using Hamming weight predictions.

Success Rate

The success rate for different number of traces is determined on a set of simulated power traces with signal to noise ratio of one. Figure 7.29 shows the success rate for different attacks, for Hamming weight predictions (left) and identity predictions (right). Five bins for observation are used in histograms.

Figure 7.29 shows some remarkable properties. First of all, correlation analysis does not compromise the full DES cipher key in an identity prediction attack. Apparently, the absolute difference combining function is no longer appropriate as value to correlate predictions with. That still part of the key can be compromised, can depend on the properties of the 8 different DES S-boxes; the combining function is more suitable to one box input and output than to another one. Furthermore, cumulant MIA performs better in an identity model, as compared to a Hamming weight prediction model. Whereas the SNR remains the same, benefit is taken from better more distinction in prediction.

The order in which the higher order mutual information generalizations fully compromise the key is the similar to the order shown in AES simulations from Section 7.5. In both figures the similarity in performance of generalized mutual information and total correlation is reflected. Theoretically, these two methods should score equally, see Equation 5.12. Using identity predictions

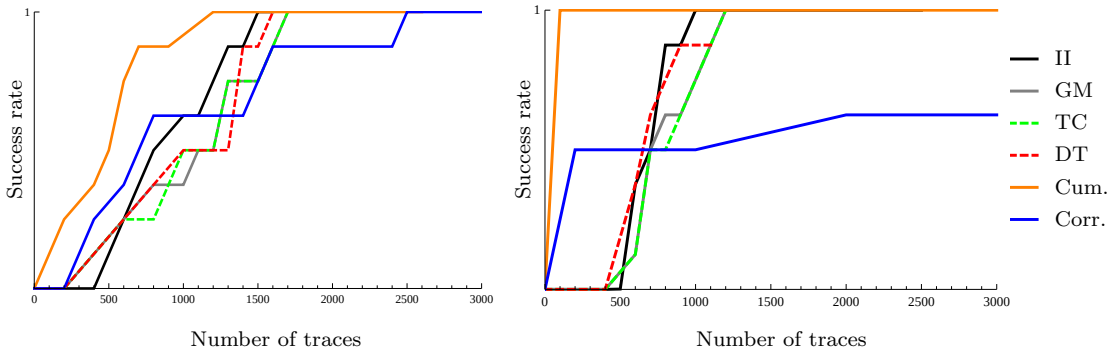


Figure 7.29: Success rate of a second order *Hamming weight* attack (left) and *identity* attack (right) on a masked DES simulation.

in histograms, performance increases and mutual difference among the methods decrease. Most striking, however, is the incapability of absolute-difference combined correlation attacks.

Number of Bins for Observation

The number of traces required for a complete successful attack on the DES cipher key is determined for different numbers of bins for observation. In Figure 7.30, results on a trace set with signal to noise ratio of one are given. Five bins for observation are used in histogram attacks.

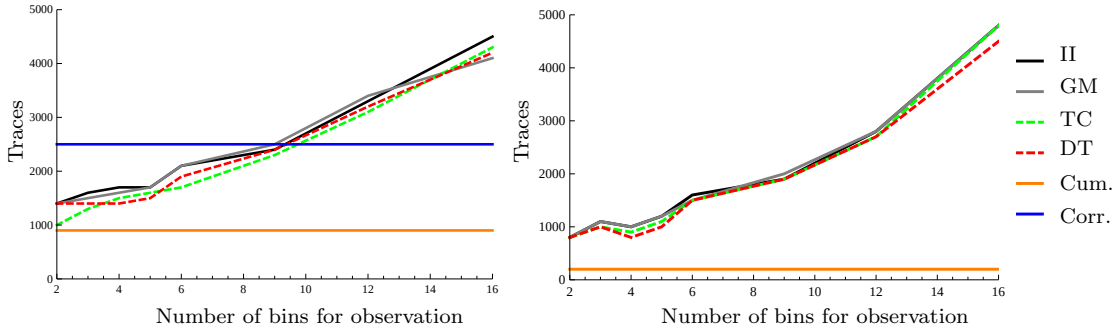


Figure 7.30: Influence of number of bins for observation on number of traces needed for a complete successful second order *Hamming weight* attack (left) and *identity* attack (right) on a masked DES simulation.

Similar to the results presented in Figure 7.29, cumulant MIA benefits the most from an identity prediction model as compared to a Hamming weight model. Since correlation attacks are not fully successful, no blue line appears in Figure 7.29 (right).

The four histogram MIA generalizations of mutual information show similar performance in both Hamming weight and identity prediction models. Moreover, the four methods do not show large mutual differences in performance, indicating no absolute prevalence for any of the methods from this figure. Again, selecting four to five bins for observation yields decent scores for all methods. Especially in Figure 7.29 (right), one can observe the similarity in performance of generalized mutual information and total correlation. Although selecting two bins for observation seems to be a good choice on these simulations, this choice is not considered adequate in practice, since there two bins may not provide enough distinguishing properties.

Resistance to Noise

Noise resistance on second order attacks is investigated on a set of 1000 traces. Figure 7.31 (left) shows the noise resistance using Hamming weight prediction models, Figure 7.31 (right) indicates noise resistance using identity prediction models. Note that the variance of the Hamming weight of uniformly random 4-bit strings equals 1, whereas the identity of a uniformly random string has variance 21.25, hence a standard deviation of $\sqrt{21,25} \approx 4.61$. Therefore, the horizontal axis in Figure 7.31 (right) is scaled by a factor of 4.61 in comparison with the chart on the left. Five bins for observation are used for histograms.

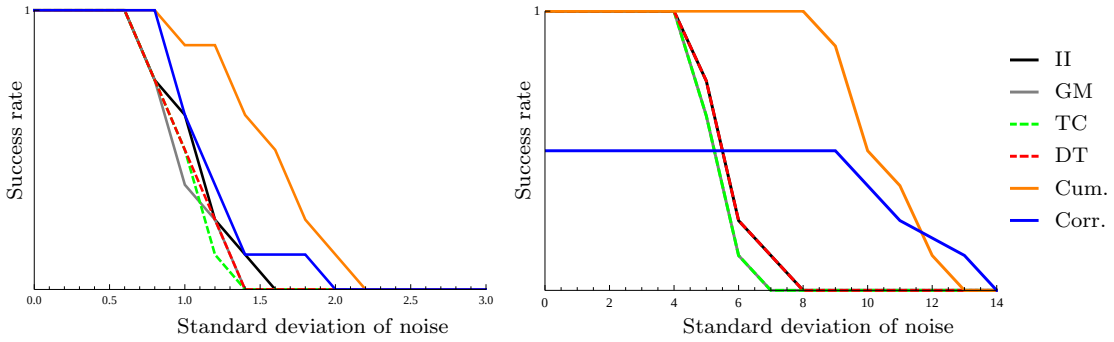


Figure 7.31: Noise resistance of a second order *Hamming weight* attack (left) and *identity* attack (right) on a masked DES simulation.

As can be seen in Figure 7.31 (left), cumulant MIA is most resistant to noise. The four different higher order generalizations are approximately equally resistant to simulated noise. Furthermore, correlation attacks are no long the most noise resistant in a Hamming weight scenario. However, Figure 7.31 (right) indicates that although only part of the key is compromised by correlating the absolute difference of two power consumption values to an identity prediction, it is the most noise resistant type. Interaction information and dual total correlation completely overlap, as do total correlation and generalized mutual information. Theoretically, total correlation and generalized mutual information should yield equal candidate rankings, which is reflected in these results.

7.6.2 Software Embedding

An implementation of masked DES is available on Riscure's Sample Type Card 2. The masking scheme on this card is described in Section 5.1.1. Table 7.9 contains information about the minimum number of traces required to compromise the 8 subkeys for different methods, using 5 bins for observation and Hamming distance predictions. Multiple higher order generalizations of mutual information are compared, as well as cumulant-MIA and correlation analysis using an absolute difference combining function. Empty cells in the table indicate that this subkey is not correctly compromised within the available 4000 traces.

In Table 7.9, information about total correlation and dual total correlations is missing. Using one of these two methods does not yield any correct guess. Moreover, using an identity prediction model, no score arises from any MIA method, no matter which higher order generalization is used. Correlating absolute difference with identity neither yields a correct subkey guess. However, absolute difference appears to be an adequate preprocessing function, yielding correlation with Hamming weight-based predictions.

Another observation arising from the table is that from all MIA attacks, cumulant MIA is the only one able to compromise all eight subkeys. Consequently, it might be seen as the only successful type of mutual information analysis on this trace set. This result is also supported by simulation results, where cumulant MIA scores best among the mutual information attacks. When moving to

Method Box:	1	2	3	4	5	6	7	8
Hist. II 5		4000	1000		2000	2000		4000
Hist. II 10		4000	2000		2500	3000		
Hist. GM 5	1500	2000	1000		1500	2500	2000	2500
Hist. GM 10		4000	2000		3000	2000	4000	
Cum.	2000	4000	3000	3000	3000	3000	3000	3000
Corr.	2000	500	2500	1000	2500	500	2000	2000

Table 7.9: Number of traces required for a successful DES round input-output *Hamming distance* second order attack on Sample Type Card 2.

histogram MIA, using more bins for observation in a histogram generally increases the number of traces required to compromise a certain S-box subkey. A slight difference can be observed between the performance of interaction information and generalized mutual information. Apparently, the single mutual information of prediction and one observation might have significant contribution.

When comparing the performance of mutual information attacks with correlation attacks, one can conclude that the latter one is most efficient, unlike in simulations. Apparently the absolute difference combining function is suitable to model leakage of two variables into one variable on this card. Moreover, the combined values show linear dependence with Hamming weight of predictions.

This card, Sample Type Card 2, is the same card as the one attacked in Section 7.2.1, but now with countermeasures enabled. However, the unprotected card already showed strong linear leakage. Similar to the first order results on this card, a second order correlation attack is sufficient to compromise the cipher key.

In Gierlichs et al. [GBPV10] a masked DES implementation is attacked. The authors compare generalized mutual information, interaction information and second order correlation attacks, using the normalized product combining function from Definition 3.3.1. As target, a low-noise card leaking Hamming weight of intermediate values is used. Using identity prediction models, correlation attacks are more efficient than MIA. Moreover, generalized MI scores better than interaction information in their results. In Hamming weight attacks, the methods do not differ much in score.

Furthermore, a masked DES implementation is examined by Batina et al. [BGP+11]. Besides results that match those presented by Gierlichs et al. [GBPV10] on a low-noise embedding, a high-noise scenario is analyzed. Here, correlation analysis is indicated as be the most noise-resistant. Moreover, interaction information shows to be more noise resistant than generalized mutual information. All these results match the conclusions drawn in this section.

7.7 Consumption of Time and Memory

Some results on the time consumption of the Mutual Information Analysis Module are given here by means of typical examples. Furthermore, data about the memory consumption of histogram MIA is presented.

The high security DES embedding attacked in Section 7.3.2 is taken as example to assess time consumption. Generally the number of traces is much greater than the number of time points. A successful histogram attack requires 10 minutes on 450000 traces, two time points per trace. A cumulant MIA of the same number of traces costs 15 minutes; a correlation attack on this trace set requires only 35 seconds of time (thus, histogram MIA is as much as 17 times slower). Histograms of the successful identity attack require 0.52 megabytes of space per sample.

Second order histogram analysis is much more time and memory consuming. Histograms are three-dimensional and the number time combinations might be significantly larger than the number of samples examined. For example, to analyze Sample Type Card 3 in Section 7.5, 1336 sample combinations are examined. When selecting five bins for observation and 9 for (Hamming weight) prediction, a total of 2,8 gigabytes is required to store all histograms, i.e., 2.14 megabytes per

sample. To analyze this set of 8000 traces and to write the histograms to disk, approximately 140 minutes of time are required. Reading the histogram for further analysis costs only 12 minutes, a great improvement. A second order correlation attack on this trace set costs 39 minutes of calculation time, so second order MIA is about 3.5 times slower here. However, a cumulant attack requires 782 minutes, which is way too much.

Overall, a mutual information attack does require significantly more time than a correlation attack. Evidently, this is due to the intensive process of density estimation. For first order analysis, mutual information using histograms is estimated about 20 times slower than correlation coefficients. In second order analysis, this ratio reduces to around 4 for full attacks. One advantage is that a second order analysis becomes more efficient when multiple scoring models are compared: an analysis on the already stored histograms consumes only a fraction of the time spent to create these histograms.

Cumulant MIA generally performs slower than histogram MIA; only when the number of time points is much greater than the number of traces to analyze cumulant MIA outperforms histogram MIA. However, this does not happen often at attacks on contemporary embeddings.

7.8 Overview

In this section, the results presented in this chapter are summarized and evaluated on multiple aspects.

Table 7.10 shows the 'winners' of the attacks on real implementations presented in this chapter: the method yielding a complete and correct cipher key within the smallest number of traces. Table 7.11 indicates the 'winners' on simulated trace sets.

Correlation analysis is abbreviated by C; histogram mutual information analysis is abbreviated by MH and cumulant MIA is abbreviated by MC. The number of bins for observation is appended, if applicable. The column efficiency (eff.) indicates the ratio in number of traces required for success between the best and second best method. Hamming weight-based prediction models are distinguished from identity-based predictions, the Sample Type Card identifier (st$x$) is included for completeness, as well as the type of information leakage (Hamming weight/distance or bitwise/identity). Evidently, in simulations, the prediction method accords to the information that is simulated to leak via power consumption.

		Real embedding HW			Real embedding ID		
Section and attack	Leaking	1st	2nd	eff.	1st	2nd	eff.
7.1 AES S-box output, st3	HW	C	MH5	1:2			
7.2 DES S-box output, st2	HW	C	MH5	1:3	C	MC	1:2
7.3.2 DES rd. output, st9a	Bit		–		MH5	–	
7.3.3 DES rd. output, st9b	Bit	MH6	–		MH6	–	
7.4.1 DES rd. diff., st8a	HD	C	MC,MH5	1:2		–	
7.4.2 DES rd. diff., st8b	HD	MH10, C	MC	2:3		–	
7.5 AES masked S-box, st3	HW	MH5	C	1:7			
7.6 DES masked rd., st2	HW	C	MC	2:3		–	

Table 7.10: Overview of performance of mutual information analysis versus correlation analysis on practical embeddings, including efficiency ratio: the ratio in number of traces needed for the best, resp. second best attack.

Estimation type. Table 7.10 shows that if mutual information analysis appears to be the most efficient attack on a practical embedding, histogram MIA dominates cumulant MIA. In contrast Table 7.11 shows that cumulant MIA dominates histogram MIA in simulations. This might be due

	Simulation HW			Simulation ID		
Section and attack	1st	2nd	eff.	1st	2nd	eff.
7.1 AES S-box output	C	MC	5:6			
7.3 DES rd. output	MC	C	5:6	MC	C	10:11
7.5 AES masked S-box	C	MC	1:2			
7.6 DES masked rd.	MC	MH4	2:3	MC	MH4	1:3

Table 7.11: Overview of performance of mutual information analysis versus correlation analysis on simulations, including efficiency ratio: the ratio in number of traces needed for the best, resp. second best attack.

to the fact that assumptions on which cumulant MIA is based (e.g., near-Gaussian distributions) are fulfilled in simulations, but may not hold in practice. Therefore, although cumulant MIA has theoretical advantages, histogram MIA is preferred on practical embeddings for the sake of generality. Moreover, cumulant MIA is more time-consuming than histogram MIA.

Simulations versus real-world embeddings. Generally, both tables show that power consumption of practical embeddings can not easily be simulated: attacks on simulations do not always show equivalent results to attacks on practical implementations. This occurs especially when countermeasures are applied; whereas correlation attacks are more efficient on masked AES simulations, mutual information analysis is more successful on a real embedding. The converse happens for a masked DES implementation. Evidently, profiling information leakage is the core challenge is side channel analysis. Since information leakage is not always easily predictable, simulation of information leakage may not always be representative. Hence, adequately simulating power consumption to compare mutual information analysis with CPA is a challenge on its own.

Gaussian assumption. The performance of cumulant MIA on identity-predictions is somewhat peculiar. Even though the Gaussian assumption is not fulfilled, cumulant MIA performs reasonably good, especially in simulations. Similar is the performance of correlation attacks on both identity predictions and Hamming weight predictions; in some cases power consumption seems to be correlated with both types of prediction (e.g., Sample Type Card 2 in Section 7.2.1); but, this does not happen very often (Hamming weight and identity of 4-bit values are correlated with $\rho = 0.81$). The success of cumulant MIA with identity-predictions can not fully be explained. Besides that, power consumption does not always show a near-Gaussian behavior, see for example Section 7.3.2.

Software or hardware. Mutual information analysis is shown to be especially helpful in case leakage seems to have a nonlinear relation with (Hamming weight of) predicted values. In general, nonlinear leakage is more likely to occur in hardware implementations, as compared to software embeddings. This is mostly due to physical properties of the embedding. Therefore, one can conclude that mutual information analysis is especially interesting to use on hardware embeddings, like Sample Type Card 9 in Section 7.3.

Number of bins. Concerning the number of bins for observation, one can conclude that generally taking more bins for observation than for prediction does not improve performance. Moreover, especially when power consumption follows a near-Gaussian distribution, four to five bins are sufficient to adequately model the densities. That there is no difference in targeted intermediate value (8-bit AES or 4-bit DES), might be clarified by the fact that all Sample Type Cards are equipped with an 8-bit CPU. Hence, power consumption is caused by processing 8-bit values, where four bits constitute to algorithmic noise in case of DES. In an identity attack on DES, 16 bins can yield a good result too, since the histogram is a square: a one-to-one mapping between predictions and observations is theoretically possible.

Unfortunately, no real masked DES embedding exploiting leakage unrelated to Hamming weight has been available during this project. Analysis of a card with these properties would be especially useful in order to further assess the benefits of identity-based prediction models in second order MIA. However, already in first order analysis these methods have proven their value.

Chapter 8

Conclusions

This chapter provides the reader with an overview of the main achievements from this master's thesis project. Furthermore, remarks and recommendations for usage of the Mutual Information Analysis Module are presented. Finally, directions for further research are proposed.

8.1 Main Achievements

At the beginning of this research project, several goals have been set, as indicated in Section 1.2. In this section, the goals are evaluated and achievements for each goal are presented.

- Mutual information analysis has extensively been described, for both first order attacks and second order attacks. Two mutual information estimation techniques are thoroughly described and assessed on adequacy in side channel analysis context: histograms and cumulants. Bin decision in histograms is assessed, as well as the Gaussian assumption in cumulants. For the sake of generality, histogram-based MIA is preferred over cumulant-based MIA.

- Mutual information analysis is compared to correlation power analysis both theoretically and practically. Scenarios in which correlation analysis would be preferred over MIA are sketched, and vice versa. The recommendations for parameter decisions in a mutual information analysis are summarized in the next section.

- A thorough investigation into the performance of mutual information analysis is carried out, using a specially developed Java-module. Statements made by Batina et al. [BGP+11] concerning the preference of correlation-based attacks over mutual information analysis on implementations leaking nearly linear information are confirmed in the results presented in Chapter 7: CPA is more efficient than MIA when power consumption and predictions are linearly related.

  Moreover, a high-security DES embedding (Section 7.3) and a masked AES embedding (Section 7.5) show resistance to correlation analysis, but not to mutual information analysis. This is because a linear prediction model does not hold in practice, resp., the selected preprocessing function strongly reduces information from paired observations. This proves mutual information analysis to be a valuable contribution to the field of side channel analysis.

  In particular, mutual information analysis is shown to be of significant interest for Riscure BV. Rather than a prototype, a Mutual Information Analysis Module is delivered ready for full usage within Riscure's side-channel and fault analysis tool Inspector.

Publication
The results of this project contribute to existing literature in multiple ways. First of all, attacks on modern high-level cryptographic implementations are provided (e.g., in Section 7.3). Furthermore, results of MIA on a masked AES S-box are given (in Section 7.5) and two new higher order distinguishers are evaluated. Practical results on cumulant-based MIA are provided for both first order and second order analysis. All of this is, to the best of our knowledge, not yet treated in literature. In general, MIA is compared to CPA from a practical point of view, and the applicability of MIA in a professional side channel testing environment (such as Riscure) is assessed.

These results are believed to be of scientific value; therefore, a paper presenting the contributions will be written shortly after the end of this master's thesis project. The paper will be submitted for publication at COSADE 2012.

8.2 Recommendations

First order analysis. Mutual information analysis has proven its value in several scenarios. As argued in Section 4.5.2, correlation analysis is preferred over mutual information analysis when dependence between predictions and power consumption is nearly linear. However, when power consumption shows nonlinear relations with dependence, MIA can be more efficient. Nonlinear dependencies are more likely to occur in unprotected hardware implementations, than in unprotected software implementations, due to internal logic. Therefore, mutual information analysis is particularly advised for attacks on hardware implementations. However, since the type of information leakage might not be known in advance, and since correlation analysis is significantly faster than mutual information analysis, it is advised to first perform a correlation attack. If this attack does not yield the desired result, a mutual information attack is an excellent alternative. Normalized mutual information is not recommended as distinguisher, as argued in Section 6.3.3.

Second order analysis. Protected implementations are of special interest for mutual information attacks. Correlation-based second order attacks strongly depend on adequacy of a preprocessing function. On the one hand, in case the preprocessing function seems to be adequate, as in the attack presented in Section 7.6.2, correlation analysis is more efficient than a mutual information attack. On the other hand, if power consumption only has a weakly linear relation with preprocessed observations, mutual information attacks show more efficiency. This is the case in Section 7.5.2. Information on this dependence might however not be known to the attacker. Most of the contemporary chips are indeed protected on hardware level or software level. Chip manufacturers apply countermeasures to their products to prevent side channel analysis: to disconnect power consumption from values processed by a device. Although this does not seem completely possible in practice, at least a linear dependence is likely to vanish. Therefore, MIA is expected to be more efficient than correlation analysis. However, since second order correlation analysis is significantly faster than second order mutual information analysis, it is advised to first perform a correlation attack, using a preprocessing function. A mutual information attack is again an excellent alternative in case correlation attacks do not yield adequate results.

Density estimation type. The histogram-based density estimation is known to be nonparametric, in the sense that no underlying distribution is assumed. Mutual information using cumulants, in contrast, is based on the assumption of near-Gaussian probability distributions. The high-security hardware implementation analyzed in Section 7.3.2, for example, shows that this assumption does not hold in every practical embedding. Moreover, MIA using cumulants for estimation generally consumes more time than a histogram-based MIA. Therefore, mutual information using histograms is recommended as most adequate MIA method to use on practical implementations.

Number of bins. The influence of the number of bins for observation is thoroughly investigated in this thesis. On the majority of the practical embeddings, MIA performs most efficiently when around 5 bins are used for observation. Especially when Hamming weight-based predictions are made, this tends to be the best choice. When performing an attack on high-security DES using

identity-based predictions, 16 bins might also be a good choice, as shown in Sections 7.3.2 and 7.3.3.

Higher order generalization. In a second order mutual information attack, an attacker can select one of the four generalizations as distinguisher. Generalized mutual information is advised to use as first distinguisher. This distinguisher generally shows best performance on practical embeddings, since it can benefit from implementations where masks are not generated uniformly at random. However, it is advised to store histograms and perform another attack on these histograms, using the interaction information distinguisher. This method might perform better than generalized mutual information in noisy environments (and on simulated trace sets). Moreover, analyzing already stored histograms consumes significantly less time than performing a new histogram attack on the same trace set. Therefore, performing another MIA only requires a fraction of the time required for the first analysis, but provides an attacker with results of a full analysis.

Interesting time points. A general challenge in differential power analysis concerns selecting the time points where information leakage is expected to appear. As indicated in this thesis, limiting the search space as much as possible is of crucial interest; not only might noise from 'uninteresting' time points yield ghost peaks, computation time increases when the number of time points to be analyzed increases. Therefore, an attacker must carefully select the points in time on which MIA is performed. One possible direction to do this is via Principal Component Analysis, as investigated by Hogenboom [Hog10] at Riscure. Unfortunately, MIA does not intrinsically provide an attacker with information about the interesting parts in a power trace.

8.3 Further Research

The research in this thesis has provided results on mutual information analysis, both on simulations and practical embeddings. Directions for further research are numerous. A brief overview is given here.

- Two models for mutual information estimation are analyzed in this thesis: histograms and cumulants. Many other methods exist (see Section 4.4.3) and all have their respective drawbacks and benefits. An investigation on the suitability of other density estimation models for mutual information-based side channel analysis is interesting, especially on real-world high security embeddings instead of on simulated traces.
- Histograms are assumed to have equally sized bins. Relaxing this assumption might reveal more information. For example, using many bins in dense parts can provide an attacker with more detailed information about likely outcomes. However, this does require a priori knowledge of underlying distributions. If knowledge is not present yet, estimation reduces to a bootstrap problem. One can inquire into MIA using histograms with variable binsizes, based on assumptions on power consumption distribution. A Gaussian distribution, which is assumed in cumulant-based MIA, might be a good first choice to adapt binsizes to.
- In simulations, linear relations between (Hamming weight of) intermediate values and power consumption are modeled. Although this corresponds to reality often, some practical implementations (e.g., in Section 7.3) do not resemble the simulations. Research into other simulation models can provide more accurate simulations, on which MIA and CPA can further be compared. For example, a nonlinear leakage model can be used to further inquire into the benefits of identity-based predictions.
- No practical masked DES software embedding showing nonlinear leakage has been available during the execution of this project. A masked hardware embedding is analyzed in Section 7.3.2, where a first order MIA scores already significantly better than any correlation attack. However, it is interesting to test second order mutual information analysis using identity-based prediction models on masked software implementations, so mutual information analysis can further prove its generality under these circumstances.

Appendix A

Cryptographic Algorithms

In this appendix, the Data Encryptions Standard (DES) and the Advanced Encryption Standard (AES) are explained. These are two symmetric cryptographic algorithms widely used for security on smart cards. The mutual information attacks on intermediate values described in this thesis all concern implementations of DES or AES; therefore, the overview is limited to these two algorithms.

A.1 Data Encryption Standard (DES)

The Data Encryption Standard was an early cryptographic standard adopted by the United States National Institute of Standards and Technology (U.S. NIST) in 1977 [FIP77]. It was reaffirmed multiple times until its withdrawal in 2005. By that time, DES was no longer resistant to cryptographic attacks due to the increase of computational power over time. It was replaces by the Advanced Encryption Standard (AES), which is described in the next section.

DES is performed on plaintexts of 64 bits, using a key of 64 bits. However, the entropy of the key equals 56 bits; eight bits are used for parity check. To improve security of DES-based implementations, a variant consisting of three consecutive DES executions, called triple-DES, is used. This scheme is not in the scope of this thesis. In this section, the DES algorithm is explained, as well as the key schedule algorithm. The reader is referred to FIPS publication 46 [FIP77] for a more detailed description, including all permutations.

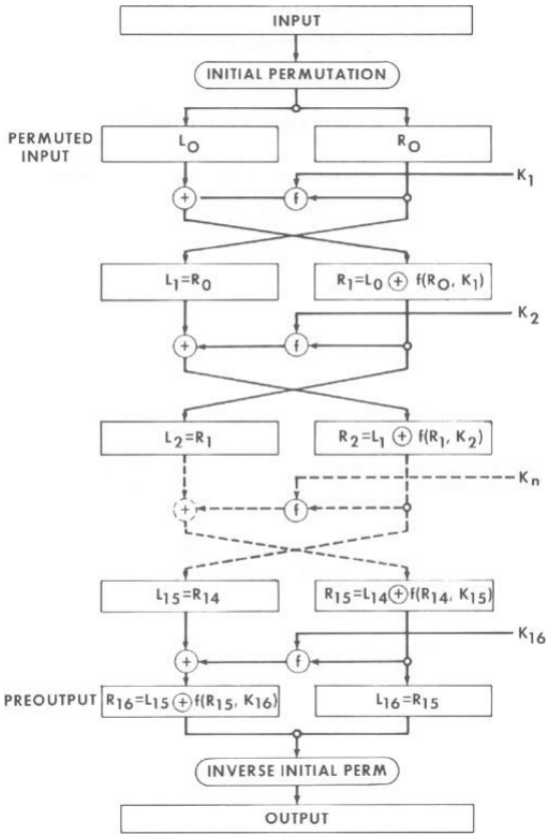


Figure A.1: Scheme of DES algorithm.[FIP77]

A.1.1 Algorithm

DES is a block cipher operating in rounds, the round input and output are two blocks of 32 bits. DES is a symmetric encryption system,

which means that the key for encryption is the
same as the key for decryption. Furthermore, a DES decryption involves the same steps as a DES encryption, only the round keys are used in reversed order.

Figure A.1 depicts the encryption scheme. Here, the input is a 64-bit value, which is permuted and halved to obtain the left half (L_0) and right half (R_0) of the first round input. The right half undergoes a nonlinear transformation f, involving a 48-bits round key K_1, before it is XOR'ed with the left half. Then, left half and right half are swapped, this is the input for the next round.

This procedure is repeated for 16 rounds. Then, the initial permutation is applied inversely to the concatenated left and right half, yielding the algorithm's output.

The nonlinear round function f operates as depicted in Figure A.2. First, the 32-bit input is expanded to 48 bits. Then, a 48-bit round key is XOR'ed with the expanded input. The results is split into eight parts of six bits. Each part is then substituted by four bits: the output of a nonlinear substitution box (S-box). All eight S-boxes are distinct. The eight parts of four bits are again concatenated and permuted, yielding the 32-bit output.

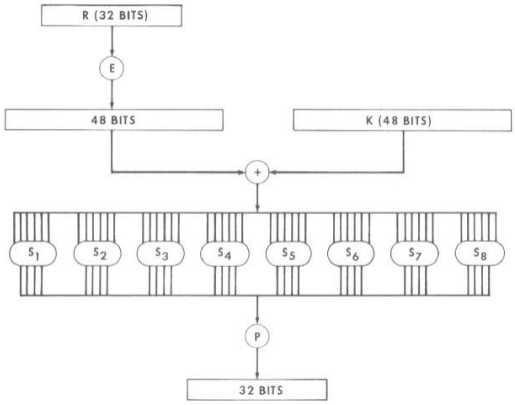


Figure A.2: Scheme of DES round function.[FIP77]

A.1.2 Key Schedule

A DES cipher key consists of 56 bits of information and eight redundant parity check bits. The key schedule to generate round keys from the original cipher key is depicted in Figure A.3.

As can be seen, the cipher key is permuted before being split in to equally sized parts. This permutation only involves the 56 relevant bits of the cipher key, both C_0 and D_0 are henceforth 28-bit values. The values C_i and D_i ($i = 1, \ldots, 16$) can be obtained from preceding C_{i-1} and D_{i-1}, using a number of left shifts. A left shift is a rotation of the 28 bits, in such a way that the 28 bits after one left shift are equal to those previously in positions $2, 3, \ldots, 28, 1$. The number of left shifts is not equal for each round; in round 1, 2, 9 and 16 one left shift is performed, in all others two.

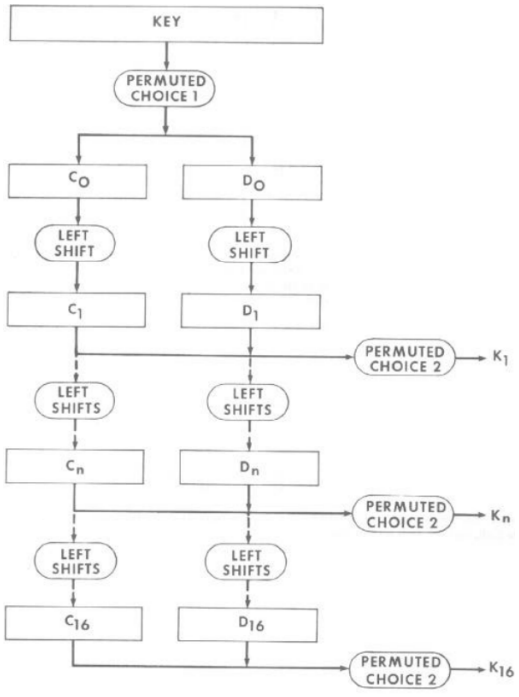


Figure A.3: Scheme of DES key schedule.[FIP77]

The two round halfs C_i and D_i are concatenated and permuted, to give rise to a round key K_i ($i = 1, \ldots, 16$).

A.2 Advanced Encryption Standard (AES)

The Advanced Encryption Standard is a cryptographic standard adopted by the United States National Institute of Standards and Technology (U.S. NIST) in 2001. The NIST announced an election process for this new standard in 1997, for which submission of a candidate algorithm was open to anyone. A limited number of finalists was then judged by the cryptographic community in terms of security, performance and implementation feasibility. Finally, the winning algorithm was announced to be Rijndael, submitted by Daemen and Rijmen [DR02]. In this section, the AES algorithm and its properties are described, based on the official NIST description [FIP01].

Nowadays, AES is the most commonly used symmetric key algorithm. It is no 'security by obscurity' algorithm; everything about the algorithm is publicly known. AES supports cryptographic keys of 128, 192 and 256 bits. Plaintext data is encrypted in blocks of 128 bits each. The description of AES wil be limited to AES-128, i.e. with keys of 128 bits only. For brevity, 'AES' is from now on used as call sign for AES-128.

A.2.1 Algorithm

AES [FIP01] encrypts 128-bit data blocks under a 128-bit key. Both the data block and the key are represented by a 4×4 data block, where each entry represents one byte. The data block is called the *state*. A byte is denoted in hexadecimal notation, ranging in `00-ff` in stead of 0-255. A word is an array of 4 bytes, i.e. 32 bits. Four words are 128 bits.

In encryption, a total of 10 transformation rounds are applied to the state. For each round, a round key generated by the key schedule algorithm. Decryption works similarly, but in reversed order. Each round consists of four steps: `AddRoundKey()`, `SubBytes()`, `ShiftRows()` and `MixColumns()`. In the first 9 rounds, these are applied successively to the state. In the last round, `MixColumns()` is omitted; only the first three steps are performed.

Algorithm A.1 gives AES in pseudo-code. The different round routines are explained in Section A.2.2. The key schedule algorithm is explained in Section A.2.3, this algorithm produces `w`, a concatenation of 11 round keys.

A.2.2 Round Transformation

`AddRoundKey()`

The state is XOR'ed with the round key. In the first round, the round key is equal to the original cipher key. The length of each round key is equal to the length of the state, i.e. 128 bits.

`SubBytes()`

`SubBytes()` performs a deterministic byte substitution to each byte (i.e. entry) in the state. This substitution function S is called the S-box. Each byte is represented by an element of $\mathbb{F}_{256} = GF(2^8)$, the finite field with 256 elements. AES uses

$$\mathbb{F}_{256} = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1),$$

where $\mathbb{F}_2[X] = \mathbb{Z}[X]/2\mathbb{Z}$, the ring of polynomials with coefficients in $\{0, 1\}$. All arithmetic is performed in $\mathbb{F}_{256}$, i.e. addition modulo 2 and reduction modulo $x^8 + x^4 + x^3 + x + 1$.

On a given input $x \in GF(2^8)$, the S-box outputs $S(x) = Ax^{-1} + b$. The matrix A and vector b are fixed constants in the algorithm:

Algorithm A.1 AES Algorithm

Variables: `byte[16] in, byte[16] out, word[44] w.`

1. `byte[4][4] state=in`
2. `AddRoundKey(state,w[0,3])`
3. `for (round=1,...,9):`
   (a) `SubBytes(state)`
   (b) `ShiftRows(state)`
   (c) `MixColumns(state)`
   (d) `AddRoundKey(state, w[4*round, 4*round+3])`
4. `// round 10`
   (a) `SubBytes(state)`
   (b) `ShiftRows(state)`
   (c) `AddRoundKey(state,w[40,43])`
5. `out=state`

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Hence, the S-box function includes first determining the field inverse of the element x. The output value is converted to hexadecimal format and stored in the output state. After every byte has been substituted, `SubBytes()` returns the output state. The `SubBytes()` operation was intended to create a rather complex connection between ciphertext and the key; a property Shannon called confusion in his 1949 paper [Sha49]. Changing only one bit of the key should completely change the ciphertext, in an unpredictable way.

Since the S-box is a deterministic function and finite field inversion and matrix multiplication are computationally intensive, the output of the S-box function can be stored as a lookup table T in the embedding. Moreover, in hardware implementations this table can be implemented using logical components.

`ShiftRows()`

The rows $i = 0, 1, 2, 3$ (i.e. all) are shifted i positions to the left; the i leftmost entries are placed on the right.

```
00  44  88  cc                    00  44  88  cc
11  55  99  dd   ---ShiftRows()-->  55  99  dd  11
22  66  aa  ee                    aa  ee  22  66
33  77  bb  ff                    ff  33  77  bb
```

`MixColumns()`

Performs a matrix multiplication with the state columns. The columns $(b_{0,i}, b_{1,i}, b_{2,i}, b_{3,i})^T$ are viewed as degree-3 polynomial over GF_{2^8}: $b_{0,i} + b_{1,i}x + b_{2,i}x^2 + b_{3,i}x^3$. These columns are

multiplied by $\texttt{03}x^3+\texttt{01}x^2+\texttt{01}x+\texttt{02}$ and reduced mod x^4+1. The polynomial $3x^3+x^2+x+2$ is invertible mod x^4+1. The result $c_{0,i}+c_{1,i}x+c_{2,i}x^2+c_{3,i}x^3$ is translated as the column $(b_{0,i}, b_{1,i}, b_{2,i}, c_{3,i})^T$ in the resulting state. Multiplication by this polynomial and reducing it mod x^4+1 is the same as multiplying by the matrix

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}.$$

Both `ShiftRows()` and `MixColumns()` are linear transformations. These are intended to disconnect the statistical properties of plaintext and ciphertext. Changing one bit in the plaintext should create an avalanche effect, i.e. should lead to a major change in the ciphertext. This property was named diffusion by Shannon [Sha49] and was, together with confusion, indicated to be one of the most important properties of a sound cryptographic system.

A.2.3 Key Schedule

The original cipher key k is used to generate a key schedule of 44 words $\texttt{w}_i$, stored in the word `w[44]`. Each word `w` consists of 4 bytes. The 128 bit cipher key k is placed in the first four words. In encryption round i, words $4i$ up to $4i+3$ are used ($i = 0, \ldots, 10$).

Algorithm A.2 AES Key Expansion Algorithm

Variables: `byte[16] key, word[44] w.`

```
for (i=0,...,3):
  w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
for (i=4,...,43):
  word temp = w[i-1]
  if (i mod 4 == 0):
    temp = SubWord(RotWord(temp)) XOR Rcon(i/4)
  w[i] = w[i-4] XOR temp
```

The `SubWord()` function applies an S-box function to each of the four input bytes in the input word. The `RotWord()` function with input word `(t1,t2,t3,t4)` performs a cyclic permutation and outputs `(t2,t3,t4,t1)`. This means that if the word `temp = (t1,t2,t3,t4)`, then `SubWord(RotWord(temp))` outputs the word (S(t2),S(t3),S(t4),S(t1)). The `Rcon[]` is a constant word: `Rcon[i]` =$(\texttt{02}^{i-1},0,0,0)$, the exponentiation of 2 in $GF(2^8)$. It has an index starting at one: $i = 1, .., 10$.

In words explained, the key expansion starts with placing the cipher key in the first four words. Every following word $w[i]$, where $i \neq 0 \mod 4$ is equal to the XOR of $w[i-1]$ and $w[i-4]$. If $i = 0 \mod 4$, $w[i-1]$ is transformed by `RotWord()` and `SubWord()` before being XOR'ed with $w[i-4]$. Then, `Rcon[i]` is XOR'ed with this intermediate result, yielding $w[i]$.

List of Tables

List of Figures

Nomenclature

α_t	Minimum of observations at time t, page 68
$b_t(i)$	Lower bound of bin i at time t, page 68
$\mathcal{D}$	Plaintext space, page 30
$\mathbf{D}$	Plaintext as random variable, page 30
d	Plaintext, page 30
E	Expectation, page 11
F	Frequency matrix, page 34
$f_{\mathbf{H},\mathbf{O}}$	Joint density function of $\mathbf{H}$ and $\mathbf{O}$, page 8
γ_1	Skewness, page 13
γ_2	Kurtosis, page 13
$\mathcal{H}$	Leakage prediction space, page 30
$\mathbf{H}$	Predicted power consumption as random variable, page 30
$\tilde{\mathbf{H}}$	Normalized prediction random variable, page 40
h	Hypothetical power consumption (prediction), page 30
$H(\mathbf{X})$	Entropy of X, page 3
$H(\mathbf{X}, \mathbf{Y})$	Joint entropy of X and Y, page 4
$HD(v_1, v_2)$	Hamming distance between values v_1 and v_2, page 21
$HW(v)$	Hamming weight of value v, page 21
$I(\mathbf{X}; \mathbf{Y})$	Mutual information of $\mathbf{X}$ and $\mathbf{Y}$, page 8
J	Joint density matrix, page 34
$\hat{k}$	Key guess, page 30
κ	Correct key, page 31
κ_i	i-th cumulant, page 14
$\mathcal{K}$	Key space, page 30
$\mathbf{K}$	Key as random variable, page 30

l	Number of time points, page 30
μ_i	i-th central moment, page 12
M	Mutual Information matrix, page 31
m	Mask, page 49
m_i	i-th central sample moment, page 12
m'_i	i-th raw sample moment, page 12
$m_{\mathbf{X}}$	Sample mean of a sample from $\mathbf{X}$, page 40
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2, page 7
n	Number of key guesses, or dimension, page 30
$\mathcal{O}$	Observation space, page 30
$\mathbf{O}$	Observation as random variable, page 30
$\tilde{\mathbf{O}}$	Normalized observation random variable, page 40
o	Observation, page 30
ϕ	Leakage prediction function, page 21
$P(\mathbf{X} = x)$	Probability of event $\mathbf{X} = x$, page 4
q	Number of measurements, page 30
$\rho(\mathbf{X}, \mathbf{Y})$	Correlation coefficient of $\mathbf{X}$ and $\mathbf{Y}$, page 11
$s_{\mathbf{X}}$	Sample standard deviation of a sample from $\mathbf{X}$, page 40
τ	Time point of leakage, page 31
t	Point in time, page 30
u	Number of bins in prediction model, page 33
v	Number of bins in observation, page 33
$\mathbf{X}$	Random variable, page 3
x	Intermediate value, page 19
x_m	Intermediate value x concealed by mask m, page 49
ω_t	Maximum of observations at time t, page 68
$\oplus$	Exclusive-OR operation, page 21

Bibliography

[BE92] Lee J. Bain and Max Engelhardt. *Introduction to Probability and Mathematical Statistics*. Classic Series. Duxbury, 1992.

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual Information Analysis: a Comprehensive Study. *Journal of Cryptology*, 2011.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[DR02] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer-Verlag, 2002.

[FGD+10] Florent Flament, Sylvain Guilley, Jean-Luc Danger, Moulay Aziz Elaabid, Houssem Maghrebi, and Laurent Sauvage. About Probability Density Function Estimation for Side Channel Analysis. In *COSADE 2010*, 2010.

[FIP77] FIPS. Specification for the Data Encryption Standard (DES). Federal Information Processing Standards Publication 46, 1977.

[FIP01] FIPS. Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001.

[GBPV10] Benedikt Gierlichs, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. Revisiting Higher-Order DPA Attacks: Multivariate Mutual Information Analysis. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 221–234. Springer-Verlag, 2010.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis – A Generic Side-Channel Distinguisher. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5141 of *Lecture Notes in Computer Science*, pages 426–442. Springer-Verlag, 2008.

[Han78] Te Sun Han. Nonnegative entropy measures of multivariate symmetric correlations. *Information and Control*, 36(2):133–156, 1978.

[Hog10] Jip Hogenboom. Principal component analysis and side-channel attacks. Master's thesis no. 634, RU Nijmegen, 2010.

[JB80] Carlos M. Jarque and Anil K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3):255–259, 1980.

[JB04] Aleks Jakulin and Ivan Bratko. Quantifying and visualizing attribute interactions: An approach based on entropy. *http://arxiv.org/abs/cs.AI/0308002 v3*, 308002:3, 2004.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. *Lecture Notes in Computer Science*, 1:388–397, 1999.

[KM90] John E. Kolassa and Peter McCullagh. Edgeworth series for lattice distributions. *Annals of Statistics*, 18(2):981–985, June 1990.

[Kol94] John E. Kolassa. Series Approximation Methods in Statistics. In *Series Approximation Methods in Statistics*, Lecture Notes in Statistics. Springer, 1994.

[LB10] Thanh-Ha Le and Mael Berthier. Mutual Information Analysis under the View of Higher-Order Statistics. In *Proceedings of the 5th international conference on Advances in Information and Computer Security*, IWSEC'10, pages 285–300. Springer-Verlag, 2010.

[McC87] Peter McCullagh. *Tensor methods in statistics*. Monographs on statistics and applied probability. Chapman and Hall, 1987.

[McG54] William McGill. Multivariate information transmission. *Psychometrika*, 19(2):97–116, 1954.

[MMPS09] Amir Moradi, Nima Mousavi, Christof Paar, and Mahmoud Salmasizadeh. A Comparative Study of Mutual Information Analysis under a Gaussian Assumption. In Heung Youl Youm and Moti Yung, editors, *WISA 2009*, volume 5932 of *Lecture Notes in Computer Science*, pages 193–205. Springer, 2009.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[Péb08] Philippe Pébay. Formulas for Robust, One-Pass Parallel Computation of Covariances and Arbitrary-Order Statistical Moments. *Sandia National Laboratories*, SAND2008-6212, 2008.

[PR09] Emmanuel Prouff and Matthieu Rivain. Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In Pierre-Alain Fouque Michel Abdalla, David Pointcheval and Damien Vergnaud, editors, *Applied Cryptography and Network Security – ACNS 2009*, volume 5536 of *Lecture Notes in Computer Science*, pages 499–518. Springer-Verlag, 2009.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, pages 799–811, 2009.

[Sco79] David W. Scott. On optimal and data-based histograms. *Biometrika*, 66:605–610, 1979.

[Sha48] Claude E. Shannon. A Mathematical Theory of Communication. *Bell Systems Techn. Journal*, 27:623–656, 1948.

[Sha49] Claude E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[Stu26] Herbert A. Sturges. The choice of a class interval. *Journal of the American Statistical Association*, pages 65–66, 1926.

[SVC09] François-Xavier Standaert and Nicolas Veyrat-Charvillon. Mutual Information Analysis: How, When and Why? In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.

[VCS11] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Generic side-channel distinguishers: Improvements and limitations. Cryptology ePrint Archive, Report 2011/149, 2011.

[Ven10] Alexandre Venelli. Efficient Entropy Estimation for Mutual Information Analysis Using B-Splines. In *WISTP*, volume 6033 of *Lecture Notes in Computer Science*, pages 17–30. Springer-Verlag, 2010.

[Wat60] Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of Research and Development*, 4(1):66 – 82, 1960.