THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à l'Ecole normale supérieure de Paris

Side-Channel and Fault Analysis in the Presence of Countermeasures: Tools, Theory and Practice

Ecole doctorale n° 386

SCIENCES MATHEMATIQUES DE PARIS CENTRE

Spécialité INFORMATIQUE

COMPOSITION DU JURY :

M. GUILLEY Sylvain
TELECOM-ParisTech, Rapporteur

M. GOUBIN Louis
Université of Versailles-St-Quentin-en-Yvelines, Rapporteur

M. VUILLEMIN Jean
École normale supérieure, Membre du jury

M. VAUDENAY Serge
École polytechnique fédérale de Lausanne, Membre du jury

M. KOCHER Paul
Rambus, Membre du jury

Mme. TRICHINA Elena
Rambus, Membre du jury

Soutenue par Roman **KORKIKIAN**
le 27 Octobre 2016

Dirigée par **David NACCACHE**

ENS PSL RESEARCH UNIVERSITY PARIS

Acknowledgements

At the very first, I want to thank my mother *Elena Korkikian* and father *Gevork Korkikian*. Even though they have their own struggles, they always put mine first. Without them and their tremendous support there is no way that I could have made it so far.

In full gratitude I would like to acknowledge *David Naccache* for his guidance as my thesis advisor during these years. You encouraged, inspired, and assisted me in the pursuit of this higher education degree.

I also thank my co-authors and my fellow labmates for the stimulating discussions *Eric Brier, Jean-Michel Cioranesco, Quentin Fortier, Diana Maimuț, Guilherme Ozari de Almeida, Sylvain Pelissier, Adrien Pommellet, Rodrigo Portella do Canto*, and *Jean Vuillemin*. It was a true pleasure working with you all.

I am most grateful to *Elena Trichina* for her encouragement and practical advice. I am also thankful to her for reading my reports, commenting on my views and helping me understand and enrich my ideas. She supported me since the very beginning.

I would like to pay tribute to my jury *Jean Vuillemin*, *Serge Vaudenay*, *Paul Kocher*, and *Elena Trichina* for agreeing to serve on this thesis committee. I express my particular gratitude to my thesis referees *Sylvain Guilley* and *Louis Goubin* for their availability and dedication. I am very honoured to have such a prestigious committee.

The research work presented in this thesis was supported by Altis Semiconductor and the association nationale de la recherche et de la technologie. Thank you for giving me the opportunity to start my own research.

I would like to extend my profound gratitude towards my current employer *NagraVision* and my team *Security Evaluations and Attacks*. Thank you for supporting me especially towards the end of this thesis, and for offering me the possibility to apply the knowledge I accumulated.

I want to thank my teachers and professors without whom I would not have been able to start my thesis. I am especially indebted to *Renaud Pacalet* who introduced me to cryptography and hardware security, to *Sergey Pashkov* for teaching me moving forward despite all difficulties, to *Igor Simakov* and *Valentin Kuprianov* for giving me a taste of science.

I want to express my sincere thanks to *Natacha Laniado* for helping me in difficult moments. Last but not least, I want to thank *Ekaterina Leonova* for supporting me throughout the development of this work.

A sacrifice to be real must cost, must hurt, must empty ourselves.

– Blessed Mother Teresa of Calcutta Roman Catholic Nun of Charity and Love

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Summary

Cryptology continues to co-evolve with state-or-the-art communication and computing technologies. Just as previous technological breakthroughs, such as the telegraph, radio, electro-mechanical devices, and personal computers, compelled cryptography to replace broken or weak ciphers, the advert of widespread embedded devices induces new cryptographic vulnerabilities. Material vulnerabilities appear at circuit-level, where a malicious user can measure or physically stress transistors' states. Physical observations can be correlated to the processed data by side-channel analysis. Malicious data modifications lead to fault attacks. The emergence of these low layer attacks arises an assumption that cryptography is necessary for private communication and secure processing, but it is not sufficient. As such, cryptographic algorithms must be protected from malicious analysis.

This chapter describes secure embedded design essentials. Section 1.1 presents a brief introduction to the history of cryptology, which highlights changes in cryptography after several technological breakthroughs. Section 1.2 describes terminology and cryptographic concepts used throughout the thesis. Section 1.3 provides the technical details of SPN and Feistel block ciphers. Section 1.4 briefly explains public key cryptography and Section1.5 describes digital signatures. All the described cryptographic algorithms are referenced in the following chapters.

1.1 A Brief Introduction to Secure Communication

Private communications and secure processing relies on *cryptography*, defined as the study of techniques for securing digital information, transactions, and distributed computations. The first evidence of cryptography can be traced back to the Ancient Times (about 3000-2000 B.C.), in Babylonia and the Old Kingdom of Egypt [Dar04]. *Cryptanalysis*, the study of breaking cryptographic systems and gaining access to the contents of ciphertexts, has co-evolved with cryptography. The history of encryption is the history of "the contest of wits" between cryptography and cryptanalysis - new ciphers being designed to replace old broken designs, and new cryptanalytic techniques being invented to crack the improved schemes.

Encryption, as much as any other algorithm, can be seen as a sequence of instructions. These instructions describe the computations that transform a *plaintext* (a clear data) to a *ciphertext* (a scrambled output). Any computation ultimately involves a computing device, for instance, a smart-card, a tablet, a mobile phone, a personal computer, etc. Hence, among other factors, the computing devices' capabilities are correlated to advances in *cryptology*, defined as the combined study of cryptography and cryptanalysis.

Looking back into the history of technology helps the understanding of cryptologic achievements. Before the invention of the telegraph in 1844, all ciphertexts were handed physically. Telegraph communications could be easily intercepted, so a need for secure communication over unprotected channels has appeared. At first, a Vigenère cipher was widely used [Sin11]. In 1863, Friedrich W. Kasiski [Kas63] discovered a solution to all periodic polyalphabetic ciphers, which until that time were considered unbreakable. Therefore, Vigenère ciphers had to be replaced.

Just as telegraph changed cryptography in 1844, radio changed cryptography in 1895. Now transmissions were open for anyone's inspection, and physical security was no longer possible. Until 1917, transmissions were encoded in Baudot code as for the use with teletypes [Mog08].[1] The American Telephone and Telegraph Company was very concerned with the ease of reading the Baudot code, so Gilbert S. Vernam [Ver58] developed an encryption machine that added the plaintext electronic pulses to a key to produce ciphertext pulses. Vernam's encryption machine was never widely used but the addition modulo-2 together with the use of the same keystream to encipher and decipher are the basis of modern cryptography.

The use of cryptographic machines dramatically changed the nature of cryptology. Cryptography became intimately related to machine design, and security personnel became involved in the protection of these machines. The basic systems remained the same, while encryption methods became reliable and electromechanical.

The next major advancement in electromechanical cryptography came with the invention of the rotor machine by Theo van Hengel and Rudolf Pieter Cornelis Spengler [MPM+96][2]. The rotor is a thick disk with two faces, each with 26 brass contacts separated by insulating material. Each contact on the input (plaintext) face is connected by a wire to a randomly chosen contact on the output (ciphertext) face. Each contact is assigned a specific letter. An electrical impulse applied to a contact on the input face will result in a different letter being an output of the ciphertext face. A single rotor thus implements a monoalphabetic substitution cipher. This rotor is set in a device that takes plaintext input from a typewriter keyboard, and sends the corresponding electrical impulse to the plaintext face. The ciphertext is generated by the rotor, and printed and/or transmitted.

German codes during the Second World War were predominantly based on the 'Enigma' machine [Sin11], which is an extension of the electromechanical rotor machine discussed above. Enigma defined a polyalphabetic substitution cipher, with a period before the repetition of the substitution alphabet that was much longer than any message, or set of messages, sent with the same key. Marian Rejewski could build the first brute-search electro-mechanical device that was dubbed the *bomba kryptologiczna* or *cryptologic bomb*. Rejewski has written [Rej82] about the device: "*The bomb method, invented in the autumn of 1938, consisted largely in the automation and acceleration of the process of reconstructing daily keys. Each cryptologic bomb (six were built in Warsaw for the Biuro Szyfrów Cipher Bureau before September 1939) essentially constituted an electrically powered aggregate of six Enigmas. It took the place of about one hundred workers and shortened the time for obtaining a key to about two hours.*"

[1] The symbol rate measurement unit, known as the *baud*, is derived from Baudot's name.

[2] Previously, the invention had been ascribed to four inventors working independently and at much the same time: Edward Hebern, Arvid Damm, Hugo Koch and Arthur Scherbius.

Shannon was one of the first modern cryptographers to apply advanced mathematical techniques to cryptology. Shannon's seminal paper [Sha49] introduces the fundamental secure private communication model still in use as we write these lines. This model, illustrated on Fig. 1.1, describes a communication between the two endpoints, sharing the same secret key K. A transmitter encrypts a plaintext P with K, *i.e.*, $C = E_K(P)$. A ciphertext C is then sent to a receiver via an unprotected channel. The receiver recovers the initial plaintext P by decrypting the ciphertext $P = E_K^{-1}(C)$. During transmission, C is observed by an *eavesdropper*. Her[3] goal is to learn P. The sender's and the receiver's goal is to secure the communication channel, so that C could not be decrypted.

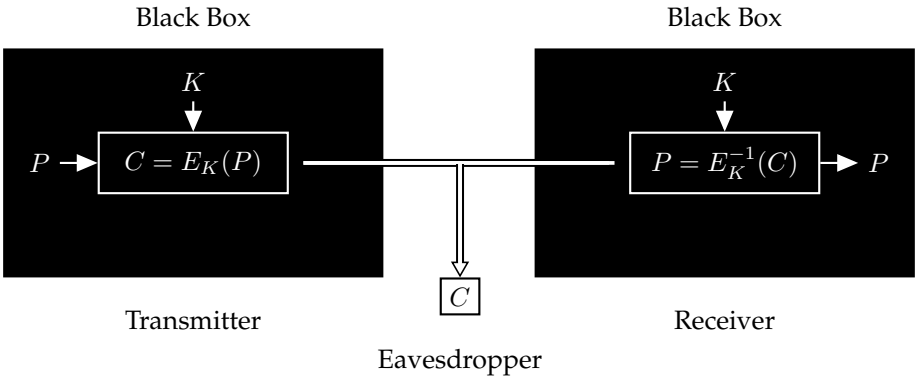


Figure 1.1 – Shannon's model of a secrecy system.

In Shannon's model, the cryptographic algorithms E_K and E_K^{-1} are assumed to be executed inside the two *Black Boxes*. The adjective *black* means that Eve does not know the secret material used inside the box, *i.e.*, keys, look-up tables, etc. The word *box* is used to indicate that there is a mechanism inside the box, which is a publicly known algorithm itself.

Shannon's model was developed during an era in which cryptography was mostly reserved for military and governmental use. During that epoch, cipher algorithm design was treated with the strictest secrecy by nations. Nevertheless, when PCs became widespread, the need for encryption in commercial applications increased. This created a need for public cryptographic algorithms. In 1973, the National Bureau of Standards (NBS, which later became the National Institute of Standards and Technology or NIST) issued a public call for a block cipher to be adopted as a standard by the U.S. government. NBS approved the Data Encryption Standard in 1976 [oS77]. This was a historically significant trigger for cipher development.

Just as telegraph and radio changed cryptography in the 19th century, embedded systems drastically influenced cryptography and cryptanalysis at the end of the 20th century. As smart objects increasingly find application in communication, medical, tracking, and other daily services, an adversary can gain access to a device during the encryption process. Gaining physical access to those devices implies that they can be examined and manipulated, so the Black Box assumption is being increasingly put in question. This imposed a significant change in the adversarial model, namely, not only the channel but also endpoints can be attacked as illustrated in Fig. 1.2.

In a nutshell, a physical system, processing a cryptographic algorithm, may suffer from different circuit-level security flaws. Firstly, a device can leak information: physical observations and measurements may be correlated to the processed data. Secondly, physical stress can modify the algorithm's processing. Malicious data modifications lead to fault analysis used for cryptanalysis [BS97].

Circuit-level vulnerabilities turned out to be serious threats on par with cryptanalytic attacks [CP02], middleware vulnerabilities [KDK+14], and software vulnerabilities [SMWO11]. Embedded system design became a systematic problem considered at different abstraction levels [HSTV06], as illustrated in Fig 1.3. These levels are:

- *Protocol* level, which performs a security-related function and applies cryptographic methods, often as a sequence of cryptographic primitives. A protocol describes how algorithms should be used.

[3] As is customary in cryptography, we often refer to the sender as "Alice", to the receiver as "Bob", and to the eavesdropper as "Eve".

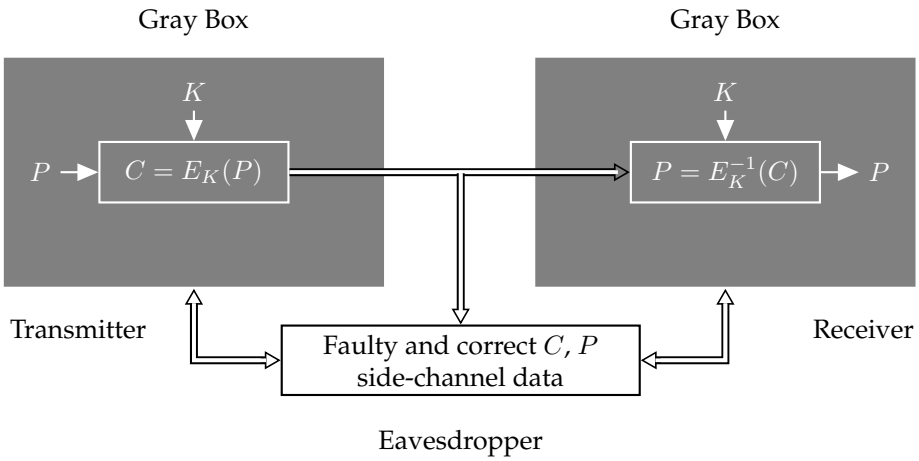


Figure 1.2 – A model of an embedded secrecy system.

The following functions are typical examples of protocol level abstractions:

  - Key agreement
  - Entity authentication
  - Secure multi-party computation

- *Algorithm* level, consisting of the design of cryptographic primitives such as hash functions or block ciphers. For example, AES [AES01], SHA [FIP95], DES [oS77], and others.
- *Architecture* level, consisting of secure hardware/software partitioning and embedded software techniques to prevent software attacks.
- *Microarchitecture* level, which deals with the hardware design of the required modules (processors and cryptoprocessors) specified at the architecture level.
- *Circuit* level, which requires implementing transistor-level and package-level techniques to thwart various physical-layer attacks, such as side-channel and fault analysis, which are the subject of this thesis.

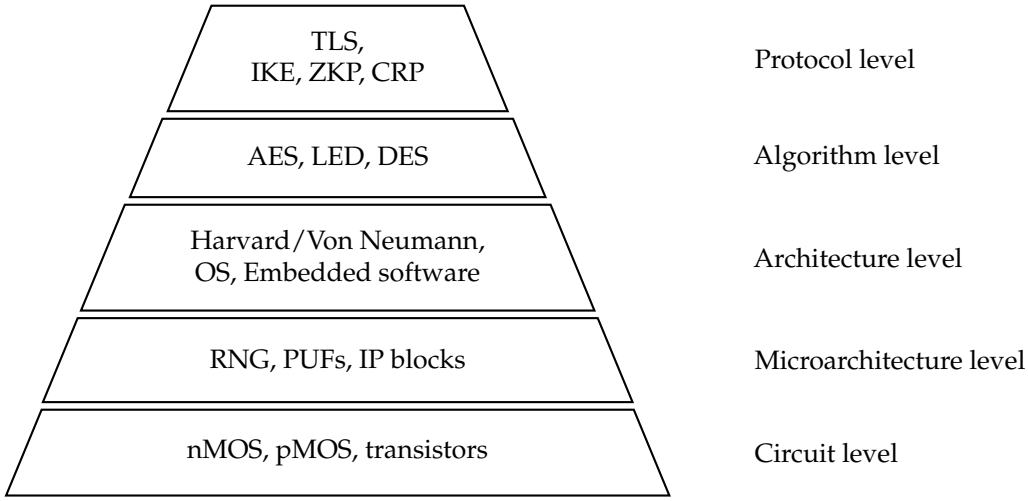


Figure 1.3 – Embedded security pyramid.

The two lowermost levels attracted a lot of attention since the middle of the 1990s. First of all, timing attacks, introduced by Kocher in 1996 [Koc96], showed that the microarchitecture level can be used to unveil secret cryptographic keys. That same year, Boneh et al. [BDL97] proposed a theoretical attack revealing a private RSA key by a single random fault injection. Then, in 1998 [KJJ99], Kocher showed

that a circuit's power consumption could also compromise system secrecy. These three articles launched the field of *hardware attacks*. The low level attacks led to a conclusion that cryptography is necessary for private communication and secure processing, but that is not sufficient. As such, cryptographic algorithms must be protected from malicious analysis, specifically side-channel and fault attacks.

The ultimate goal of this thesis is to analyse several popular protective schemes, as well as to show ways in which they can be broken. Particular attention is paid to:

- Adapt the Hilbert Huang Transform to break hiding side-channel countermeasures.
- Describe key-dependent distributions leading to "blind" key exposure, *i.e.*, without knowledge of plaintexts and ciphertexts.
- Show that the injecting of multiple faults is feasible against complex systems.

1.2 Cryptographic Terminology and Concepts

The following list of terms and basic concepts is used throughout this thesis. The definitions are taken from [HPSS08, MVOV96, MP13].

- $\mathcal{A}$ denotes a finite set called *alphabet of definitions*. $\mathcal{A} = \{0, 1\}$ is the frequently used *binary alphabet*. Note that any alphabet can be encoded in terms of the binary alphabet.
- $\mathcal{P}$ denotes a set called the *plaintext space*. According to Shannon's model illustrated on Fig. 1.1, $P \in \mathcal{P}$ called a *plaintext* is an initial data encrypted by a transmitter.
- $\mathcal{C}$ denotes a set called the *ciphertext space*. In Shannon's model, an element $C \in \mathcal{C}$ is called a *ciphertext*.
- $\mathcal{K}$ denotes a set called the *key space*. An element $K \in \mathcal{K}$ is called a *key*.
- Each element $K \in \mathcal{K}$ uniquely determines a bijection from $\mathcal{P}$ to $\mathcal{C}$, denoted by E_K. E_K is called an *encryption function* or an *encryption transformation*. Note that E_K must be a bijection if the process is to be reversed for a unique plaintext message to be recovered from each distinct ciphertext.
- An element $K^D \in \mathcal{K}$ determines a bijection from $\mathcal{C}$ to $\mathcal{P}$, denoted by D_{K^D}. D_{K^D} is called a *decryption function* or *decryption transformation*.
- The process of applying the transformation E_K to a plaintext $P \in \mathcal{P}$ is referred to as *encrypting* P or the *encryption* of P.
- The process of applying the transformation D_{K^D} to a ciphertext $C \in \mathcal{C}$ is referred to as *decrypting* C or the *decryption* of C.
- An *encryption scheme* consists of a set $\{E_K : K \in \mathcal{K}\}$ of encryption transformations and a corresponding set $\{D_{K^D} : K^D \in \mathcal{K}\}$ of decryption transformations with the property that $\forall K \in \mathcal{K}$ there is a key $K^D \in \mathcal{K}$ such that $D_{K^D} = E_K^{-1}$; that is, $D_{K^D}(E_K(P)) = P$ for all $P \in \mathcal{P}$. An encryption scheme is also referred to as a *cipher*.
- The keys K and K^D are referred to as a *key pair* and are sometimes denoted by (K, K^D). Note that K and K^D could be identical.
- $\mathcal{M}$ is the set of messages which can be signed.[4] $\mathcal{M}$ consists of strings of symbols from $\mathcal{A}$.
- $\mathcal{S}$ is a set of elements called *signatures*, possibly binary strings of a fixed length.
- Sign_A is a transformation from $K \times \mathcal{M}$ to $\mathcal{S}$, called a *signing transformation* for entity A[5]. The transformation Sign_A is kept secret by A, and will be used to create signature for messages from $\mathcal{M}$.
- Ver_A is a transformation from the set $K^{-1} \times \mathcal{M} \times \mathcal{S}$ to the set {*true*, *false*}. Ver_A, called a *verification algorithm* for A's signatures, is publicly known, and is used by other entities to verify signatures created by A.

[4] To prevent ambiguity between a plaintext $P \in \mathcal{P}$ and an RSA's prime number p the messages space notation $\mathcal{M}$ for asymmetric ciphers is different from the plaintexts space notation $\mathcal{P}$ used in symmetric ciphers.

[5] The names of Alice and Bob are usually abbreviated to A and B respectively

Definition 1 [Ring] A *ring* is a nonempty set R together with two operations, "+" and "·" such that:

1. $(R, +)$ is an abelian group;
2. $\cdot$ is associative, that is for all $a, b, c \in R,\ a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
3. left and right distributive laws hold: for all $a, b, c \in R$

$$a \cdot (b + c) = a \cdot b + a \cdot c \text{ and } (b + c) \cdot a = b \cdot a + c \cdot a$$

Definition 2 [Field] Let R be a ring.

1. R is a *ring with identity* if the ring has a multiplicative identity.
2. R is *commutative* if "·" is commutative.
3. R is an *integral domain* if it is commutative with identity and $a \cdot b = 0$ implies $a = 0$ or $b = 0$, for any $a, b \in R$.
4. R is a *division ring* (also called a skew field) if a nonzero element of R form a group under "·".
5. R is a *field* if it is a commutative division ring.

If p is prime, then the set $\mathbb{F}_p$ of integers modulo p with its addition, subtraction, multiplication, and division rules is a field. Finite fields are also sometimes called *Galois fields*, after Évariste Galois [Gal97], who studied them in the 19th century. Yet another notation for $\mathbb{F}_p$ is GF(p), in honor of Galois. An additional notation for $\mathbb{F}_p$ is $\mathbb{Z}_p$, although in number theory the notation $\mathbb{Z}_p$ is more commonly reserved for the ring of p-adic integers.

Definition 3 [Symmetric Cipher] An encryption scheme $\{E_K : K \in \mathcal{K}\}$, $\{D_{K^D} : K^D \in \mathcal{K}\}$ is called *symmetric cipher* (also *symmetric-key, single-key, one-key*, and *conventional* [Mol06]) if for each key pair (K, K^D), it is computationally easy to determine K knowing only K^D, and to determine K^D knowing only K.[6]

Two types of symmetric ciphers are commonly distinguished: *block ciphers* and *stream ciphers*.

Definition 4 [Block Cipher] A *block cipher* is an encryption scheme that splits the plaintext P into strings, called *blocks*, of fixed length n, called the *block length*, over an alphabet $\mathcal{A}$, and enciphers one block at a time.

Stream ciphers are, in one sense, very simple block ciphers having block length equal to one.

Definition 5 [Keystream] Let $\mathcal{K}$ be the key space for a set of encryption transformations. A sequence of symbols $K_1, K_2, K_3 \ldots K_i \in \mathcal{K}$, is called a *keystream*.

Definition 6 [Stream cipher] Let $\mathcal{A}$ be an alphabet of q symbols and let E_K be a simple substitution cipher with block length 1 where $K \in \mathcal{K}$. Let $P_1, P_2, P_3 \ldots$ be a plaintext string and let $K_1, K_2, K_3 \ldots$ be a keystream from $\mathcal{K}$. A *stream cipher* takes the plaintext string and produces a ciphertext string $C_1, C_2, C_3 \ldots$ where $C_i = E_{K_i}(P_i)$.

Most well-known symmetric encryption techniques are block ciphers. Two important classes of block ciphers are *substitution ciphers* and *transposition ciphers*.

Definition 7 [Simple Substitution Cipher] Let $\mathcal{A}$ be an alphabet of q symbols and $\mathcal{P}$ be the set of q^n strings of length n over $\mathcal{A}$. Let $\mathcal{K}$ be the set of all permutations over $\mathcal{A}$. Define for each $K \in \mathcal{K}$ an encryption transformation E_K as:

$$E_K(P) = (K(P_1), K(P_2), \ldots, K(P_n)) = (C_1, C_2, \ldots, C_n) = C$$

where $P = (P_1, P_2, \ldots, P_n) \in \mathcal{P}$. In other words, replace (substitute) each symbol $P_i \in \mathcal{A}$ in an n-tuple by another symbol $K(P_i) \in \mathcal{A}$ according to some fixed permutation K. E_K is called a *simple substitution cipher* or a *mono-alphabetic substitution cipher*.

Definition 8 [Simple Transposition Cipher] Consider a symmetric block encryption scheme with block length n. Let $\mathcal{K}$ be the set of permutations of the set $\{1, 2, \ldots n\}$. For each $K \in \mathcal{K}$ define the encryption function

$$E_K(P) = (P_{K(1)}, P_{K(2)}, \ldots, P_{K(n)})$$

[6] In most practical symmetric ciphers $K = K^D$.

where $P = (P_1, P_2, \ldots, P_n) \in \mathcal{P}$ is the message space. The set of all such transformations is called a *simple transposition cipher*.

The modern design of most block ciphers is based on the concept of iterated product ciphers. Product ciphers were suggested and analysed by Claude Shannon in his seminal publication [Sha49]. To describe product ciphers, the concept of *composition of functions* is introduced.

Definition 9 [Composition of Functions] Let $\mathcal{S}$,$\mathcal{T}$, and $\mathcal{U}$ be finite sets and let $f : \mathcal{S} \to \mathcal{T}$ and $g : \mathcal{T} \to \mathcal{U}$ be functions. The *composition* of g and f, denoted $g \circ f$ (or simply gf), is a function from $\mathcal{S}$ to $\mathcal{U}$ defined by $(g \circ f)(x) = g(f(x))$ for all $x \in \mathcal{S}$.

Definition 10 [Product Cipher] A *product cipher* is a composition of $t \geq 2$ transformations $E_{K_1} E_{K_2} \ldots E_{K_t}$ where each E_{K_i}, $1 \leq i \leq t$, is either a substitution or a transposition cipher.

The composition of substitutions and transpositions repetitively applied by a block cipher is called a *round*. A substitution is said to add *confusion* to the encryption process whereas transposition is said to add *diffusion*. Product ciphers carry out encryption in multiple rounds, each of which uses a different subkey derived from the original key. One widespread implementation of such ciphers is called a *Feistel network* [Fei73], named after Horst Feistel, and notably implemented in the DES cipher. Many other realizations of block ciphers, such as the AES, are classified as *Substitution-Permutation Networks*.

In practical ciphers, confusion is often implemented as a set of look-up tables or *S-boxes*, denoted as **S**. An S-box is a nonlinear transform used to map a b-bit element into q-bit element:

$$\mathbf{S} : \mathbb{F}_{2^b} \to \mathbb{F}_{2^q} \tag{1.1}$$

During the confusion stage the current n-bit string is fed into an array of m S-boxes, where $n = b \times m$. The same set of S-boxes may be used in each round, or S-boxes may change from round to round.

The diffusion layer, denoted as **D**, is a linear transform that reshuffles n-bit inputs.

$$\mathbf{D} : (\mathbb{F}_{2^b})^m \to (\mathbb{F}_{2^b})^m \tag{1.2}$$

The main purpose of diffusion is to spread small input variations over a significant amount of output bits. **D** is designed so that the output bits of any given S-box are spread over different S-boxes in the next round.

A key mixing operation, denoted as **A**, combines the n-bit input with an n-bit round key K_i.

$$\mathbf{A}\,|_{K^{[i]}} : \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \to \mathbb{F}_{2^n} \tag{1.3}$$

The round keys $K^{[i]}$ are derived from the master key K according to a *key schedule* algorithm. The key schedule is often made of a simple confusion-diffusion operations set.

Digital signature is another fundamental cryptographic primitive, which is used in authentication and non-repudiation. The process of *signing* combines a message and some secret information held by the signing entity into a binary string called a *signature*.

Definition 11 [Digital Signature] A *digital signature scheme* consists of three probabilistic, polynomial time algorithms (Gen, Sign, Vrfy) along with an associated message space $\mathcal{M} = \{m_i\}$ such that:

- The randomized *key-generation algorithm* Gen takes as input the security parameter r (in unary). It outputs a pair of keys $(K_{\text{pub}}, K_{\text{priv}})$ where K_{pub} is called the public key or the verification key, and K_{priv} is called the *private key*, the *secret key*, and the *signing key*.
- For security parameter r, the (possibly randomized) signing algorithm Sign takes as input a secret key K_{priv} and a message $m \in \mathcal{M}$ and outputs a signature s. We write this as $s \leftarrow \mathsf{Sign}_{K_{\text{priv}}}(m)$.
- For security parameter r, the deterministic verification algorithm Vrfy takes as input a public key K_{pub}, a message $m \in \mathcal{M}$, and a (purported) signature s. It outputs a single bit b, with $b = 1$ signifying "accept" and $b = 0$ signifying "reject". We write this as $b \leftarrow \mathsf{Vrfy}_{K_{\text{pub}}}(m, s)$.

To describe fault attacks against block ciphers the following definitions are required:

Definition 12 [The Hamming Weight] The *Hamming weight* of a string x over an alphabet of definitions $\mathcal{A}$ is defined as a number of non-zero symbols in the string. More formally, $\text{HW}(x) = |\{i : x_i \neq 0\}|$.

Definition 13 [T-Radical Branch Number] *T-radical branch number* $\mathcal{B}_T$ of a linear diffusion layer $\mathbf{D}$ is defined as:

$$\mathcal{B}_T(\mathbf{D}) = \min_{\mathrm{HW}(x)=T} \{\mathrm{HW}(\mathbf{D}(x))\}, \;\; x \in (\mathbb{F}_{2^b})^m$$

Definition 14 [Entropy] Let $X \in \mathbb{F}_{2^b}$ be a discrete random variable. Then, the *entropy* of X is defined to be the following quantity expressed in bits

$$\mathrm{H}(X) = -\sum_{x \in \mathbb{F}_{2^b}} \Pr[X = x] \log_2(\Pr[X = x]) \tag{1.4}$$

Note that if variable X is uniformly distributed (*i.e.*, $\Pr[X = x] = 2^{-b}, \; \forall x \in \mathbb{F}_{2^b}$) then $\mathrm{H}(X) = b$.

Theorem 1 [Primitive Root Theorem] Let $p \in \mathbb{N}$ be a prime number. Then there exists an element $g \in \mathbb{F}_p$ whose powers give every element of $\mathbb{F}_p$, *i.e.*:

$$\mathbb{F}_p = \{1, g, g^2, \ldots, g^{p-2}\}$$

Elements with this property are called *primitive roots* of $\mathbb{F}_p$ or *generators* of $\mathbb{F}_p$.

The number of primitive roots in the finite field $\mathbb{F}_p$ is given by Euler's phi function $\phi(p-1)$.

Definition 15 [Euler's Phi Function] *Euler's phi function* (also known as *Euler's totient function*) is the function $\phi(p)$ defined as follows

$$\phi(p) = \#\mathbb{F}_p = \#\{0 \leq a \leq p : \mathrm{GCD}(a, p) = 1\}$$

Theorem 2 [Fermat's Little Theorem] Let $p \in \mathbb{N}$ be a prime number and let $a \in \mathbb{N}$. Then

$$a^{p-1} = \begin{cases} 1 \bmod p \text{ if } p \nmid a \\ 0 \bmod p \text{ if } p \mid a \end{cases}$$

where $p \nmid a$ denotes that a is not divisible by p and $p \mid a$ denotes that a is divisible by p.

1.3 Block Ciphers

1.3.1 Substitution-Permutation Networks

A *Substitution Permutation Network* (SPN) is a composition of invertible transforms. A typical SPN-based block cipher, shown on Figure 1.4, consists of N_r rounds described by equation (1.5).

$$E_K : \mathbf{A}\,|_{K^{[N_r]}} \circ \Big|\Big|_{i=1}^{m} \mathbf{S}_i^{[N_r]} \circ \left(\bigcirc_{r=1}^{N_r-1} \mathbf{A}\,|_{K^{[r]}} \circ \mathbf{D}^{[r]} \circ \Big|\Big|_{i=1}^{m} \mathbf{S}_i^{[r]} \right) \circ \mathbf{A}\,|_{K^{[0]}} \tag{1.5}$$

where the notation $||_{i=1}^{m} \mathbf{S}_i^{[r]}$ indicates S-box outputs concatenation.

Note that the very first and last operations performed in this SPN are sub-key mixing operations. This is called *whitening* and is regarded as a useful way to prevent an attacker from even beginning to carry out an encryption or decryption operation if the key is unknown.

At the last round, $\mathbf{D}$ is not applied. Consequently, the encryption algorithm can also be used for decryption, if appropriate modifications are made to the key schedule and if all the transformations $\mathbf{D}^{[r]}$, $\mathbf{S}_i^{[r]}$, and $\mathbf{A}\,|_{K^{[r]}}$ are replaced by their inverses. To ensure invertibility the SPNs' S-boxes must be bijective.

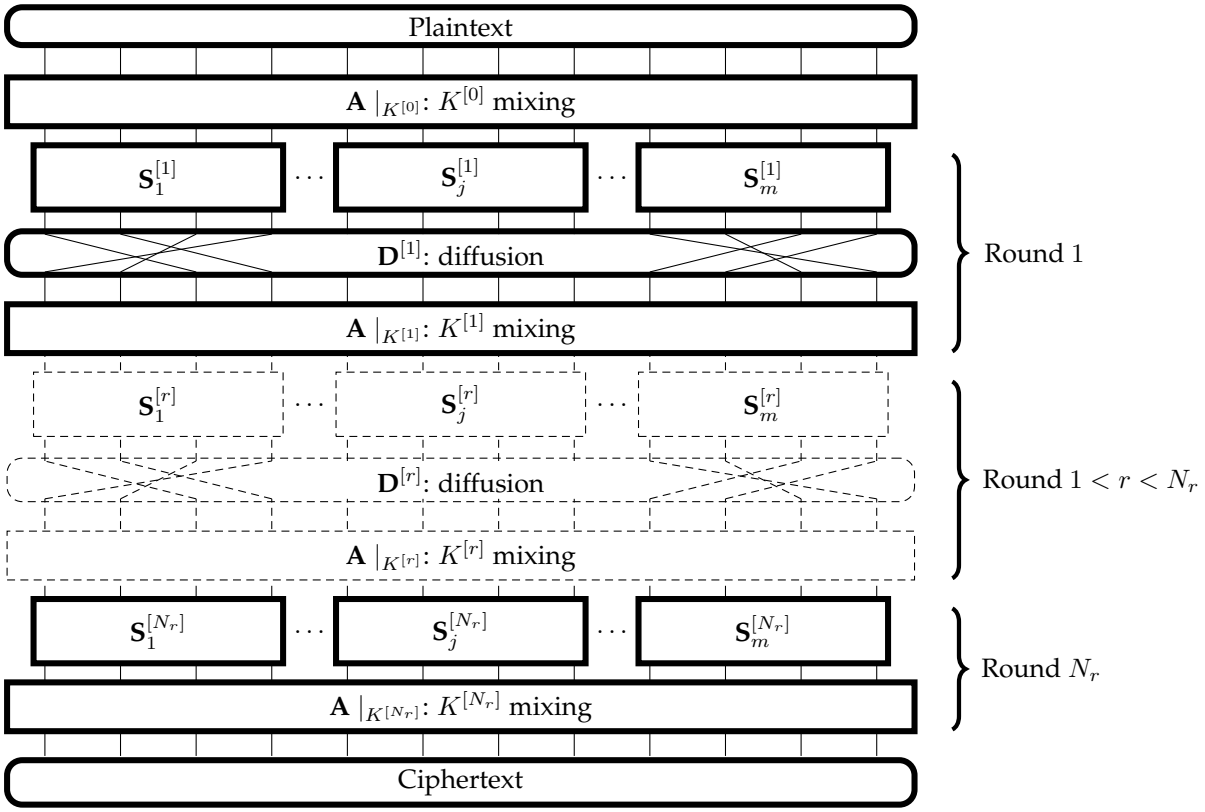


Figure 1.4 – A typical SPN-based block cipher.

The Advanced Encryption Standard

The *Advanced Encryption Standard* (AES) is an SPN-based block-cipher that processes 128-bit blocks and supports keys of 128, 192 or 256 bits [AES01]. Key length is denoted by $N_K = 4, 6$, or 8, and reflects the number of 32-bit words in the key. At start, the 128-bit plaintext P is split into a 4×4 matrix S of 16 bytes called . The state goes through a number of rounds to become the ciphertext C.

The number of rounds N_r is a function of N_K. Possible $\{N_r, N_K\}$ combinations are $\{10, 4\}$, $\{12, 6\}$ and $\{14, 8\}$. A particular round $1 \leq r \leq N_r$ takes as input a 128-bit state $S^{[r]}$ and a 128-bit round key $K^{[r]}$ and outputs a 128-bit state $S^{[r+1]}$. This is done by successively applying four transformations called SUBBYTES, SHIFTROWS, MIXCOLUMNS and ADDROUNDKEY.

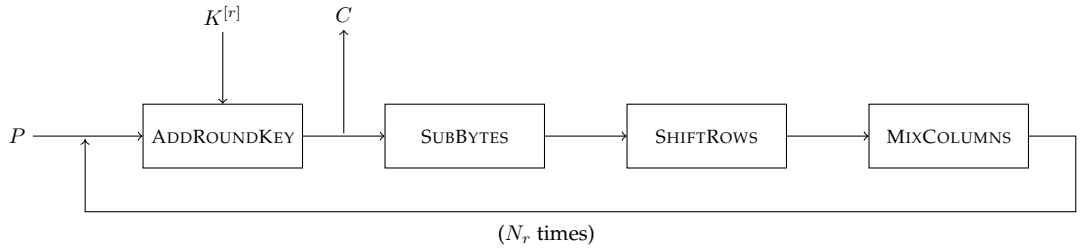


Figure 1.5 – AES encryption flowchart.

AES encryption starts with an initial ADDROUNDKEY transformation followed by N_r rounds consisting of four transformations, in the following order: SUBBYTES, SHIFTROWS, MIXCOLUMNS and ADDROUNDKEY. MIXCOLUMNS is skipped in the final round ($r = N_r$). If during the last round MIXCOLUMNS is bypassed, we can look upon the AES as the 4-block iterative structure shown in Fig. 1.5. Decryption has a similar structure where the order of transformations is reversed (Fig. 1.6) and where inverse transformations are used (Note that ADDROUNDKEY is idempotent).

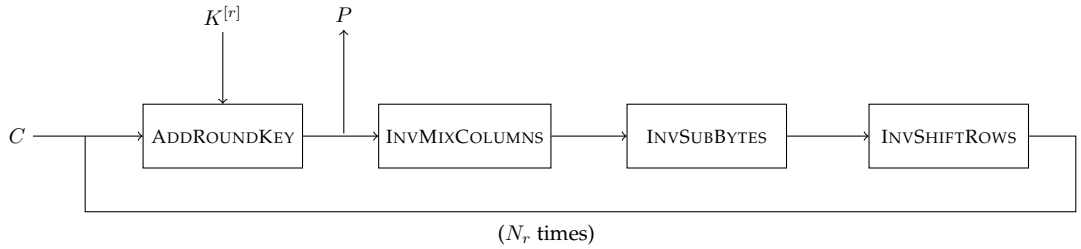


Figure 1.6 – AES decryption flowchart.

1.3.2 Feistel Networks

A *Feistel Network* illustrated on Fig. 1.7 is an alternative block cipher design [oS77]. The building blocks, such as confusion, diffusion, and key mixing, are the same; the difference is at the high-level design.

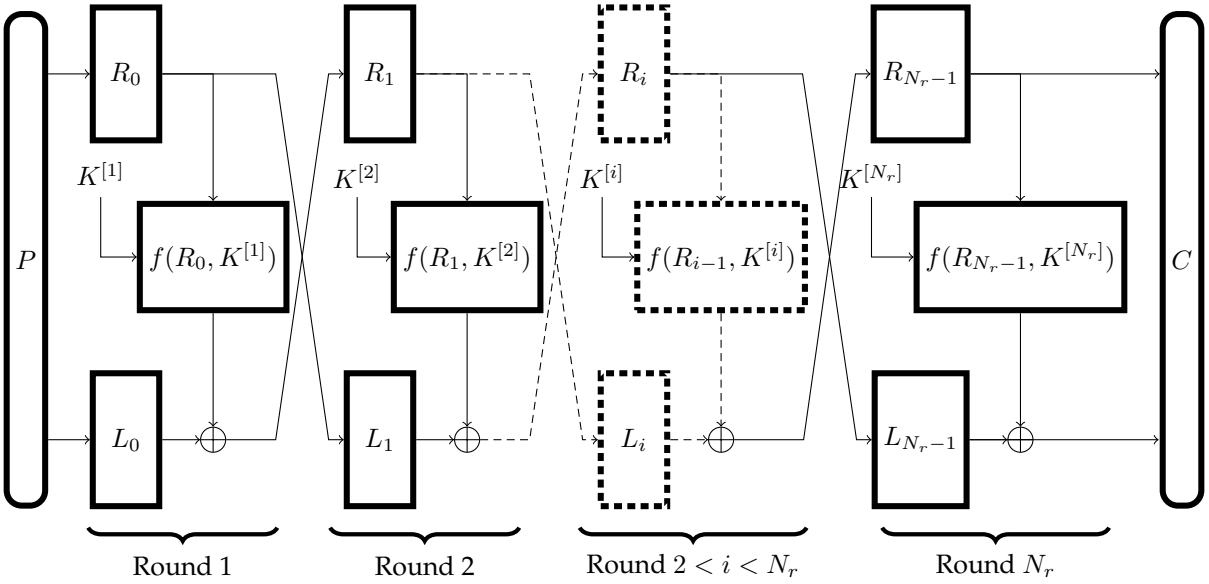


Figure 1.7 – A typical Feistel network.

Similar to SPNs, a Feistel network operates in a series of rounds. Each round applies a *round function* that needs not be invertible. Round functions typically contain components like S-boxes and mixing permutations, but a Feistel network can deal with *any* transformation irrespective of their design [KL07]. A Feistel network applies a set of subkeys $K^{[1]}, K^{[2]}, ..., K^{[r]}$ derived from a master key K.

The i-th round of a Feistel network operates as follows. The input to the round is divided into two halves of size $n/2$ denoted L_{i-1}, R_{i-1} (with L and R denoting the "left half" and "right half" of the input, respectively). The i-th round function f_i takes an $n/2$-bit input R_{i-1} and a round key $K^{[i]}$ to produce an $n/2$-bit output. The output (L_i, R_i) of the round is given by

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f_i(R_{i-1}, K^{[i]})$$

Splitting an n-bit plaintext into two $n/2$ values gives the initial left L_0 and right R_0 halves.

A Feistel network is invertible regardless of the round functions f_i. Given the output (L_i, R_i) of the i-th round, (L_{i-1}, R_{i-1}) can be computed as follows:

$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus f_i(R_{i-1}, K^{[i]})$$

Data Encryption Standard

DES is the most famous Feistel network cipher. This section provides a high-level overview of the DES main components. The detailed description can be found in the DES specification [oS77].

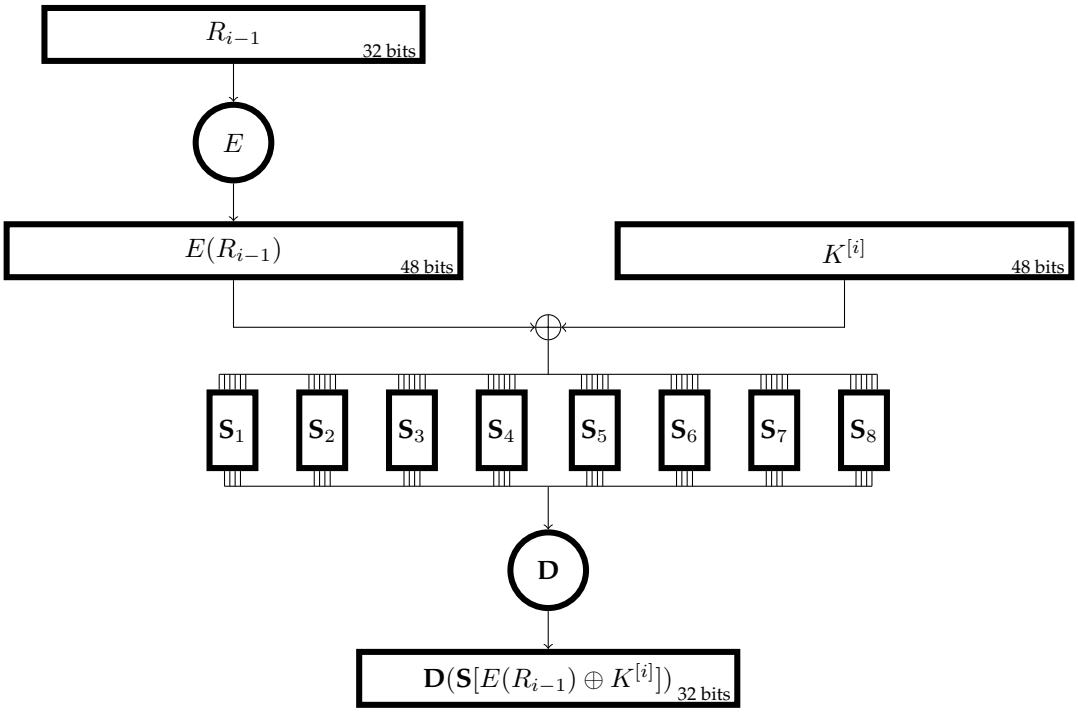


Figure 1.8 – DES round function $f(R_{i-1}, K^{[i]})$.

The DES block cipher is a 16-round Feistel network with a block length of 64 bits and a key length of 56 bits.[7] The DES key schedule derives 48-bit round keys $K_1, ..., K_{16}$. All rounds apply the same non-invertible round function $f(R_{i-1}, K^{[i]})$ illustrated on Fig. 1.8. The round function transforms a 32-bit input R_{i-1} and a 48-bit round key $K^{[i]}$ into a 32-bit output as follows:

$$f(R_{i-1}, K^{[i]}) = \mathbf{D}(\mathbf{S}[E(R_{i-1}) \oplus K^{[i]}])$$

An input R_{i-1} is *expanded* to 48-bit value R'_{i-1}. This is done by simply duplicating half the bits of R_{i-1}, denoted by $R'_{i-1} = E(R_{i-1})$ where E represents the *expansion function*.[8] Following this step, computation proceeds similary to an SPN: the expanded value R'_{i-1} is xored with the round key $K^{[i]}$, and the resulting value is divided into 8 blocks, each of which is 6-bit long. Each block is passed through a (different) S-box that yields a 4-bit output. All S-boxes outputs are concatenated into a 32-bit value. As a final step, a mixing permutation $\mathbf{D}$ [9] is applied to obtain the round function's output.

An *initial permutation* (IP) of the 64-bit input block is added to the DES beginning. To maintain the property that the encryption network can be reused for decryption, DES requires the inverse operation IP^{-1} to be applied to the output of the network.

1.4 Public Key Cryptography

If Alice and Bob want to exchange messages using a symmetric cipher, they must first agree on a secret key K. The eavesdropper (Eve) monitors the communication channel between the sender and receiver, so the key K cannot be sent in clear. A solution for secure key exchange was proposed by Whitfield

[7] Every 8-th bit of 64-bit key is used for parity.
[8] Be aware that the expansion function notation E is different from the encryption notation E_K.
[9] To avoid confusion with a plaintext notation P, the standard DES mixing permutation notation P is replaced by $\mathbf{D}$.

Diffie and Martin Hellman [DH76].[10] As usual, there are spaces of keys $\mathcal{K}$, plaintexts $\mathcal{M}$, and ciphertexts $\mathcal{C}$. However, an element $K \in \mathcal{K}$ is a pair of keys:

$$K = (K_{\text{priv}}, K_{\text{pub}})$$

composed of the *private key* and the *public key*, respectively. For each public key K_{pub} there is a corresponding encryption function:

$$E_{K_{\text{pub}}} : \mathcal{M} \to \mathcal{C}$$

and for each private key K_{priv} there is a corresponding decryption function:

$$D_{K_{\text{priv}}} : \mathcal{C} \to \mathcal{M}$$

These have the property that if the pair $K = (K_{\text{priv}}, K_{\text{pub}})$ belongs to the key space $\mathcal{K}$, then:

$$\forall m \in \mathcal{M} : \quad D_{K_{\text{priv}}}\left(E_{K_{\text{pub}}}(m)\right) = m$$

For an asymmetric cipher to be secure, it must be difficult for Eve to compute the decryption $D_{K_{\text{priv}}}(c)$ function even if she knows the public key K_{pub}. Note that under this assumption, Alice can send K_{pub} to Bob using an insecure communication channel, and Bob can send back the ciphertext $E_{K_{\text{pub}}}(m)$, without worrying that Eve will be able to decrypt the message m.

1.4.1 Diffie-Hellman's Key Exchange

Diffie-Hellman's key exchange solves the secure key exchange problem over unprotected channels, where all traffic is observed by Eve. The protocol relies on the *Discrete Logarithm Problem* (DLP), *i.e.*, the absence of an efficient general method for computing discrete logarithms on conventional computers.

Definition 16 [Discrete Logarithm] Let g be a primitive root of $\mathbb{F}_p$ and let $h > 1$ to be an element of $\mathbb{F}_p$. The *Discrete Logarithm Problem* (DLP) in $\mathbb{F}_p$ is the problem of finding an exponent x such that

$$g^x = h \bmod p$$

The number x is called the *discrete logarithm of h to the base g* and is denoted by $\log_g(h)$.

The Diffie-Hellman protocol is illustrated on Fig. 1.9. The first step is to agree on a large prime p and a primitive root $g \bmod p$. Secret values, that cannot be transmitted over the insecure channel, are shown in red in Fig. 1.9. The prime p and the integer g are publicly known, *e.g.*, they might be posted in a public directory. The next step for Alice is to pick a secret integer x_A. Bob picks an integer x_B that he keeps secret. Bob and Alice use their secret integers x_A, x_B to compute Y_A and Y_B respectively. Y_A and Y_B are public, so Alice and Bob can exchange these values over the insecure channel. Finally, Bob and Alice use their secret integers x_A, x_B to compute the values $Y_B^{x_A} \bmod p$ and $Y_A^{x_B} \bmod p$, which are identical. This common value is used to derive the shared key K.

Eve knows the values Y_A and Y_B, so she knows $g^{x_A} \bmod p$ and $g^{x_B} \bmod p$. She also has the values of g and p. If she can solve the DLP, then she can find x_A and x_B, which allows her to compute the shared secret value K. Alice and Bob are safe unless Eve is able to solve the DLP. More precisely, they are safe until Eve can solve the *Diffie-Hellman Problem* (DHP). The DHP is no harder than the DLP.

Definition 17 [Diffie-Hellman Problem] Let p be a prime number and g a generator. The *Diffie-Hellman Problem* (DHP) is the problem of computing the value of $g^{ab} \bmod p$ given $g^a \bmod p$ and $g^b \bmod p$.

[10] It turns out that the concept of public key encryption was originally discovered by James Ellis while working at the British Government Communications Headquarters (GCHQ). Ellis's discoveries in 1969 were classified by the British government and were not declassified and released until 1997, after Ellis' death. It is now known that two other GCHQ researchers, Malcolm Williamson and Clifford Cocks, discovered the Diffie-Hellman key exchange algorithm and the RSA encryption scheme, respectively, before their rediscovery and public dissemination by Diffie, Hellman, and Rivest, Shamir, and Adleman. To learn more about the fascinating history of public key cryptography, see for example [Ada97, Ell97, HPSS08, Sin11].

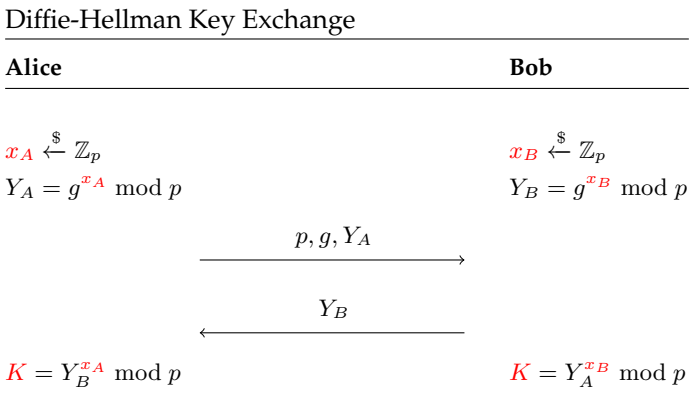


Figure 1.9 – Diffie-Hellman key exchange.

1.4.2 The Rivest-Shamir-Adleman Algorithm

Bob and Alice have the usual problem of exchanging secret information over the insecure channel. Diffie-Hellman key exchange accomplishes the task of secure communication relying on the conjectured hardness of the DHP. The RSA public key algorithm is based on another paradigm, namely, the difficulty of factorizing large numbers.

RSA is the acronym of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977 [RSA78]. RSA key generation, encryption, and decryption are summarized on Fig. 1.10. All the secret algorithm's components are shown in red.

Alice's secret key is a pair of large primes p and q. Her public key is the pair (N, e) consisting of the product $N = pq$ and an encryption exponent e relatively prime to $(p-1)(q-1)$, *i.e.*, GCD $(e, (p-1)(q-1)) = 1$. Bob takes his plaintext and converts it into an integer $m = N$. Bob encrypts m using the public key

$$c = m^e \bmod N$$

The integer c is the ciphertext, that Bob sends to Alice over the insecure channel. Using Fermat's Little Theorem Alice can recover the plaintext m by computing

$$c^d \bmod N = m^{ed} \bmod N = m^{1+k(p-1)(q-1)} \bmod N = m \bmod N$$

A crypto scheme is said to be *malleable* if the attacker is capable of transforming the ciphertext into another ciphertext which leads to a known transformation of the plaintext [PP09]. The attacker does not decrypt the ciphertext; however, he is capable of manipulating the plaintexts in a predictable manner. In case of RSA this is easily achievable. Let the attacker to replace the ciphertext c by $r^e c$, where $r \in \mathbb{N}$. If the receiver decrypts the manipulated ciphertext, he computes:

$$(r^e c)^d = r^{ed} m^{ed} = rm \bmod N$$

The malleable ciphertext problem can be resolved by padding, which embeds a random structure into the plaintext before encryption. Modern techniques such as *Optimal Asymmetric Encryption Padding* (OAEP) for padding RSA messages are specified and standardized in Public Key Cryptography Standard #1 (v2.2) [Lab12].

The security of RSA depends on the adversary inability of computing d from the public key (e, N), which is equivalent to the problem of factoring N into its prime factors p and q as is proven by Bach [BMS86]. Therefore the primes p and q must be correctly chosen. Choosing p and q as strong primes has been recommended as a way of maximizing the difficulty of factoring N [RS97].

The definition of a strong prime is given in [RS97]. Let $|p|$ be used to denote the length of p in binary. The following definition is using English words "large prime" that are clarified with specific recommendations on sizes.

The RSA public key cryptosystem

Alice (sender)	**Bob (receiver)**
..... Key generation	
$p \xleftarrow{\$} \mathbb{N}$	
$q \xleftarrow{\$} \mathbb{N}$	
$e \xleftarrow{\$} \mathbb{N}$ such that $\text{GCD}(e, (p-1)(q-1)) = 1$	
$d = e^{-1} \bmod (p-1)(q-1)$	
$N = pq$	
	$\xrightarrow{N, e}$
..... Encryption	
	$m \in \mathbb{N},\ m < N$
	$c = m^e \bmod N$
	$\xleftarrow{c}$
..... Decryption	
$m = c^d \bmod N$	

Figure 1.10 – RSA key generation, encryption and decryption.

Definition 18 [A Strong Prime] A prime p is considered to be a strong prime if the following conditions are satisfied:

- p is a large prime (say, $|p| \geq 256$).
- The largest prime factor of $p-1$, denoted p^-, is large (say $|p^-| \geq 100$). That is
$$p = a^- p^- + 1$$
for some integer a^- and large prime p^-.
- The largest prime factor of $p^- - 1$, denoted p^{--}, is large (say $|p^-| \geq 100$). That is
$$p^- = a^{--} p^{--} + 1$$
for some integer a^{--} and large prime p^{--}.
- The largest prime factor of $p+1$, denoted p^+, is large (say $|p^+| \geq 100$). That is
$$p = a^+ p^+ - 1$$
for some integer a^+ and large prime p^+.

Definition 19 [Factorization Problem] The integer factorization problem (FACT) is the following: given a positive integer N, find its prime factorization, *i.e.*, find pairwise distinct primes p_i and positive integers e_i such that $N = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$.

The best achieved factoring results against 768-bit RSA was reported by [KAF+10]. The authors applied number field sieve factoring method [LLJMP93].

RSA algorithm is also based on another problem, namely, the *e-th root problem* (ERP).

Definition 20 [*e-th Root Problem*] Let G be a group and $e < |G|$ be an integer. An element $b \in G$ is called e^{th} root of an element $a \in G$ if we have:
$$b^e = a$$

If $GCD(e, |G|)$ holds, then an e^{th} root always exists and is unique.

Definition 21 [*e-th Root Problem*] Given a group G of unknown order, a positive integer $e < |G|$ and an element $a \in G$, find an element $b \in G$ such that $b^e = a$.

1.5 Digital Signatures

Symmetric and asymmetric encryption schemes solve the problem of secure communication over insecure channels. *Digital signatures* solve a different problem, analogous to the purpose of a pen-and-ink signature on a physical document. The signer (Alice) has a message m and she wants to create an additional piece of information s that can be used to prove the message m belongs to her. The verifier (Bob) wants to ascertain that the pair (m, s) originates from the signer.

1.5.1 RSA Digital Signature

The RSA algorithm can also be used for signing a message m and verifying its signature. The signature algorithm is similar to decryption, except that the message m is "decrypted" with the private key (d, p, q) as shown on Fig. 1.11. The validity of the signature s is verified similarly to the RSA encryption

$$s^e \bmod N = m^{ed} \bmod N = m$$

The RSA digital signature

Alice (signer)		**Bob (verifier)**
..... Key creation		
$p \xleftarrow{\$} \mathbb{N}$		
$q \xleftarrow{\$} \mathbb{N}$		
$e \xleftarrow{\$} \mathbb{N}$ such that $\text{GCD}(e, (p-1)(q-1)) = 1$		
$d = e^{-1} \bmod (p-1)(q-1)$		
$N = pq$		
	$N, e \longrightarrow$	
..... Signature		
$m \in \mathbb{N},\ m < N$		
$s = m^d \bmod N$		
	$s \longrightarrow$	
..... Verification		
		Check that $m = s^e \bmod N$

Figure 1.11 – RSA digital signature.

The NSA document [NSA15] recommends to use at least 3072-bit modulus when RSA is used for key establishment and authentication. The computation of RSA signatures can be accelerated by a factor four using the Chinese Reminder Theorem (CRT) [QC82] where a private key is given by the components (p, q, d_p, d_q, i_q) with $d_p = d \bmod (p-1)$, $d_q = d \bmod (q-1)$ and $i_q = q^{-1} \bmod p$. The CRT-RSA process is the following:

First compute:

$$s_p = m^{d_p} \bmod p$$
$$s_q = m^{d_q} \bmod q$$

Then compute a final signature using, either Garner's recombination method:

$$s = \text{CRT}(s_p, s_q) = s_q + q\left(q^{-1}(s_p - s_q) \bmod p\right) \tag{1.6}$$

or Gauss's recombination method:

$$s = \text{CRT}(s_p, s_q) = \left(s_p q(q^{-1} \bmod p)\right) + \left(s_q p(p^{-1} \bmod q)\right) \tag{1.7}$$

CHAPTER 2

SIDE-CHANNEL ATTACKS

Summary

Data and computation are the physical quantities in a physical structure, for example, the charge on a capacitor or a transistor's state. As embedded systems increasingly find applications in communication, medical object, tracking, and other services an adversary can access a device and analyse device's physical quantities. Leaked physical information can be used to extract secret data by side-channel attacks.

Side-channel attacks are a serious concern as with moderate efforts they allow to extract secret information from various embedded systems. Companies and institutions spend money to research and to develop countermeasures against side-channel attacks. As we write these lines a query "Differential Power Analysis" on Google patents service [Goo] results in more than 4,000 entries. Similar query on IEEE Xplore digital library [IEE] results in more than 6,500 publications.

Side-channel attacks, *i.e.*, the methods exploiting device's physical leakage, are explained in this chapter. Section 2.1 discusses static and dynamic power dissipation in CMOS circuits. Section 2.2 explains what type of information can be extracted with different side-channels. Section 2.3 sub-divides SCA into 4 groups: *simple*, *model-response*, *template*, and *algebraic* attacks, and presents three statistical distinguishers: Difference of Means, Pearson Correlation Coefficient, and Mutual Information. Section 2.4 describes side-channel countermeasures.

2.1 Why Circuits Leak?

The physical interpretation of data processing (a discipline named the *physics of computational systems* [MC80]) draws fundamental comparisons between computing technologies and provides physical lower bounds on the area, time, and energy required for computation [Ben73, Key75]. In this framework, a corollary of the second law of thermodynamics states that in order to perform a transition between states, energy must be lost irreversibly. A system that conserves energy cannot make a transition to a definite state and thus cannot make a decision (compute) ([MC80], 9.5).

At any given point in the evolution of a technology, the smallest logic devices must have a definite physical extent, require a certain minimum time to perform their function and dissipate a minimal switching energy when transiting from one state to another. Therefore, side-channel leakage is inherited from the nature of computations and cannot be avoided.

At the current technology evolution step CMOS devices are the most pervasive. The following sections explain power dissipation of a CMOS inverter as a basic logic element. Inverter's power consumption is the cornerstone of all the CMOS side-channel physical leakages.

2.1.1 CMOS Power Dissipation

The CMOS inverter is the atomic element of all CMOS-semiconductor logic cells. An inverter (Fig. 2.1a) consists of an nMOS and a pMOS transistors that switch synchronously. When an input logic level is `1`, a pMOS is open while an nMOS drains the output signal to the ground as shown on Figure 2.1b. The opposite happens when an input logic level is `0`, an nMOS does not conduct while a pMOS connects the output to the V_{dd} line as shown on Figure 2.1c.

An inverter's physical layout is illustrated on Figure 2.2, where V_{cc} is a ground level, V_{dd} is the power supply, V_{in} is an input signal, V_{out} is the inverter's output, T_{PNP} is a parasitic bipolar transistor (p^+/N-well$/p^+$), T_{NPN} is a parasitic bipolar transistor (n^+/P-substrate$/n^+$). The shunting resistors R_{well} and R_{sub} represent the effective resistance from the well tap to the PNP base and the substrate tap to the NPN base respectively. p-type substrate is shown in gray and n-type substrate is shown in red.

As stated at the beginning of this chapter, to perform computations energy must be consumed irreversibly. A CMOS inverter consumes and transforms electrical energy supplied from V_{dd} and V_{in}. The inverter's power dissipation consists of two components: *dynamic* and *static* leakages.

Dynamic Power Dissipation (DPD) occurs when signals change their logic state and transistor energy is drawn from the power supply to charge up internal nodes. A small amount of current also flows from V_{dd} to the ground when the p- and n- channel transistors turn on shortly simultaneously during logic transaction.

Static Power Dissipation (SPD) occurs in a stable logic mode when no transactions are performed. When the semiconductor is powered up it continues to leak a small amount of power at almost all $n-p$ and $p-n$ junctions.

Dynamic Power Dissipation

DPD has attracted most of cryptanalysts' attention as the major cause of side-channel leakage. A state, when an inverter's output is changing, is called *transiting*. When transiting an inverter can be represented as the circuit shown on Figure 2.3 with two switches, load capacitance C_L and node's capacitances $C_P \neq C_N$, associated with the gate's fanout and with the routing wires, as well as the parasitic capacitances.

DPD consists of two components: one is the switching power due to charging and discharging of load capacitances, the other is short-circuit power due to the non-zero rise and fall time of input waveforms.

Short-Circuit Power Dissipation The short-circuit current i_{sc} flows from V_{dd} to the ground when the p- and n- channel transistors turn on shortly at the same time during logic transaction. Short circuit power dissipation accounts for more than 20% of total power dissipation [Gan]. Short circuit current

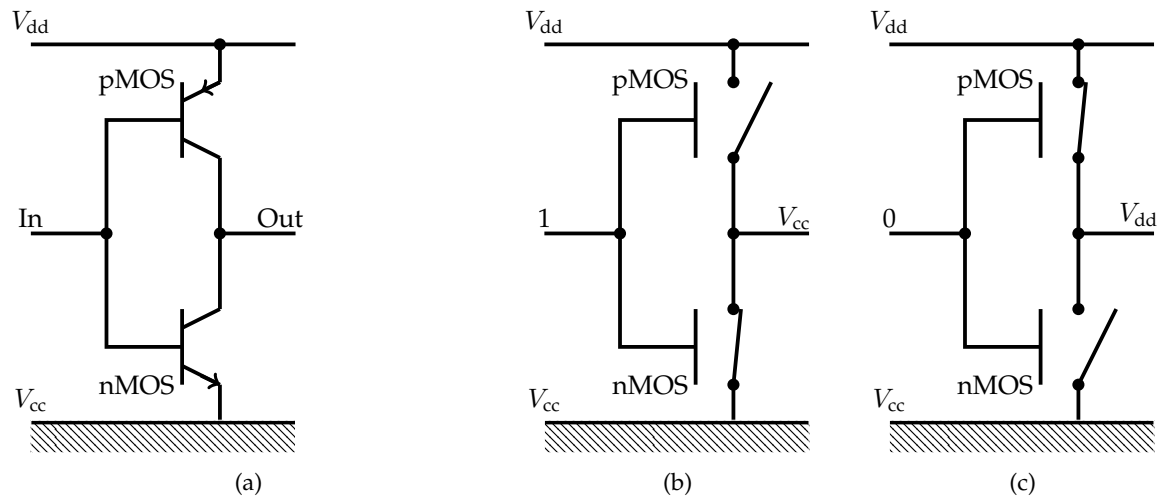


Figure 2.1 – CMOS inverter.

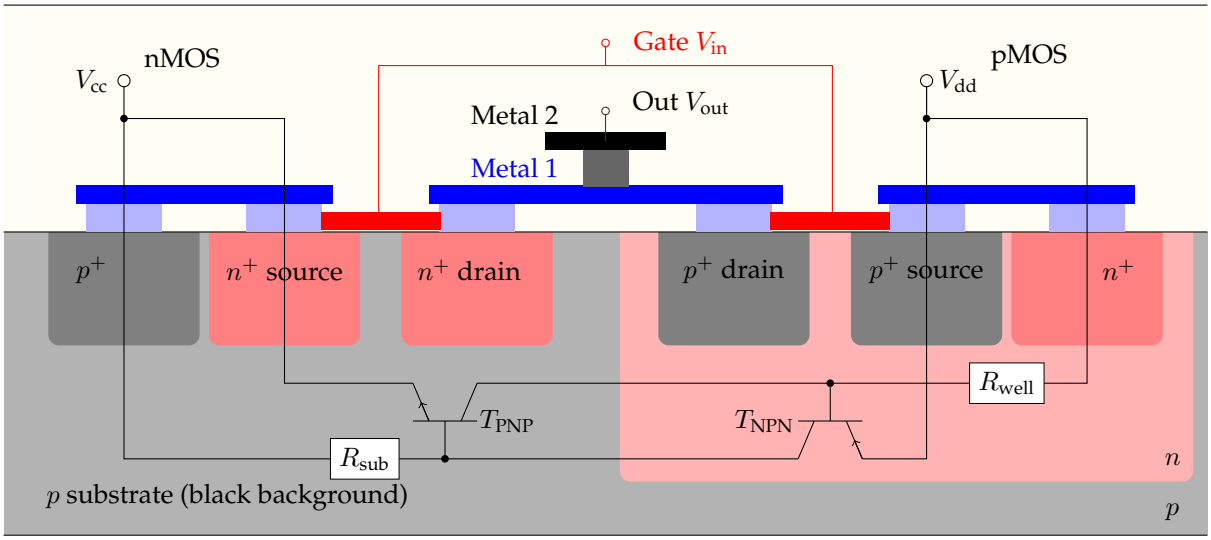


Figure 2.2 – CMOS inverter layout.

depends on the input's transition time, capacitive load, and transistor sizes of the logic gate [VS94]. As clock frequency increases transitions increase and so does short-circuit power dissipation.

Switching Power Dissipation When the logic level changes from `0` to `1` a pMOS transistor cuts the connection between the Out and the V_{cc} so there is almost no current going through C_L. In contrast a switch from `1` to `0` opens the nMOS transistor so an additional current i_L flows through C_L. When there is no transition, *i.e.*, input logic level remains constant, energy is not wasted on capacitive charge. Consequently, the current flowing through the inverter can be computed as shown in Table 2.1.

Dynamic leakage is very dependent on parasitic capacitances. These parasitic capacitances define an upper limit of the clock frequency of a transistor and form unbalanced power consumption during logic level change [MRM00, Uye92].

Static Power Dissipation

Static leakage becomes increasingly important since n- and p- regions are heavily doped. SPD is usually not taken into account for side-channel attacks due to the ulterior dependency between binary data and measurement. However, static power consumption can be used for side-channel attacks as recently shown by [Mor14a].

Table 2.1 – Current flowing through the inverter during logic level change.

Transition	Value of i
$0 \rightarrow 0$	$i \approx 0$
$1 \rightarrow 1$	$i \approx 0$
$0 \rightarrow 1$	$i \approx i_{\text{sc}}$
$1 \rightarrow 0$	$i \approx i_{\text{sc}} + i_L$

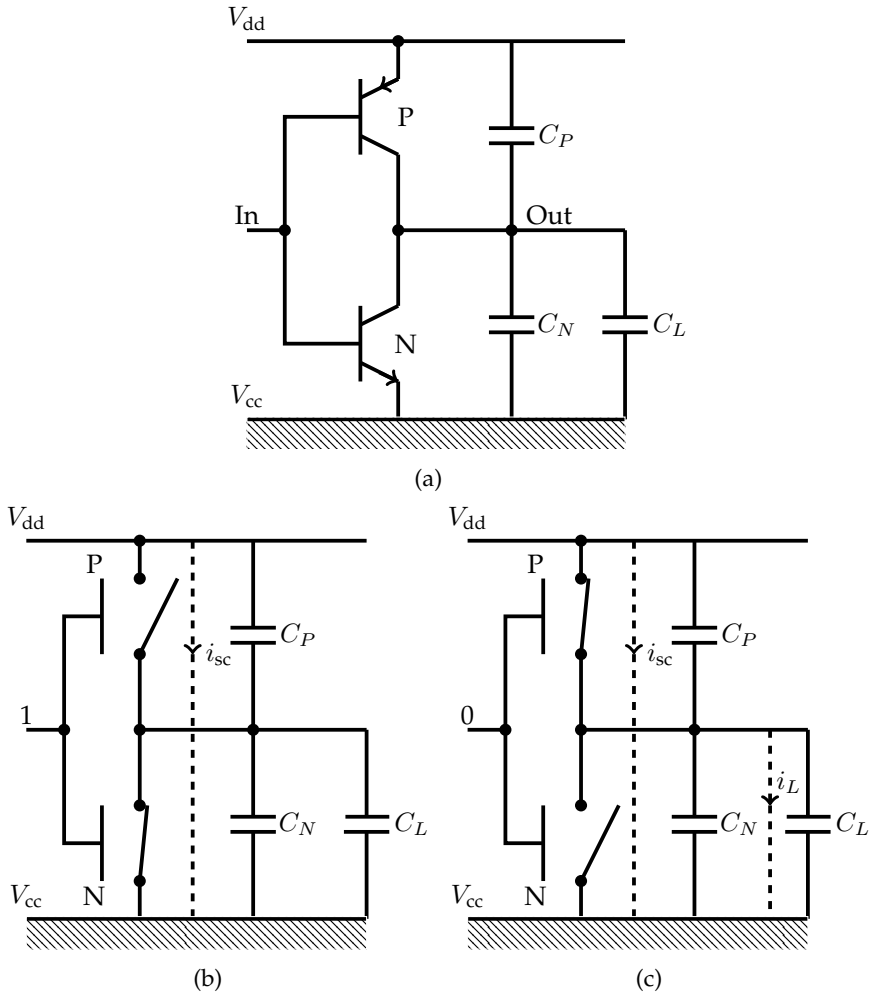


Figure 2.3 – Inverter electrical model during a transition.

The static power of CMOS inverter P_{LEAK} is determined by the leakage current through each transistor:

$$P_{\text{LEAK}} = I_{\text{LEAK}} V_{\text{dd}}$$

where V_{dd} is the supply voltage, and I_{LEAK} is the cumulative leakage current due to all the leakage components.

Six channel leakage mechanisms are illustrated on Fig. 2.4.

- I_1 is the reverse bias $p - n$ junction leakage;
- I_2 is the subthreshold leakage;
- I_3 is the oxide tunneling current:
- I_4 is the gate current due to hot carrier injection;
- I_5 is the Gate Induced Drain Leakage (GIDL);
- I_6 is the channel punch trough current.

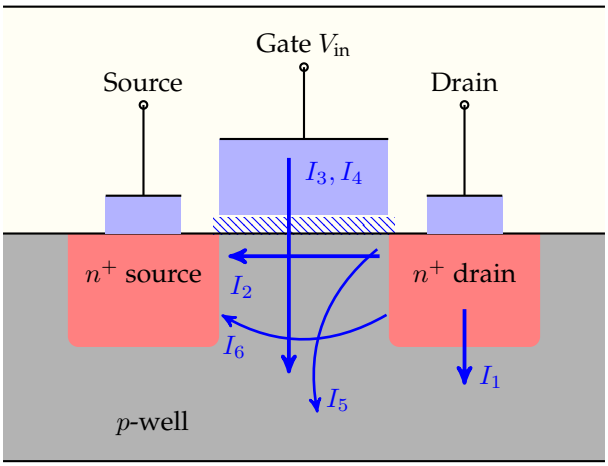


Figure 2.4 – Summary of static leakage currents.

Currents I_2, I_5, I_6 are off-state leakage mechanisms so SPD is data-dependent. I_1 and I_3 occur in both ON and OFF states. I_4 can occur in the off-state, but more typically occurs during the transistor bias states in transition.

p–n Junction Reverse Bias Current I_1 The diode is *reverse-biased* when the n-type semiconductor is connected to the positive voltage (V_{dd}) and the p-type terminal is connected to negative voltage or ground ((V_{cc})). As illustrated on Fig. 2.5 there are several reverse-biased $p-n$ junctions causing leakage in the inverter:

- p-well to n-well
- n^+ to p-well when the input voltage V_{in} is high
- p^+ to n-well when V_{in} is low.

When the diode is reverse-biased, there is a very little flow of current due to minority carriers[1]. When the bias voltage is increased above a certain voltage called *reverse breakdown voltage*, current increases very rapidly [PSSG10].

Subthreshold Leakage I_2 The *threshold voltage*, commonly abbreviated as V_{th}, of a transistor is the minimum gate-to-source voltage differential required to create a conducting path between the transistor's source and drain. The *subthreshold leakage* is the current between the source and the drain when the gate-to-source voltage is below the V_{th}, *i.e.*, when a transistor is OFF. Subthreshold current flows because of the diffusion current of the minority carriers in the channel [DN07]. Subthreshold leakage is the most important contributor to static power in CMOS.

Tunneling Into and Through Gate Oxide I_3 Gate oxides are critical for the scaling of transistor dimensions. Drain current sensitivity is defined by gate oxides. Thinner gate oxides provide better sensitivity, and hence the maximal switching frequency. However, the reduction in the oxide thickness to nanometers causes a current flow between the substrate and the gate through the oxide. This current is caused by carriers tunneling through the insulator [Cha13]. The tunneling of electrons (or holes) from the bulk and source/drain overlap regions through the gate oxide potential barrier into the gate (or vice-versa) is referred to as *gate oxide tunneling effect*.

Hot Carriers Injection from Substrate to Gate Oxide I_4 In a transistor, due to the high electric field near the Si/SiO_2, interface electrons or holes can gain sufficient energy from the electric field to cross the interface and enter the oxide layer. This effect is known as *hot carrier injection*. The injection from Si to

[1] Of the order of nano-Ampere to micro-Ampere

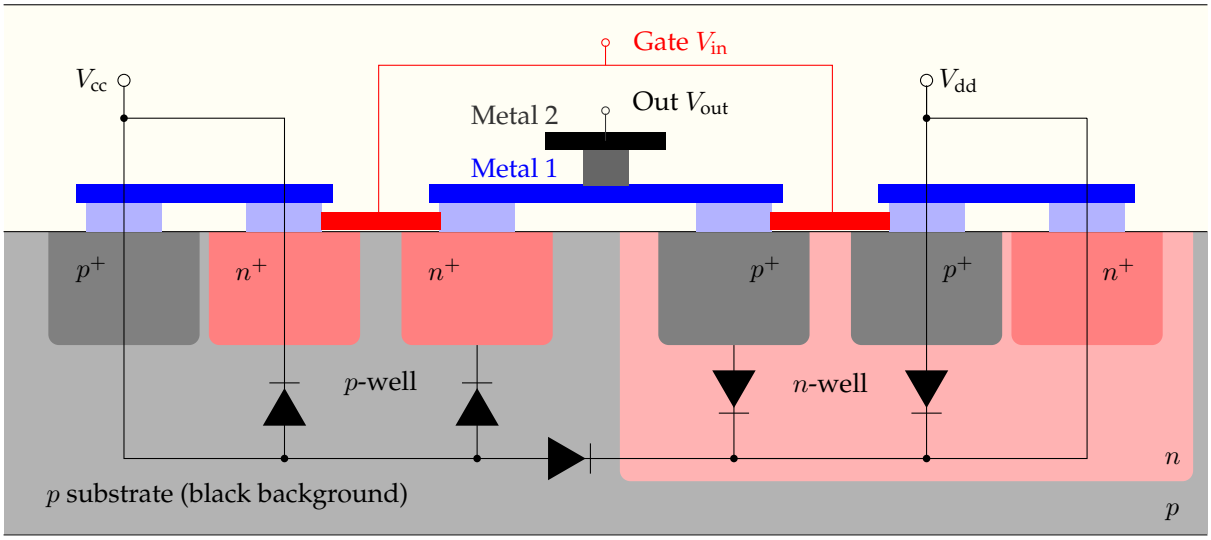


Figure 2.5 – CMOS inverter layout.

SiO_2 is more likely for electrons than holes as electrons have a lower effective mass than that of holes and the barrier height or holes (4.5 eV) is superior than that of electrons (3.1 eV) [TN98].

Gate Induced Drain Leakage I_5 This leakage current component was observed in DRAM trench transistor cells and in EEPROM memory cells [SPS01]. The gate-induced-drain-leakage (GIDL) current is generally known to originate from the difference between the vertical electric fields at the gate and the drain. This leakage current is known to be very sensitive to gate oxide thickness, the drain concentration, the lateral doping gradient, and the applied drain-to-gate voltage.

Punch through I_6 Punch through in a transistor is an extreme case of channel length modulation where the depletion layers around the drain and source regions merge into a single depletion region. The field underneath the gate then becomes strongly dependent on the drain-source voltage, as is the drain current. Punch through causes a rapidly increasing current with increasing drain-source voltage. This effect is undesirable as it increases the output conductance and limits the device's maximum operating voltage [VZ04].

2.1.2 Additional CMOS Side-Channels and Power Transformations

Modern semiconductor devices include billions of transistors and interconnections in which data-dependent current flows [Mor14b]. The data-dependent currents not only dissipate power, but also transform electrical energy:

- Small moving charges produce a variable magnetic field, which itself produces a variable electric field. When data-dependent currents flow through conductive elements, observed electromagnetic activity can be used as a side-channel vector [AARR03,GMO01,QS01]. EM measurements introduce several benefits to the attacker in comparison with power consumption. EM emissions can be measured locally, over a small chipset's area. EM acquisitions can also characterize the direction of register switch from `0`-to-`1` or from `1`-to-`0` [PSQ07].
- In 2005 it was observed that not only signal amplitude, but also power spectrum, can leak secret information [GHT05]. Following the introduction of Differential Frequency Analysis [GTC05], power analysis on frequency domain was investigated in a thread of papers [Luo10,MG11,PGQK09, SDB+10]. Frequency analysis applies Fourier transform to map a time-series into the frequency domain. Since each Fourier point is a linear combination of all other sample points, a spectrum is a direct function of the initial signal amplitude and hence, power spectra can also be used in side-channel attacks. [Luo10] rightly noted that the term Differential Spectral Based Analysis (DSBA) is semantically preferable because DFA does not exploit variations in frequencies, but

differences in spectra. As a matter of fact all time-domain power models and distinguishers remain in principle fully applicable in the frequency domain.

- This thesis shows that, in addition to the signal's amplitude and spectrum, traditionally used for side-channel analysis, *instantaneous frequency* (IF) variations may also leak secret data. To the authors' best knowledge, "pure" frequency leakage has not been considered as a side-channel vector so far. By opposition to the constant frequency used in Fourier Transform IF is understood as local phase differences that express frequency variations. The detailed description of this side-channel vector is given in Chapter 4.
- Another exploitable effect occurs during transistor saturation when the consumed electrical energy is freed during photon emission [Boi04]. When a current flows between the source and the drain, electrons gain energy and accelerate due to the electrical field. The radiative de-excitation of the charge carriers in the pinch-off zone generates photons which are visible in the near-infrared spectral range [DBCR+10]. Photon emission side-channel analysis is not a theoretical attack but a practically applied technique [TDF+14]. A photon emission signal can be measured with an avalanche diode. Photonic emission analysis has a clear advantage when a small surface or an entire semiconductor activity is observed both in time and space [KNSS13].
- Noise is an additional 'exotic' side-channel vector reported in [GST14]. Inverter transactions changes V_{dd}'s power domain which is filtered by an on-board capacitor. A capacitance tries to maintain a stable V_{dd} level by emitting current to the power line and this activity creates vibrations that can be recorded by a sensitive microphone.

CMOS energy transformations cause various physical phenomena which may be better localized and more precisely measured than the electrical power dissipation.

2.2 CMOS Leakage Models

The previous section explained the dependency between logic levels, *i.e.*, binary data, and CMOS physical phenomena. The fact that modern semiconductor devices contain billions of transistors arises reasonable doubts about the feasibility of data extraction; hence this section presents the essentials of side-channel leakage models and the main statistical theorems used in side-channel attacks.

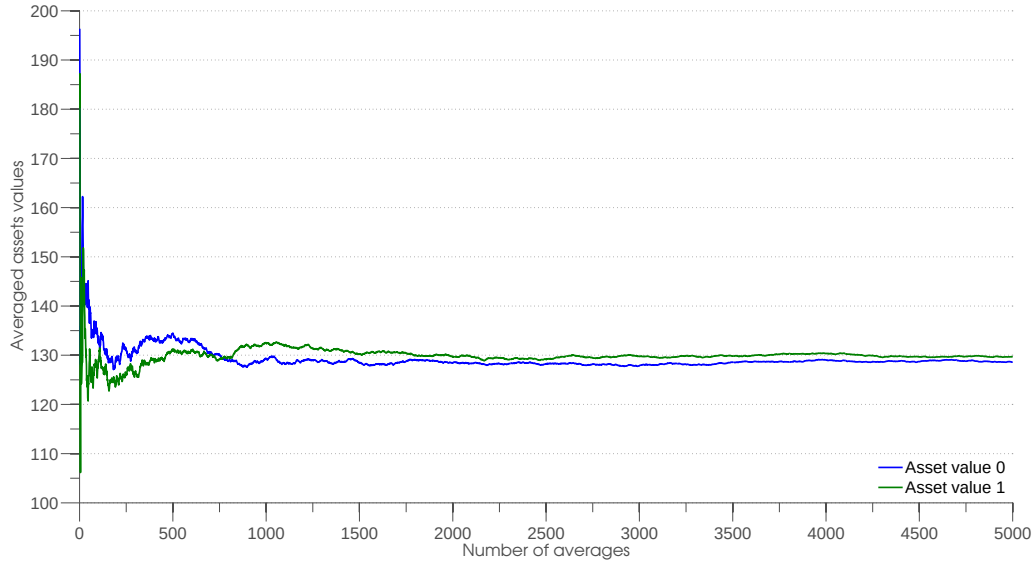
Physical responses, linked to the hardware states, categorize all side-channels models into two groups. The first group contains all physical phenomena where leakage is defined by the current hardware state. This is called the *Hamming weight model*. The second category requires the knowledge of the previous and current hardware states since the difference between `0`-to-`1` and `1`-to-`0` transactions is negligible. This is the *Hamming distance model*. Roughly speaking, the Hamming weight model exploits physical differences between transactions `0`-to-`1` and `1`-to-`0`, while the Hamming distance model exploits differences between transactions and the stable state.

Noise and simultaneous activity contribute a significant part of the final measured current values, which has to be taken into account during side-channel analysis. Side-channel assets, *i.e.*, leakage of a targeted operation, cannot be precisely measured for the following reasons:

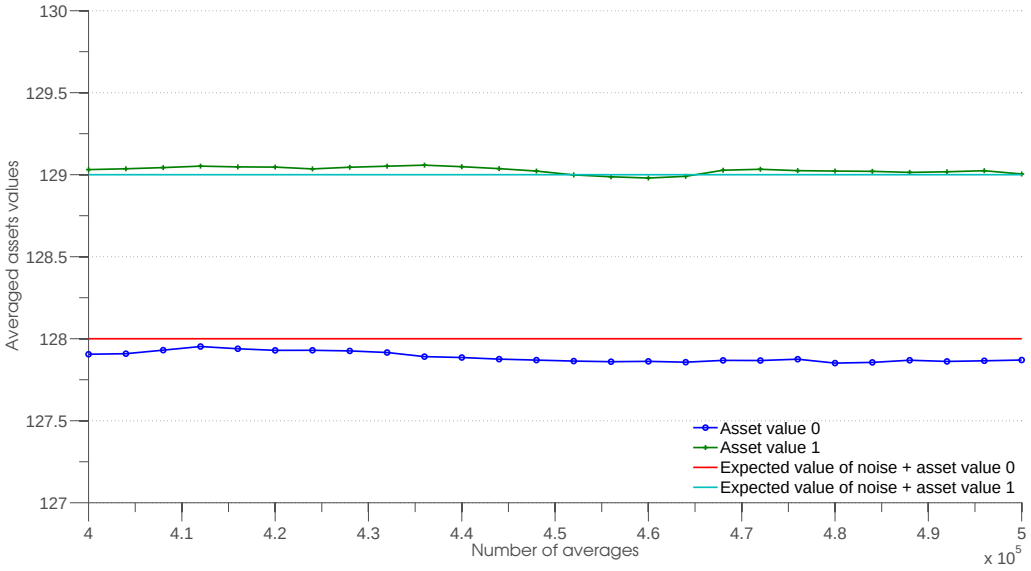
1. Parallel hardware activity.
2. Environmental noise.
3. Measurement noise.
4. Device parameters' variations, such as temperature, clock instability, etc.

Noise cannot be completely eliminated, so side-channel attacks are based on statistical methods and one cornerstone of these methods is a *law of large numbers* [Sen13]. According to the law of large numbers, as the number of identically distributed, randomly generated variables increases, their sample mean (average) approaches their theoretical mean.

The law of large numbers allows to distinguish two values even with a significant amount of noise, as illustrated by the following example. Consider two asset values equal to `0` and `1`. Each time the asset value is measured, Gaussian noise $f(x, \mu, \sigma)$, $\mu = 128, \sigma = 74$ is added to the measurement. Figure 2.6a shows the asset values averaged for 0 to 5,000 trials. Clearly, those values are converging to certain levels



(a) 0 - 5,000 times averaged assets values.



(b) 40,000 - 50,000 times averaged assets values.

Figure 2.6 – Averaged assets values.

that are better illustrated on Figure 2.6b, which represents asset values averaged for 40,000 to 50,000 measurements. According to Figure 2.6b, graphs are converging to the asset values plus the expected noise value, thus they can be easily distinguished.

Consider an encryption algorithm, executed by a hardware where a side-channel leakage can be measured. Hardware is composed of the circuit processing the encryption algorithm and additional blocks used, for example, for communication. A side-channel emanation from the hardware, denoted by $\chi(t), t = 1, ..., N$, is recorded during each encryption. Sample $\chi(t)$ contains leakage from the algorithm's state $\chi_S(t)$, leakage from the rest of the circuit and noise are denoted together as $\theta(t)$.

$$\chi(t) = \chi_S(t) + \theta(t)$$

What information can be gained from the side-channel measurement $\chi(t)$? To answer this question a leakage model has to be introduced.

Table 2.2 – Information that can be obtained with side-channel leakage.

	Time	SPD[2]	DPD[3]	EM[4]	PSD[5]	IF[6]	Noise	PEA[7]	TSA[8]
Hamming weight	✓	✓	✓	✓	✓	✓	✓	✓	✓
Hamming distance	✓	✗	✓	✓	✓	✓	✓	✓	✗
Temporal location	✗	✗	✓	✓	✗	✓	✓	✗	✗
Spatial location	✗	✗	✗	✓	✗	✗	✗	✓	✓
References	[BvdPSY14] [BM06] [Koc96]	[Mor14a]	[BCO04] [KJJ99] [Man03]	[AARR03] [KOP09]	[MG10] [TOT+14]	[KNdAdC14]	[GST14]	[KNSS13] [SNK+12]	[HS14]

Definition 22 [Leakage model function] A leakage model function is a mapping between the processed binary data and the side-channel leakage signal $\chi_S(t)$.

A leakage model function quantifies the deterministic part of the side-channel related to the processed data. A precise leakage model function, which for a given binary data S, outputs exact side-channel leakage value $\chi_S(t)$ is difficult to develop. Instead, most of the models are approximated as a linear dependency between a binary data $S \in \mathbb{F}_{2^n}$ and a leakage function $\chi_S(t)$. Thus for the Hamming weight model this dependency is given by equation 2.1 and for the Hamming distance model by equation 2.2.

$$\chi_S(t) = a_0 + a_1 \mathrm{HW}\left(S_1^{[t]}\right) + a_2 \mathrm{HW}\left(S_2^{[t]}\right) + ... + a_n \mathrm{HW}\left(S_n^{[t]}\right) \tag{2.1}$$

$$\chi_S(t) = a_0 + a_1 \mathrm{HW}\left(S_1^{[t-1]} \oplus S_1^{[t]}\right) + a_2 \mathrm{HW}\left(S_2^{[t-1]} \oplus S_2^{[t]}\right) + ... + a_n \mathrm{HW}\left(S_n^{[t-1]} \oplus S_n^{[t]}\right) \tag{2.2}$$

where $S_i^{[t]}$ is the i-th bit of a current value $S^{[t]}$; $S_i^{[t-1]}$ is the i-th bit of the previous state value $S^{[t-1]}$ that was overwritten with a current value $S^{[t]}$; a_i are the weights.

When the Hamming weight model coefficients $a_i, i \in [1, n]$ are equal to 1 then the maximum information that can be obtained from the leakage $\chi_S(t)$ is the Hamming weight of variable S. When those coefficients form a unique sum for any S then the maximum gained information is the value of S. The same observation applies to the Hamming distance model coefficients.

The best way to define model coefficients is to use profiling methods, such as linear regression analysis [SLP05]. Problems and enhancements of leakage models were discussed in several papers [DPRS11, HSS12]. A practical evaluation of an 8-bit microcontroller leakage model function reported in [ABDM00] concludes that the choice of the model's coefficients is critical for side-channel attacks and may lead to erroneous results in the case of inadequate selection. In some cases an adversary can develop device-specific or leakage-specific models [TOT+14].

Table 2.2 presents information that can be obtained from side-channel leakage. Apart from the binary information, *i.e.*, Hamming weight or Hamming distance, side-channel leakage can characterize time and die spatial location of a targeted hardware. Thus side-channels can be used for reverse engineering [Nov03, TQL+12].

[2]SPD - Static Power Dissipation
[3]DPD - Dynamic Power Dissipation
[4]EM - Electromagnetic Analysis
[5]PSD - Power Spectrum Density
[6]IF - Instantaneous Frequency
[7]PEA - Photonic Emission Analysis
[8]TSA - Temperature Side-Channel

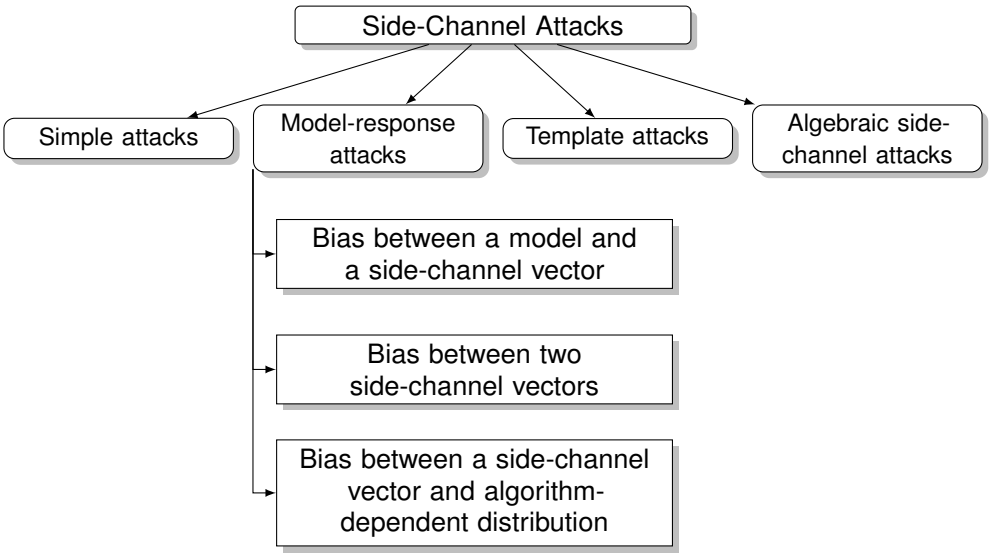


Figure 2.7 – A taxonomy of block cipher side-channel attacks.

Once an adversary can link side-channel leakage to binary data he can build models for the known information and unknown key part and verify them with available physical experiments. The following subsection provides a taxonomy of side-channel attacks.

2.3 A Taxonomy of Side-channel Attacks

Current state-of-the-art side-channel methods can be classified into four groups: *simple*, *model-response*, *template* and *algebraic* side-channel attacks as given by Fig. 2.7. All these attacks perform operations over a *side-channel* vector, *i.e.*, side-channel samples of the same operation taken at the same temporal and spatial location.

2.3.1 Simple Attacks

Simple attacks are based on direct leakage observations. Some cryptographic protocols can be broken by observing the presence or the absence of operations in a power trace [Nov02]. Sometimes cryptographic algorithms can leak Hamming weight information [Man03, SNK+12].

2.3.2 Model-Response Attacks

Model-response attacks compute a statistical bias (response) for each modelled subkey leakage. Model-response attacks belong to the wider category of *distinguishing attacks*. The general definition of distinguishing attacks is a form of cryptanalysis that allows an attacker to distinguish the ciphertexts from random data. This definition can be adapted to side-channel attacks, when an attacker builds data models and verifies them with statistical likelihood estimators, such as difference of means [KJJ99], correlation power analysis [BCO04], or mutual information analysis [GBTP08]. The efficiency of different distinguishers was compared by several authors, *e.g.*, [HRG14, WO11]. Correlation-enhanced collision attacks can be also considered as distinguishing attacks, which verify a collision by cross-correlating power traces [BDGP14, MME10]. Bias computation allows to further subdivide model-response attacks into three groups:

1. Bias computed between a leakage model and a side-channel acquisition, for example Differential Power Analysis, Mutual Information Analysis, *etc.* [BCO04, GBTP08, KJJ99].
2. Bias computed between two side-channel acquisitions, for example Collision Attacks [BK07, SWP03, SLFP04].

Algorithm 1 General first order side-channel attack algorithm

Require:

P_i : set of plaintexts $i = 1, ..., M$;
C_i : set of ciphertexts $i = 1, ..., M$
$\chi^i(t)$: side-channel measurement during encryption $i = 1, ..., M$, $t = 1, ..., N$;

Ensure:

A key value K^*;

1: **for all** $K_j \in \mathbb{F}_{2^q}$ **do**
2: **for** $1 \leq i \leq M$ **do**
3: $L_{(K_j,S_i)} = f(P_i, C_i, K_j)$ ▷ Build a leakage model
4: **end for**
5: **end for**
6: **for all** $K_j \in \mathbb{F}_{2^q}$ **do**
7: **for** $1 \leq t \leq N$ **do**
8: $I_{(K_j,t)} = \rho(L_{(K_j,S_{[1:M]})}, \chi^{[1:M]}(t))$ ▷ Compute the dependency
9: **end for**
10: **end for**
11: $K^* = \text{argmax}_{K_j} \left(\text{argmax}_t(I_{(K_j,t)})\right)$

3. Bias computed between a side-channel vector and an algorithm-dependent distribution. This method was introduced approximately at the same time for side-channel attacks [LDLL14] and fault attacks [KPN14]. A detailed description of this method is given in Chapter 5.

In all cases an attacker has to build a model which includes known information and a secret data. When the model is built correctly the likelihood estimation of the correct secret data shall result in a distinguishable value. The cornerstone advantage of the last method, *i.e.*, of a bias between a side-channel vector and an algorithm-dependent distribution, is that an attacker does not need plaintexts and ciphertexts to build a model. Instead he exploits built-in distributions which are unique for each subkey. The details of this approach are given in Chapter 5.

All model-response attacks are based on distinguishers, noted as $\rho(X, Y)$ in Algorithm 1. A *distinguisher* is an informal name designating a dependency measure between two random variables or two data sets. Three popular distinguishers *difference of means*, *Pearson correlation coefficient*, and *mutual information* are presented in the following subsections.

To describe distinguishers the following notations are required. A cryptographic device sequentially encrypts M known plaintexts P_i $(i = 1, ..., M)$ resulting in M known ciphertexts C_i $(i = 1, ..., M)$. Side-channel information, measured during encryption i, is denoted as $\chi^i(t)$, where $t = 1, ..., N$ indicates the sampling time. A (first order) attack targets a chosen state $S_i \in \mathbb{F}_{2^b}$ that depends both on known data P_i or C_i and on a small part of the key K.

In model-response attacks, where a bias is computed between a model and a side-channel vector, the adversary predicts the side-channel leakage of an intermediate value, referred to as *leakage function* or *selection function*. The prediction for the known data and key guess K_j is denoted by $L_{(K_j,S_i)}$ in accordance with [GDMPV09]. An adversary computes $L_{(K_j,S_i)}$ for all $i = 1, ..., M$ encryptions and all key candidates $K_j \in \mathbb{F}_{2^q}$ and then applies distinguishers to detect a dependency between a model and a real side-channel leakage $\chi^i(t)$. The position of the asset leakage $\chi^i_{S_i}(t)$ in the acquired trace $\chi^i(t)$ is unknown, so all time samples $t = 1, ..., N$ have to be examined. When all the parameters, such as time, spatial location and the key guess, are correct the distinguisher will output a significantly different value (maximal or minimal) from all other key candidates.

Let $\mu_{(K_j,\delta)}(t)$ denote the empirical mean and $\sigma^2_{(K_j,\delta)}(t)$ denote the sample variance of all $\chi^i(t)$ when the leakage function is $L_{(K_j,S_i)} = \delta$.

$M_{(K_j,\delta)} \leq M$ denotes the number of elements in a set for which $L_{(K_j,S_i)} = \delta$.

The general (first order) attack algorithm is described by Algorithm 1.

Difference of Means

The difference of means is the first reported side-channel distinguisher [KJJ99]. The principle idea of the attack is to classify all encryptions into two groups. The first group includes encryptions where the target operation leaks a significant amount of side-channel information $G_{(K_j,\delta)} = \{\chi^i(t) \mid L_{(K_j,S_i)} = \delta\}$ while the second group includes all the encryptions where the target operation leaks less side-channel information $G_{(K_j,\gamma)} = \{\chi^i(t) \mid L_{(K_j,S_i)} = \gamma \neq \delta\}$. After selection the difference between means of the two groups is computed. The resulting curve shall have significant spike for the correct key hypothesis at the time of leakage function $L_{(K_j,S_i)}$, while for the wrong key guess the resulting curve would converge to zero at all sampled points $t = 1, ..., N$.

The remarkable difference between the correct and the wrong key guesses can be explained by the law of large numbers. When the side-channel samples are selected correctly, *i.e.*, for the correct key and time, then samples in both groups form different distributions as illustrated on Fig. 2.8a. Conversely, when the selection is wrong, *i.e.*, for the wrong time sampling or key, then the distributions of samples are close to each other as illustrated on Fig. 2.8b. Formally the difference of means can be written as given in (2.3).

$$D_{K_j}(t) = \mu_{G_{(K_j,\delta)}}(t) - \mu_{G_{(K_j,\gamma)}}(t) \tag{2.3}$$

and the correct key $K^* = \text{argmax}_{K_j} |D_{K_j}(t)|$.

Difference of means can be replaced by the t-test [SC89]:

$$T_{K_j}(t) = \frac{D_{K_j}(t)}{\sqrt{\frac{\sigma^2_{G_{(K_j,\delta)}}(t)}{M_{G_{(K_j,\delta)}}} + \frac{\sigma^2_{G_{(K_j,\gamma)}}(t)}{M_{G_{(K_j,\gamma)}}}}} \tag{2.4}$$

where $\sigma_{G_{(K_j,\delta)}}$ and $\sigma_{G_{(K_j,\gamma)}}$ are groups' standards deviations, $M_{G_{(K_j,\delta)}}$ and $M_{G_{(K_j,\gamma)}}$ are groups' cardinalities.

Pearson Correlation

The difference of means does not take into account all possible leakage function values; hence, a logical development is to group side-channel leakage according to the leakage function $L_{(K_j,S_i)}$ [BCO04]. For example, if side-channel information grows linearly with Hamming weight, then a Pearson Correlation Coefficient can identify the correct key:

$$\rho_{K_j}(t) = \frac{\text{cov}\left(\chi^i(t), L_{(K_j,S_i)}\right)}{\sigma\left(\chi^i(t)\right)\sigma\left(L_{(K_j,S_i)}\right)}, \text{ for } i = 1, ..., M \tag{2.5}$$

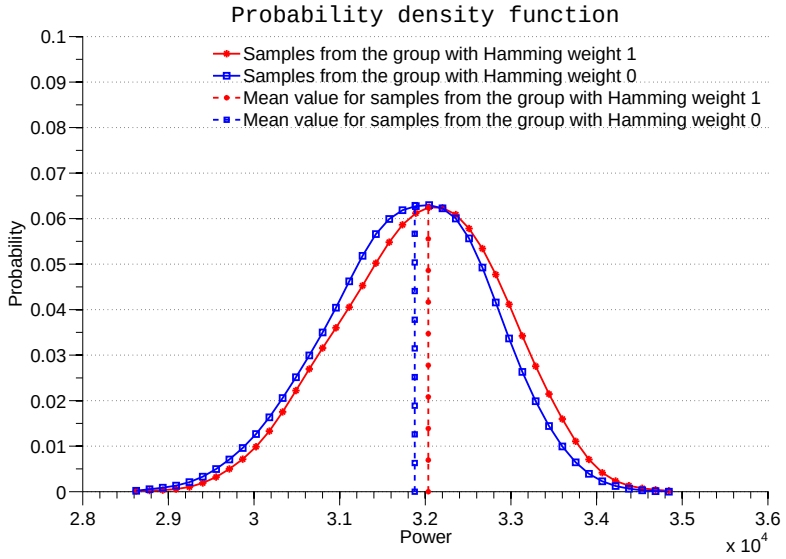
When a key guess is correct then the average of side-channel measurements $\chi^i(t)$ grouped according to the leakage function value $L_{(K_j,S_i)}$ has a linear trend. Conversely, when a key guess is wrong or if sample time is incorrect, the averaged curve is flat as illustrated on Fig. 2.9.

Linear correlation techniques can be only applied when the binary model and side-channel leakage are monotonic or linearly dependent. To address an arbitrary dependency between the binary model and the side-channel leakage a Mutual Information Analysis can be applied.

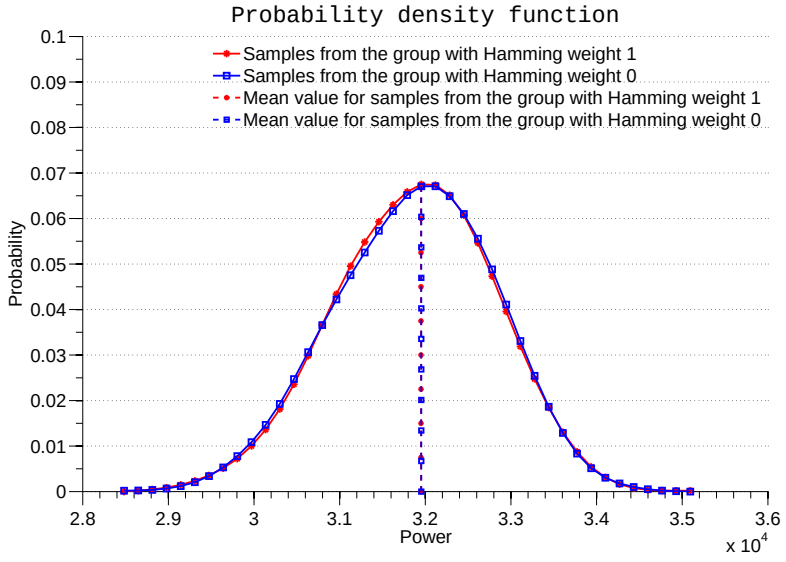
Mutual Information

Mutual information is a measure of variable mutual dependency which can be of any type since this estimator is based on joint distribution $p(X, Y)$:

$$I_{K_j}(t) = \sum_{i\in[1,N]} \sum_{\forall L_{(K_j,S_i)}} p(\chi^i(t), L_{(K_j,S_i)}) \log\left(\frac{p(\chi^i(t), L_{(K_j,S_i)})}{p(\chi^i(t))p(L_{(K_j,S_i)})}\right) \tag{2.6}$$

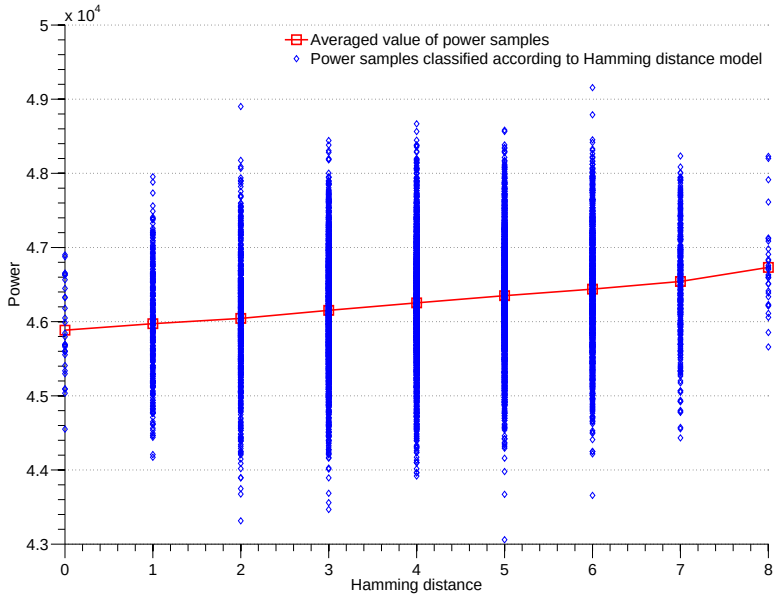


(a) Correct guess

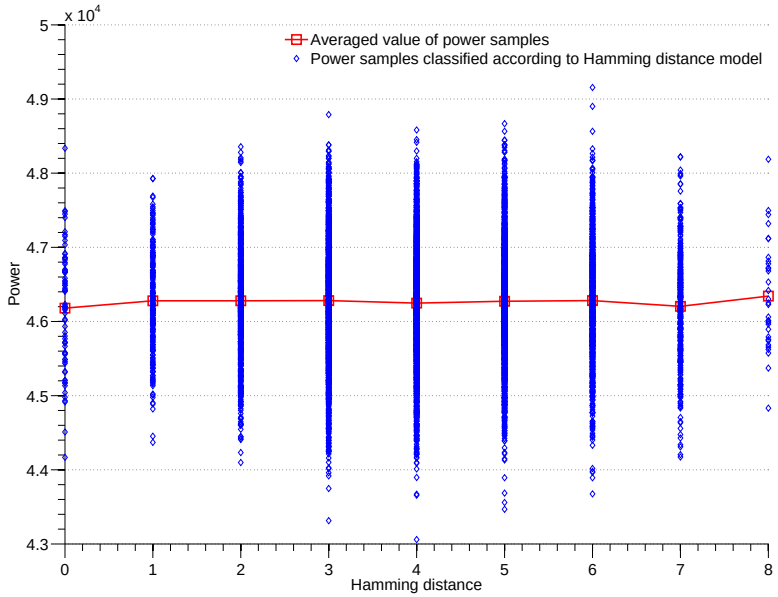


(b) Wrong guess

Figure 2.8 – Difference of means.



(a) Correct guess

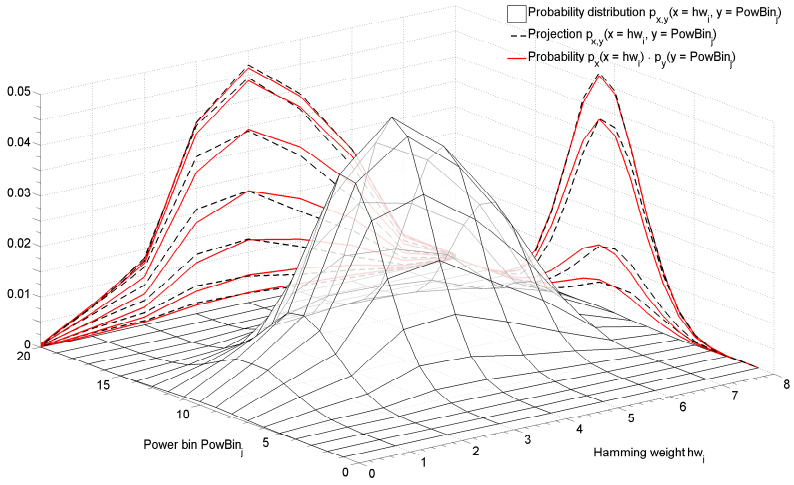


(b) Wrong guess

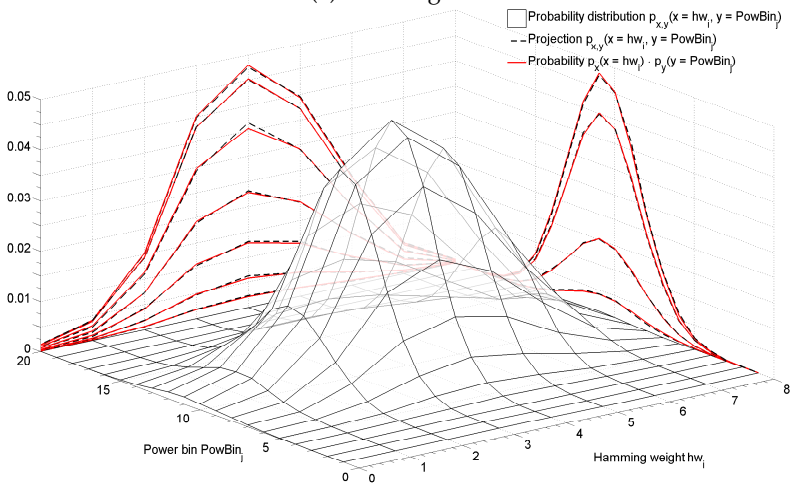
Figure 2.9 – Correlation coefficient.

where $p(x, y)$ is the joint probability distribution of X and Y; $p(x)$ and $p(y)$ are the marginal probability density functions of X and Y respectively.

Mutual information is non-negative and is equal to zero if and only if X and Y are independent random variables. Similarly to the difference of means and to correlation techniques mutual information can distinguish the correct key as illustrated on Fig. 2.10. The main difference can be seen on projections of the joint probability $p(x, y)$ and the marginal probabilities product $p(x)p(y)$. When variables x and y are mutually dependent, the joint probability $p(x, y)$ is different from the marginal probability product $p(x)p(y)$ as illustrated on the $X - Z$ and $Y - Z$ projections of Fig. 2.10a otherwise the joint and marginal probability are the same as on Fig.2.10b.



(a) Correct guess



(b) Wrong guess

Figure 2.10 – Mutual information.

2.3.3 Template Attacks

Template attacks as an evolution of distinguishing methods, attempt to assign a pattern to each key value. The pattern includes the mean value $\mu_{K_j}^{|N|}$ and the noise distribution $\Omega_{K_j}^{|N|}$ [CRR03]. To build a template of each possible key value an attacker needs full control over a device [CK14b] thus limiting the practical applicability of template attacks.

Constructing the template consists in estimating the set of parameters $(\mu_{K_j}^{|N|}, \Omega_{K_j}^{|N|})$ as described by Algorithm 2. The subsequent attack phase consists in acquiring the trace $\chi_{K^*}(t)$ from the target system and identifying the key value K^* using Baye's rule:

Algorithm 2 General template algorithm

Require:

K_j : set of keys $K_j \in \mathbb{F}_{2^q}$;

$\chi^i_{K_j}(t)$: set of side-channel queries for K_j: $t = 1, ..., N,\ i = 1, ..., M_{K_j}$;

Ensure:

A template for each key value $(\mu^{|N|}_{K_j}, \Omega^{|N|}_{K_j})$

1: **for all** $K_j \in \mathbb{F}_{2^q}$ **do**

2: $\mu^{|N|}_{K_j} = \frac{1}{M_{K_j}} \sum_{i=1}^{M_{K_j}} \chi^i_{K_j}(t)$ ▷ Compute the averaged signal

3: $\Theta_{K_j}[i,:] = \begin{pmatrix} \chi^1_{K_j}(t) - \mu^{|N|}_{K_j} \\ \cdots \\ \chi^{M_{K_j}}_{K_j}(t) - \mu^{|N|}_{K_j} \end{pmatrix}$ ▷ Compute the matrix of noise Θ_{K_j}

4: $\Omega^{|N|}_{K_j} = \frac{1}{M_{K_j}} \left[\Theta^T_{K_j} \Theta_{K_j}\right]$ ▷ Construct the noise covariance matrix

5: **end for**

$$K^* = \text{argmax}_{K_j} \left(\frac{1}{\sqrt{(2\pi)^N |\Omega^{|N|}_{K_j}|}} \exp\left(-\frac{1}{2} (\chi_{K^*}(t) - \mu^{|N|}_{K_j})^T \cdot \left(\Omega^{|N|}_{K_j}\right)^{-1} \cdot (\chi_{K^*}(t) - \mu^{|N|}_{K_j}) \right) \right)$$

where $\mu^{|N|}_{K_j}$ and $\chi_{K^*}(t)$ contains N samples, $\left(\Omega^{|N|}_{K_j}\right)^{-1}$ is an inverse noise covariance matrix, $|\Omega^{|N|}_{K_j}|$ is the determinant of the noise covariance matrix.

The computation complexity of Bayes' rule strongly depends on the number of samples N. To simplify and to speed up the computation the number of samples shall be reduced. Only meaningful samples, called *points of interests* (POI), shall remain for template construction.

Depending on the reduction algorithm, template attacks can be classified into several groups. The first group of template constructions assumes visual point selection [CRR03, RO05]. An adversary sums up pairwise differences between all the averaged signals $\mu^{|N|}_{K_1}, ..., \mu^{|N|}_{K_{2^q}}$ and then selects $N_{\text{POI}} \leq N$ points among the highest peaks, forming keys' signatures $\mu^{|N_{\text{POI}}|}_{K_1}, ..., \mu^{|N_{\text{POI}}|}_{K_{2^q}}$.

Another approach is to use Principal Component Analysis (PCA) [APSQ06]. PCA is a statistical approach allowing to identify patterns in data. PCA computes the eigenvalues of a covariance matrix and then uses only the most significant of them, *i.e.*, the highest eigenvalues.

Yet another pattern recognition method is Fisher's Linear Discriminant Analysis (LDA) [CK14a]. LDA is looking for a projection where samples are projected very close to each other. At the same time, the projected means are as far as possible.

After the reduced template creation, given in Algorithm 3, the same Bayes' rule can be computed with the following equation:

$$K^* = \text{argmax}_i \left(\frac{\exp\left(-\frac{1}{2} (\chi_{K^*}(t_{\text{POI}}) - \mu^{|N_{\text{POI}}|}_{K_j})^T \cdot \left(\Omega^{|N_{\text{POI}}|}_{K_j}\right)^{-1} \cdot (\chi_{K^*}(t_{\text{POI}}) - \mu^{|N_{\text{POI}}|}_{K_j}) \right)}{\sqrt{(2\pi)^{N_{\text{POI}}} |\Omega^{|N_{\text{POI}}|}_{K_j}|}} \right)$$

where index t_{POI} implies that only samples at POI positions are used for computation, $\chi_{K^*}(t_{\text{POI}})$ and $\mu^{|N_{\text{POI}}|}_{K_j}$ have N_{POI} samples, $\Omega^{|N_{\text{POI}}|}_{K_j}$ and $\left(\Omega^{|N_{\text{POI}}|}_{K_j}\right)^{-1}$ are reduced to $N_{\text{POI}} \times N_{\text{POI}}$ matrices. Storing and processing the reduced template $\left(\mu^{|N_{\text{POI}}|}_{K_j}, \Omega^{|N_{\text{POI}}|}_{K_j}\right)$ is more efficient that storing and processing the entire measurement template $\left(\mu^{|N|}_{K_j}, \Omega^{|N|}_{K_j}\right)$.

Template attacks have several drawbacks that restrict their experimental applicability:

- Computational artefacts during noise covariance matrix and inverse noise covariance matrix computation.

Algorithm 3 General template algorithm with reduced number of samples

Require:

K_j : set of keys $K_j \in \mathbb{F}_{2^q}$;

$\chi^i_{K_j}(t)$: set of side-channel queries for K_j: $t = 1, ..., N,\ i = 1, ..., M_{K_j}$;

Ensure:

A template for each key value $(\mu^{|N_{\text{POI}}|}_{K_j}, \Omega^{|N_{\text{POI}}|}_{K_j})$

1: **for all** $K_j \in \mathbb{F}_{2^q}$ **do**

2: $\mu^{|N|}_{K_j} = \frac{1}{M_{K_j}} \sum_{i=1}^{M_{K_j}} \chi^i_{K_j}(t)$ ▷ Compute the averaged signal

3: $\mu^{|N_{\text{POI}}|}_{K_j} = \text{POI}\left(\mu^{|N|}_{K_j}\right)$ ▷ Compute the reduced averaged signal

4: $\Theta_{K_j}[i,:] = \begin{pmatrix} \chi^1_{K_j}(t_{\text{POI}}) - \mu^{|N_{\text{POI}}|}_{K_j} \\ \cdots \\ \chi^{M_{K_j}}_{K_j}(t_{\text{POI}}) - \mu^{|N_{\text{POI}}|}_{K_j} \end{pmatrix}$ ▷ Compute the matrix of noise Θ_{K_j}

5: $\Omega^{|N_{\text{POI}}|}_{K_j} = \frac{1}{N_{\text{POI}}} \left[\Theta^T_{K_j} \Theta_{K_j}\right]$ ▷ Construct the noise covariance matrix

6: **end for**

- LDA is highly dependent on the condition of equal covariances.
- Template attacks require a full access to an educative device (a sample to experiment with).
- Template attacks can output several candidates when side-channel information reveal only Hamming weight.

Template attacks are usually applied to extract key information, for example Hamming weight. At the same time template attacks can be used to characterize intermediate key-related information which will later reveal the key value itself. One of these methods, called *algebraic side-channel attacks*, is based on overdefined systems of equations created with Key Expansion procedures [CP02].

2.3.4 Algebraic Side-Channel Attacks

Algebraic attacks apply side-channel information to solve a system of overdefined equations [CP02]. Algebraic side-channels, however, require a profiling phase, as template attacks. Using side-channels an attacker can recover simple targets, for example key Hamming weight, and then recover the key value [RS10]. Therefore algebraic side-channel attacks inherit all the disadvantages of template attacks, such as the need of an educative device.

2.4 Side-Channel Countermeasures

Chip manufacturers have been developing SCA countermeasures since Kocher's first publication [KJJ99]. Countermeasures for all SCA physical vectors can be classified into several global groups, showed on Fig. 2.11.

1. *Leakage reduction* curtails physical dependency of binary data on the logic level. The examples of leakage reduction techniques are the following:
   (a) Power attacks countermeasures:
      - Balanced power consumption [MSQL05, TAV02, TV04]
      - Trapezoidal power-clock voltage [JJKB16, KM08]
      - On-chip capacitor network [MM14, Sha03]
   (b) Timing and cache attack countermeasures:
      - Constant time execution [KD09, JRE12]

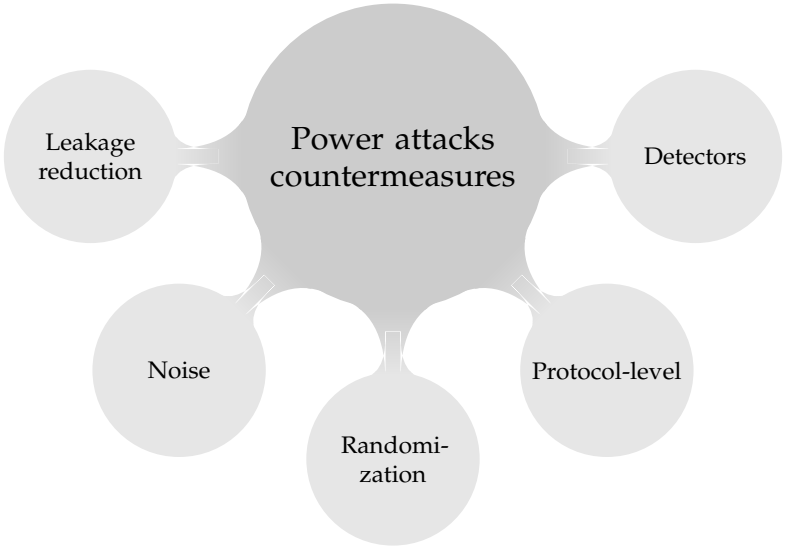


Figure 2.11 – SCA countermeasures mind-map.

- Restriction on the use of some instructions [YF13]

2. *Noise* consists in injecting an unpredictable component to the measured side-channel trace.
   (a) Typical amplitude noise countermeasure against power analysis are:
   - Randomly pre-charged data lines and registers before use [APRV07, RCN02]
   - Randomly varying supply voltage [KGS+11, YWV+05]

   (b) Typical temporal noise countermeasures against power, time and photon analysis include:
   - Asynchronous circuits [BSR06, FML+03]
   - The clock presenting instabilities in duty cycle or frequency [BLOW10, GOK+05]

   (c) Combination of amplitude and time countermeasures:
   - Dummy instructions [AG01]
3. *Randomization* changes secret data representation so that sensitive data is no longer processed in a plain form. This countermeasure restricts model construction, so it can be applied for all the SCAs. Randomization is typically implemented in the following ways:
   - Masking and blinding combine the secret data with a random data that can be removed from the final result [ITT02, TSG03]
   - Homomorphism uses the arithmetic properties of public key cryptosystems to compute their result in function in one of many ways [Cha83]
4. *Protocol-level solutions* consist in using leaky algorithms in a secure way. Protocol-level solutions are typically implemented in the following ways:
   - Limit the number of encryptions per key [MPR+11]
   - Update keys continuously [Koc05]
5. Standard-cell-based *detectors* can recognize close EM probe location [HHM+14]. Various sensors can detect package opening, close EM probe location or other circuit modifications used to improve SCA.

CHAPTER 3

FAULT ATTACKS

Summary

The hardware processing can be transiently or permanently modified if the device is forced to operate out of normal physical conditions. Device tampering can corrupt memory contents, change instruction flow, or cause other precise modifications unveiling algorithm's data. Computational errors are used by fault attack to reveal cryptographic keys and other secret algorithm data.

A successful attack on a device requires two steps explained in this chapter: *faults injection* and *faults exploration*. Section 3.1 describes fault injection techniques and explains fault models. Fault attacks against SPN ciphers are explained in Section 3.2. Fault attack countermeasures are given in Section 3.4.

3.1 Fault Attack Explanation

Any device requires a certain physical parameters to operate within a normal range. Flawless computation can not be guaranteed if those conditions are not met. A properly adjusted perturbation of a system can modify intermediate registers' data, change the program counter, skip instructions or create other exact faults. By inducing specific errors, an attacker can probe the algorithm's internals by comparing of correct and faulty results.

The idea of physical fault injection into hardware is similar to software fuzzing [SGA07], which tries to raise an exception or find a security breach by feeding software with invalid or unexpected inputs. In the case of fault attacks an opponent applies physical stress (input) which can corrupt normal a computational routine to:

- Modify intermediate data in the algorithm.
- Access an additional hardware/software functionality.
- Skip or switch-off protection mechanisms.

By injecting exact faults an attacker can, for example, break RSA with a single faulty result [BDL97] or use faults as a side-channel leakage source [LSG+10]. The following subsections overview fault attacks and explain basic key recovery techniques.

In this thesis only transient faults are considered, *i.e.*, faults that affect only one or several instructions during one execution. Permanent faults, *i.e.*, circuit modifications that persists after reset, can also be used. However, those faults are less applicable since they require expensive equipment, such as focused ion beams [HNT+13].

3.1.1 Timing Constrains in Digital ICs

In synchronous sequential logic a device's state changes only at discrete times defined by a clock signal. This is necessary to synchronize internal operations. A device embeds many hardware blocks that use the same clock, for example, each CPU operation can be considered as a separate digital circuit with its critical path, *i.e.*, the path between an input and an output defining the maximum signal propagation delay tolerated by the circuit. In some cases, the propagation delay in 45-nm CMOS technology can be reduced by temperature increase [KK06].

Registers latch data at rising clock edges. The computed data travels between registers and gets modified by the intermediate combinatorial logic blocks between two such edges. Fig. 3.1 illustrates main time constraints of a digital system.

1. *Propagation delay* d_p is the time needed to propagate the data through combinatorial logic.
2. d_{CLK2Q} the delay between the clock rising edge and the actual update of a register's output.
3. t_{skew} the skew or slight phase difference that may exist between the clock signals at the clock inputs of two different registers
4. $t_{\text{set-up}}$ a duration for which a D flip-flop input must be stable before the clock's edge to ensure reliable operation.

Therefore, all the timing constrains contribute to the limitation of the circuit's maximal operating frequency (nominal circuit period). Indeed, to ensure proper circuit operation, the clock period T_{clock} must be greater than the following sum [ADN+10]:

$$T_{\text{clock}} > d_{\text{CLK2Q}} + d_p + t_{\text{set-up}} - t_{\text{skew}} \tag{3.1}$$

Any new data entering a register can be considered as a result of a combinatorial calculation involving several registers outputting previous (input) data as illustrated on Fig. 3.1. The transformation of the previous registers' output into the next register's input bit takes a determined delay. This delay depends on the logic performed as well as on the data transiting through the logic. In addition, propagation time varies with circuit temperature and power supply voltage.

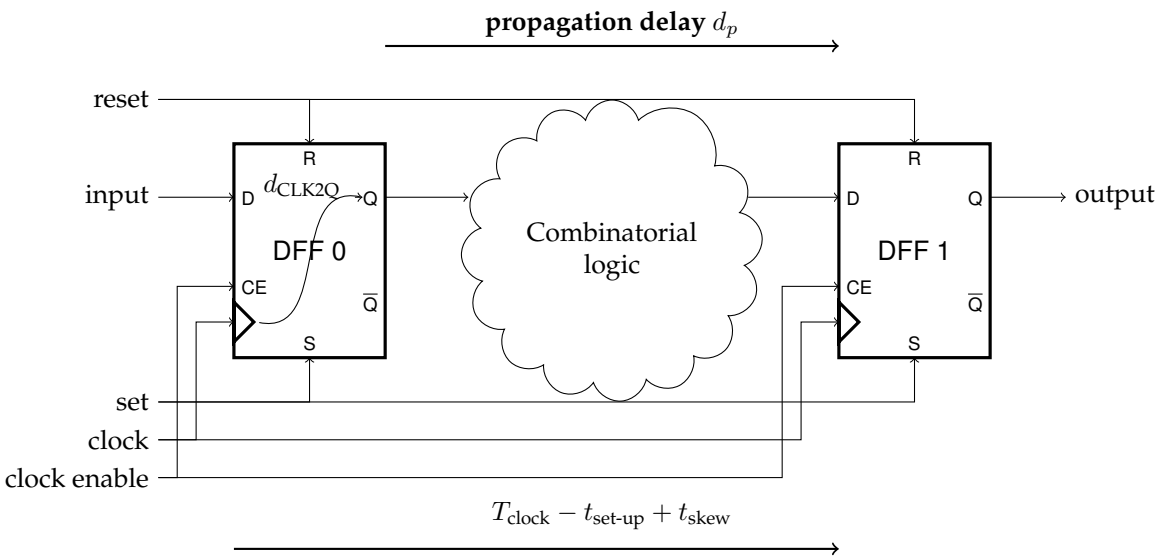


Figure 3.1 – Synchronous representation of digital ICs.

3.1.2 Fault Injection Techniques

The first paper to ever deal with the issue of fault injection was not a paper on the use of electronics in the space environment, but a paper assessing scaling trends in terrestrial microelectronics [WM62]. In this paper, the authors forecast the eventual occurrence of *single event upset* (SEU)[1] in microelectronics due to cosmic rays and further predicted that the minimal volume of semiconductor devices would be limited to about 10μm per side due to these upsets. In fact, the authors wrote in 1962 that *already at the present time the essential part of semiconductor devices, the active region, is close to the minimum size possible* [WM62].

Later on, the first confirmed report of cosmic-ray-induced upsets in space was reported in [BSH75] and the occurrence of soft errors in terrestrial microelectronics described shortly after the first observations of SEU in space [MW79].

Subsequent research included studying and simulating the effects of cosmic rays on semiconductors [FS04, DM03]. Cosmic radiation are very weak at ground level due to the Earth's atmosphere, but their effect becomes more pronounced in the upper atmosphere and outer space. This problem is further compounded by the fact that the more RAM a computer has the higher is the chance of a fault occurring. This provoked extensive research by organizations such as NASA and Boeing. Most work on fault resistance was motivated by this sensitivity to charged particles. Considerable engineering endeavours were devoted to the 'hardening' of electronic devices designed to operate in harsh environments. This has mainly been done using simulators to model circuits and study the effect of randomly induced faults. Various fault induction methods have since been discovered but all have in common similar effects on chips.

A first practical application of fault injection was proposed by [BDL97]. The authors presented a secret recovery method from a random computational error caused by a fault injection. Since then the field of fault attacks has been widely studied by many institutions and researchers [JT12].

Overclocking

Overclocking consists in decreasing the clock period T_{clock} (or, put differently, increasing clock frequency). A sudden clock frequency increase can create a situation in which the bits on the critical path did not have time to stabilize, so the flip-flops were not updated with a new clock cycle causing faulty data to be latched instead [FT09]. This led several authors to use overclocking as fault injection means [ADN+10, SGD08].

[1] SEU is a change of state caused by one single ionizing particle (ions, electrons, photons...) striking a sensitive node in a microelectronic device, such as in a microprocessor, semiconductor memory, or power transistors.

A decreased clock period can potentially affect logical paths whose propagation times exceed the decreased clock period minus the set-up time. From the attacker's perspective the ability to control precisely the clock period is crucial for inducing faults with precision. Note that temperature and power supply changes may also be used to exert such control [KK06].

Although many manufacturers claim to implement high-frequency detectors in their clock signal-processing logic, these circuits are often only simple filters that do not detect sudden short cycles [KK99].

Power Glitches

Power supply is another obvious external signal that can be maliciously tampered. Power glitches are often used to induce faults in ICs by a sudden and short negative change on power supply [ABF+03, BGV11]. The underlying fault injection mechanism was deeply investigated in a recent thesis [Zus14]. The main conclusion of the aforementioned thesis is that negative glitches increase the right part of equation (3.1), namely, register setup time $t_{\text{set-up}}$. The assumption that power glitches induce fault by violating the target's timing constraints is also supported by other researchers [BECN+06, TS09].

Another interesting conclusion is that positive glitches cause errors due to the *overshoot effect*. This effect creates a power drop immediately after the positive glitch; hence, this drop can cause an error. According to the aforementioned thesis [Zus14] positive glitches have the same effect as negative ones.

Successful power glitch injection depends on many parameters that can be controlled by an attacker: nominal power supply; glitch depth, width, falling and rising edges; delay (in respect with the targeted operation), IC temperature, and others. Several publications present optimized techniques to combine and fine-tune glitch fault injection [CPB+14].

Optical Attacks

Lasers can imitate the effect of charged particles [GHJ92]. Laser radiation can ionize an IC's semiconductor regions if its photon's energy exceeds the semiconductor's band gap [SA03]. Laser radiation with 1.06 μm wavelength can penetrate a semiconductor layer to a significant depth. With thinned and polished silicon, laser fault injection can achieve outstanding results, including one bit modifications [ADM+10]. Apart from infrared laser, visible light, green laser and blue lasers can be used to inject faults from the rear side of a chip.

There are two primary methods by which ionizing radiation releases charges in a semiconductor device: direct ionization by the incident particle itself and ionization by secondary particles created by nuclear reactions between the incident particle and the struck device. Both mechanisms can lead to IC malfunction [DM03].

- *Direct Ionization*: When an energetic charged particle passes through a semiconductor material it frees electron-hole pairs along its path as it loses energy. When all of its energy is lost, the particle comes to rest in the semiconductor, having traveled a total path length referred to as the particle's *range*. Direct ionization is the primary charge deposition mechanism for upsets caused by heavy ions, where a heavy ion is defined as any ion with atomic number greater than or equal to two (*i.e.*, particles other than protons, electrons, neutrons, or pions) [DM03]. Lighter particles such as protons do not usually produce enough charge by direct ionization to cause upsets in memory cells.
- *Indirect Ionization*: Although direct ionization by light particles does not usually produce enough charge to cause upsets, this does not mean that we can ignore these particles. Protons and neutrons can both produce significant upset rates due to indirect mechanisms. As a high-energy proton or neutron enters the semiconductor lattice it may undergo an inelastic collision with a target nucleus.

Electromagnetic Fault Injection

The first targeted electromagnetic (EM) fault injection techniques against a semiconductor device dates back to 1995 [KFA+95]. The authors used a special probe *in order to direct the faults to specific parts of*

the computer board, such the CPU buses. Back at that time EM perturbation was secondary with respect to laser and heavy-ion fault injection techniques.

EM perturbations can locally tamper the circuit's power consumption which makes EM attacks more powerful than power or clock glitches. EM fault injection hardware is less expensive than laser setups [PTL+11]. There are two main EM fault injection techniques:

1. EM harmonic, *i.e.*, a stable sinusoidal signal generated at a given frequency, can introduce a parasitic signal biasing or inject additional power into a block [AOP+09, MM09]. This fault injection techniques is commonly applied against analogue blocks, such as ring oscillators [BBA+12].
2. EM pulse [OGSM16, OGST+15, PTL+11] that takes advantage of multiple metal loops presented in modern semiconductor devices. The sudden EM variation yields a current in a metal loop thus affecting signal propagation. This effect is similar to that of a power glitch; however, EM effect is localized. Digital circuits are clocked; hence, to disturb their behaviour, EM pulses are preferable to the injection of faults during a specific clock cycle in a controllable way.

3.1.3 Basic Fault Properties

A fault attack description must specify a *fault model* [GT04]. The model clarifies the attacker's capabilities and must include parameters such as a type of error, timing, location, precision of the fault injection, the number of faults, etc. The latter is called *an order of the attack*, a term suggested in [DGRS09]; a first-order attack assumes that an attacker is capable of inducing only one error during the algorithm's execution, while second-order models assume that injecting more than one error is possible.

An adversary is capable of inducing more than one fault during a single algorithm run. Generic modelling of multi fault attacks becomes much more difficult as theoretically any set of parameter values for the first fault may be combined with any set of parameter values for the second fault. Of course, one can impose limitations, *e.g.*, the same type of error can be induced twice (the hardware settings producing the perturbation are difficult to modify between two faults). Even under these limitations, attacker's capabilities become more powerful. For example, an adversary may now induce faults in both a variable and a procedure testing this variable, thus thwarting many countermeasures designed to withstand single fault attacks.

Fault Location

Fault injection can be performed on different circuit parts, such as general registers, the program counter, instruction decoders, etc. The resulting effect is mostly considered on registers and instructions, without a detailed understanding as of where the actual fault was induced.

Register fault injection assumes two main parameters:

1. The number of modified bits:
   - *bit*: only one bit is affected
   - *byte*: an entire byte is modified
   - *multiple-bytes*: two or more bytes are modified
2. A direction of bits flipping:
   - *stuck-at* or *unidirectional* faults: bits can be modified only in one direction:
     - zeros can be set to ones, but ones cannot reset to zeros;
     - ones can be flipped to zeros, but zeros cannot be changed to 1.
   - *complement faults*: all the bits are complemented to previous value
   - *pseudo-random*: when the number and the value of bits affected by the fault depends on chip's state and physical impact. This fault is typical for clock glitching, when bits are not completely updated and keep the previous value.

- *random*: when the number and value of bits depend on the physical impact only. This distribution is typical to laser fault injection.

Fault injection into CPU, *i.e.*, instruction fault injection, has the following effects:

1. *Instructions skipping*: the number of instructions skipped with one fault injection.
2. *Legit random instruction modification*: the modified opcode can be correctly processed.
3. *Controlled instruction modification*: the modified opcode becomes one of the following:
   - The contents of the register in the corrupted opcode is modified to another value.
   - The jump is done in another location.

Generally, software implementations feature a significantly higher number of vulnerabilities, since not only registers, but also instructions can be modified. This is why hardware implementations are preferable for cryptography.

Fault Timing

The moment of fault injection must be carefully chosen. Most block ciphers can be attacked with faults at their very early or very late rounds. Intermediate rounds cannot be easily attacked.

Attacks succeed with a certain probability. Usually, an attack is not guaranteed to be successful. Therefore fault effects as well as control timing might require a probability or even a distribution to be defined. For example, some physical attacks may have higher probability of resetting bit than of setting bit [Pai99]. No control over the location implies that a specific location is expected to be hit with a probability 1/(number of locations) if a uniform distribution is assumed, etc.

3.2 Differential Fault Analysis Against SPNs

DFA requires a pair of correct and faulty ciphertexts that are the result of the same plaintext encryption [BS03, Gir05a, DLV03]. Since two encryptions perform identically up to the fault injection point, the two ciphertexts can be considered as the outputs of a reduced-round block cipher where the inputs are unknown but have a small difference [JT12]. Analyzing the propagation of this difference (called differential) over the small number of rounds, an attacker can gain key information involved in these rounds.

When the same plaintext cannot be encrypted twice, a ciphertext-based attack remains practical as shown by [FJLT13]. A bias introduced at the input of an S-box can be used to distinguish the correct key from other key candidates. Because the S-box is a pseudorandom permutation an input's entropy computed from all the faulty results with a wrong key guess would be indistinguishable from the entropy of a uniformly distributed variable. The input's entropy computed for the correct key candidate shall be different due to the introduced bias. The multiple bit-reset fault model considered in [FJLT13] is a typical fault that can introduce a bias.

While DFA uses a fault injected at the last SPN rounds, CFA exploits errors at the beginning of encryption [Hem04, BK06]. CFA looks for a collision between genuine and faulty encryptions of plaintexts P and $\tilde{P}$ respectively. Since the encryptions perform differently up to the point when the fault compensates both state values, the two plaintexts could be considered as the inputs to a reduced-round block cipher that outputs a predictable differential after several rounds.

FAs against SPNs require either full control over the cipher's input and/or the opportunity to inspect the encryption's result. Restricting the attacker's access to the plaintext and the ciphertext is considered as good countermeasure for systems where the secret data can not be easily modified, for example a shared root key used in UMTS [NN06]. However, these countermeasures cannot protect the system against all FAs.

3.2.1 Ciphertext-Based Attacks

This section recalls ciphertext-based attacks against SPN ciphers, described in Section 1.3.1.

Attack on Round N_r

Consider a fault δ introduced before the S-box transformation in the last round N_r. Since all the SPN operations are reversible the correct $S^{[N_r]}$ and faulty $\delta \oplus S^{[N_r]}$ states can be computed as follows:

$$\begin{aligned} S^{[N_r]} &= \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r]}} (C) \\ \delta_{K^{[N_r]}} \oplus S^{[N_r]} &= \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r]}} (\tilde{C}) \end{aligned} \tag{3.2}$$

Equations (3.2) can be considered as a round-reduced cipher where two known inputs $C \neq \tilde{C}$ produce an output with a predictable differential $\delta_{K^{[N_r]}}$. For a given key candidate $K_j \in \mathbb{F}_{2^n}$ and ciphertext pairs $(\tilde{C}_i, C_i)$ we have:

$$\delta_{(K_j,i)} = \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K_j} (\tilde{C}_i) \oplus \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K_j} (C_i) \tag{3.3}$$

The introduced error $\delta_{(K^{[N_r]},i)}$ is assumed to be the outcome of a non uniformly distributed variable, *i.e.*, $\mathrm{H}(\delta_{(K^{[N_r]},:)}) = h < b$. Given the nonlinearity of the S-box the corresponding set of errors $\delta_{(K_j \neq K^{[N_r]},:)}$ (or simply $\delta_{(K_j \neq K^{[N_r]})}$) computed for all the pairs $(C_i, \tilde{C}_i)$ with a wrong key shall be uniformly distributed [SLIO12]. The set of errors $\delta_{(K_j = K^{[N_r]},:)}$ (or $\delta_{(K_j = K^{[N_r]})}$) computed with the correct key shall be nonuniformly distributed; hence, an error entropy can be used for key selection.

$$\lim_{i \to +\infty} \mathrm{H}(\delta_{K_j}) = \begin{cases} b & \text{if} \quad K_j \neq K^{[N_r]} \\ h & \text{if} \quad K_j = K^{[N_r]} \end{cases} \tag{3.4}$$

The general entropy approach is described in [LRD+12], while a special case when $\mathrm{HW}(\delta_{K_j}) = 1$ is described in [Gir05a].

When the same plaintext cannot be encrypted twice, ciphertext-based attacks still apply. In that case the fault has to corrupt the input's uniformity so the entropy would be smaller for the correct key guess as stated by [FJLT13]. The fault that can corrupt the input's uniformity is given by `AND` and `OR` fault models:

$$\begin{aligned} \tilde{S}^{[N_r]} &= \delta \wedge S^{[N_r]} \\ \tilde{S}^{[N_r]} &= \delta \vee S^{[N_r]} \end{aligned}$$

Attack on Round $N_r - 1$

Another generally considered ciphertext-based attack exploits computational errors before the last $\mathbf{D}^{[N_r-1]}$ permutation at round $N_r - 1$. Both correct $S^{[N_r-1]}$ and faulty $\delta \oplus S^{[N_r-1]}$ states can be computed from the known ciphertexts:

$$\begin{aligned} S^{[N_r-1]} &= \left(\mathbf{D}^{[N_r-1]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r-1]}} \circ \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r]}} (C) \\ \delta \oplus S^{[N_r-1]} &= \left(\mathbf{D}^{[N_r-1]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r-1]}} \circ \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r]}} (\tilde{C}) \end{aligned} \tag{3.5}$$

Note that the error space is a subset of $(\mathbb{F}_{2^b})^{\mathcal{B}_{\mathrm{HW}(\delta)}(\mathbf{D})}$, where $\mathcal{B}_{\mathrm{HW}(\delta)}(\mathbf{D})$ is a $\mathrm{HW}(\delta)$-branch number, introduced in Section 1.2. The differential δ_i can be written as a function of $K^{|N_r-1|}, K^{|N_r|}, C_i, \tilde{C}_i$ as shown by equation (3.6).

$$\begin{aligned} \delta_i &= \left(\mathbf{D}^{[N_r-1]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r-1]}} \circ \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r]}} (\tilde{C}_i) \oplus \\ &\quad \left(\mathbf{D}^{[N_r-1]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r-1]}} \circ \Big\|_{\ell=1}^{m} \left(\mathbf{S}_\ell^{[N_r]}\right)^{-1} \circ \mathbf{A}^{-1} \mid_{K^{[N_r]}} (C_i) \end{aligned} \tag{3.6}$$

To recover the correct key, the entropy of $\delta_{(K_j=K^{[N_r]})}$ obtained for the correct key has to be at least distinguishable from the entropy of the variable uniformly distributed over $(\mathbb{F}_{2^b})^{\mathcal{B}_{\mathrm{HW}(\delta)}(\mathbf{D})}$. Since equation (3.6) exploits both round keys $K^{|N_r-1|}$, $K^{|N_r|}$ the search key space has to be squared to $(\mathbb{F}_{2^b})^{2\cdot\mathcal{B}_{\mathrm{HW}(\delta)}(\mathbf{D})}$.

The expression (3.6) can be simplified if the key mixing operation is a bit-wise exclusive or (XOR) between a round key and a state $\mathbf{A}\mid_K(S) : K \oplus S$ and permutation $\mathbf{D}$ is linear with respect to XOR:

$$\delta_i = \left(\mathbf{D}^{[N_r-1]}\right)^{-1}\left(\bigparallel_{\ell=1}^{m}\left(\mathbf{S}_\ell^{[N_r]}\right)^{-1}\circ\mathbf{A}^{-1}\mid_{K^{N_r}}(\tilde{C}_i)\oplus\bigparallel_{\ell=1}^{m}\left(\mathbf{S}_\ell^{[N_r]}\right)^{-1}\circ\mathbf{A}^{-1}\mid_{K^{N_r}}(C_i)\right) \tag{3.7}$$

In this case the candidate key space is reduced to $(\mathbb{F}_{2^b})^{\mathcal{B}_{\mathrm{HW}(\delta)}(\mathbf{D})}$.

The attack against AES described by equation (3.7) can be applied if an error has entropy smaller than 32. The fault when up to three out of four bytes of MIXCOLUMNS's input are modified is described in [Gir05a, DLV03, PQ03], 4 bytes modification is presented in [MSS06, Muk09].

3.2.2 Plaintext-Based Attacks

Plaintext-based attacks, namely CFA, can be used when a cipher's output cannot be directly accessed while still remaining comparable to previous encryption results.

Attack on the First Round

Similar to the previously described last round attack in Section 3.2.1 we consider a fault δ introduced after S-box transformation at the first SPN round. The correct $S^{[1]}$ and faulty $\tilde{S}^{[1]}$ states can be computed as follows:

$$\begin{aligned} S^{[1]} &= \bigparallel_{\ell=1}^{m}\mathbf{S}_\ell^{[1]}\circ\mathbf{A}\mid_{K^{[0]}}(P) \\ \tilde{S}^{[1]} &= \bigparallel_{\ell=1}^{m}\mathbf{S}_\ell^{[1]}\circ\mathbf{A}\mid_{K^{[0]}}(\tilde{P}) \end{aligned} \tag{3.8}$$

Equation (3.8) can be considered as a round-reduced cipher where two known inputs $P \neq \tilde{P}$ produce an output with a predictable differential $\delta_{K^{[0]}}$. An injected error $\delta_{K^{[0]}}$ modified the state $\tilde{S}^{[1]}$ so that:

$$\begin{aligned} S^{[1]} &= \tilde{S}^{[1]} \oplus \delta_{K^{[0]}} \\ \delta_{K^{[0]}} &= S^{[1]} \oplus \tilde{S}^{[1]} \end{aligned}$$

$$\delta_{K^{[0]}} = \bigparallel_{\ell=1}^{m}\mathbf{S}_\ell^{[1]}\circ\mathbf{A}\mid_{K^{[0]}}(P)\oplus\bigparallel_{\ell=1}^{m}\mathbf{S}_\ell^{[1]}\circ\mathbf{A}\mid_{K^{[0]}}(\tilde{P}) \tag{3.9}$$

For a given key byte candidate K_j and byte pairs $(\tilde{P}_i, P_i)$ we have:

$$\delta_{(K_j,i)} = \bigparallel_{\ell=1}^{m}\mathbf{S}_\ell^{[1]}\circ\mathbf{A}\mid_{K_j}(\tilde{P}_i)\oplus\bigparallel_{\ell=1}^{m}\mathbf{S}_\ell^{[1]}\circ\mathbf{A}\mid_{K_j}(P_i)$$

The errors $\delta_{(K^{[0]},i)}$ are assumed to be nonuniformly distributed, *i.e.*, $\mathrm{H}(\delta_{(K^{[0]})}) = h < b$. Given the properties of S-box the corresponding set of errors $\delta_{(K_j\neq K^{[0]},\forall i)}$ (or $\delta_{(K_j\neq K^{[0]})}$) computed with a wrong key for all the encryptions of $(P_i, \tilde{P}_i)$ resulted in ciphertext collision shall be uniformly distributed. The set of errors $\delta_{(K_j=K^{[0]},,\forall i)}$ (or $\delta_{(K_j=K^{[0]})}$) computed with the correct key shall converge to nonuniform distribution. Therefore the error entropy can be used to distinguish the correct key candidate:

$$\lim_{i\to+\infty}\mathrm{H}(\delta_{(K_j)}) = \begin{cases} b & \text{if} \quad K_j \neq K^{[0]} \\ h & \text{if} \quad K_j = K^{[0]} \end{cases} \tag{3.10}$$

A special case when $\mathrm{HW}(\delta_{(K^{[0]})}) = 1$ is described in [BK06].

To the author's best knowledge both the general entropy case [LRD+12] and faulty ciphertexts only attack [FJLT13] have not been adapted to the first SPN round yet. A comparison of equations (3.2) and (3.8) shows that this adaptation is indeed possible as suggested by [JT12].

3.3 Fault Attack Against CRT-RSA

The pioneering CRT-RSA attack was published in [BL96]. The attack is based on several assumptions:

- The RSA implementation is using Chinese Reminder Theorem.
- The attacker can introduce the error either in s_p or s_q (exclusive).
- The attacker can record the faulty signature.
- The attacker knows either the correct signature or the initial message m.

Assume that the final signature is computed by Gauss's recombination method (1.7)[2]:

$$s = \mathrm{CRT}(s_p, s_q) = \left(s_p q(q^{-1} \bmod p)\right) + \left(s_q p(p^{-1} \bmod q)\right) \bmod N$$

Suppose that the error appeared in computation of s_p, and the faulty result is:

$$s' = \mathrm{CRT}(s'_p, s_q) = \left(s'_p q(q^{-1} \bmod p)\right) + \left(s_q p(p^{-1} \bmod q)\right) \bmod N$$

The difference between signatures can be computed as follows:

$$\begin{aligned}\Delta = s - s' &= \left(s_p q(q^{-1} \bmod p)\right) + \left(s_q p(p^{-1} \bmod q)\right) - \left(s'_p q(q^{-1} \bmod p)\right) - \left(s_q p(p^{-1} \bmod q)\right) \\ &= \left(s_p q(q^{-1} \bmod p)\right) - \left(s'_p q(q^{-1} \bmod p)\right) \\ &= (s_p - s'_p) q(q^{-1} \bmod p) \bmod N\end{aligned}$$

Therefore, the greatest common divisor (GCD) between Δ and N is equal to q:

$$\mathrm{GCD}(N, \Delta) = \mathrm{GCD}(pq, (s_p - s'_p) q(q^{-1} \bmod p)) = q$$

Arjen Lenstra observed that the fault attack against CRT-RSA can be performed with the initial message m [Len96]. If the error is induced during the s_p computation then for the correct s and faulty s' signatures the following relationships are true:

$$\begin{aligned}s^e &= (s')^e \bmod q \\ s^e &\neq (s')^e \bmod p\end{aligned}$$

The difference between $(s')^e - m$ can be easily obtained:

$$\begin{aligned}(s')^e - m &= (s'_p)^e q(q^{-1} \bmod p) + (s'_q)^e p(p^{-1} \bmod q) - m \\ &= (s'_p)^e q(q^{-1} \bmod p) + (s'_q)^e p(p^{-1} \bmod q) - (s_p)^e q(q^{-1} \bmod p) - (s_q)^e p(p^{-1} \bmod q) \\ &= (s'_p)^e q(q^{-1} \bmod p) - (s_p)^e q(q^{-1} \bmod p) \\ &= \left((s'_p)^e - (s_p)^e\right) q(q^{-1} \bmod p)\end{aligned}$$

Therefore, the GCD between the modulus N and the difference $(s')^e - m$ is equal to q:

$$\mathrm{GCD}(N, (s')^e - m) = \mathrm{GCD}(pq, \left((s'_p)^e - (s_p)^e\right) q(q^{-1} \bmod p)) = q$$

[2]Fault attack against Garner's recombination method is similar.

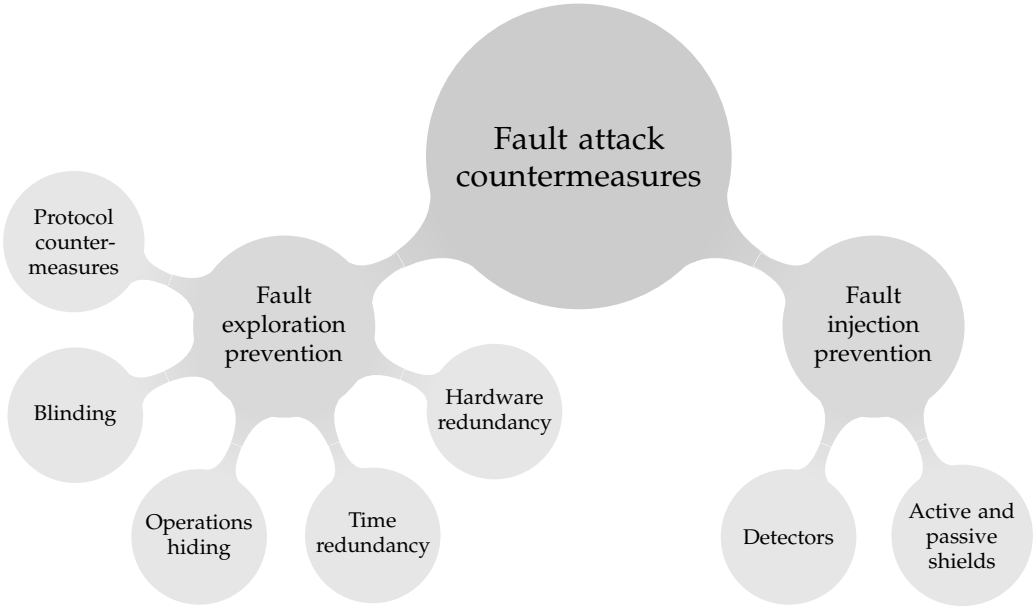


Figure 3.2 – Fault attack countermeasures mind-map.

3.4 Fault Attack Countermeasures

Fault attacks are a real industrial security concern: to pass certifications, such as FIPS 140-1 levels 3 and 4 [PUB99], device manufacturers must prove that their products can resists attacks. Various countermeasures are being developed to that end. Since there exist no generic countermeasures which can prevent all attacks, the combination of different techniques is required to achieve a sufficient security level.

Countermeasures deployment come at a price, so they are chosen to provide a good tradeoff between hardware, performance and security level. In practice some deployed countermeasure may facilitate an attack as shown in the Chapter 6.

Fault countermeasures can be subdivided into two groups: *fault injection prevention* and *fault exploration prevention*.

1. *Fault injection prevention* make a device robust to physical stress:
   (a) *Active and passive shields* are the layers covering sensitive semiconductor parts that make blocks inaccessible to fault injection [ABCS06, TMA+02, MRL+06, CDG+14]. Active shields have data passing through them. If the active shield connections are disconnected or modified the chip will not operate anymore. Passive shields are opaque materials that cover a part of or an entire device thus preventing optical fault injection or probing attacks.
   (b) *Detectors* aim at preventing a specific fault injection method, such as an abrupt laser radiation or a voltage glitch [ISYT13, ZDT+14, VSK13, BTL13].
2. *Fault exploration prevention*
   (a) *Hardware redundancy*
      - Error detection and correction techniques [MSY06] are efficient tools to check data integrity.
      - Duplicating computation in space (hardware redundancy) is in general more secure that duplicating computation in time (time redundancy) [MSY06]. One fault injection device can break time redundancy protection if two identical faults are injected with different delays. Hardware redundancy requires synchronous work of two independent fault injection devices.
   (b) *Time redundancy*

- One approach is to verify the initial data by applying additional computations (different from the original encryption). After signing a message with an RSA private key, for example, verify the signature value with the corresponding public key. A similar method can be used for (rounds of) block ciphers [KWMK02].
- Another approach is to perform twice the same computation. The double-data-rate mechanism proposed by [MVL07] performs the second computation without affecting throughput.
- Precomputed code signatures protects the integrity of the program flow [MS08].

(c) *Operations hiding* includes dummy instructions, execution randomization, code obfuscation, bus and memory encryption, glue logic and other techniques [CK09, NM14].

(d) *Blinding* method consists in "infecting" computations , *i.e.*, redundant computations which get interwoven into output values in case of an error. Infecting countermeasures corrupt the faulty output to a degree when no analysis can be performed [BHT09, YKLM03].

(e) *Protocol countermeasures* are the protection means preventing an attacker from collecting enough information from the target system, for example, several stages of key derivation [EABK14], re-keying when a new key is generated for each encryption [MSGR10].

CHAPTER 4

INSTANTANEOUS FREQUENCY ANALYSIS

Summary

This chapter describes a signal characteristic, called *instantaneous frequency*, that supplements power trace parameters, *e.g.*, power amplitude and power spectrum, applied in side-channel analysis. By opposition to the constant frequency used in Fourier Transform instantaneous frequency is understood as local phase differences to express frequency variations.

These variations enable attacks because they depend on the processed binary data. The relationship between binary data and frequency comes from the fact that higher power drops take more time to converge power back to the nominal value. Instantaneous frequency analysis does not present specific benefits when applied to unprotected designs where CPA and CSBA yields better results. However, when instantaneous frequency is used as a side-channel vector, the effect of amplitude modification can be discarded.

4.1 Motivation

As CMOS state transition energy is essentially proportional to the number of switched bits, DPD is the most popular side-channel attack vector. Because transiting also requires time, transition time and processed data might be also related.

Historically, timing attacks were developed to extract secrets from software algorithms [Koc96] while hardware algorithms were usually assumed to run in constant time and hence be immune to timing attacks. The constant hardware execution time assumption is supported by the fact that usual block-cipher hardware implementations require an identical number of clock cycles to process any data. This chapter shows that this intuition is not always true, *i.e.*, two different inputs may require distinct processing time and can hence be distinguishable.

Energy consumed during each clock cycle creates a waveform in the power domain. A duty cycle, *i.e.*, the time during which the power wave is not equal to its nominal value, can be considered as the execution time of a hardware implemented algorithm. As shown later the duty cycle may depend on the processed data. Fourier transform can not determine local duty cycles since frequency is defined for the sine or cosine function spanning the whole data length with constant period and amplitude. However,recent techniques described in this chapter that can detect local frequencies and hence determine wave duty cycle.

Dynamic Voltage Scrambling (DVS) is a particular side-channel countermeasure that triggers random power supply changes aiming to decorrelate the signal's amplitude from the processed data [BZ07, KGS+11]. While DVS degrades DPA's and DSBA's performances, nothing prevents the existence of more subtle side-channel attacks exploiting DVS-resistant die-hard information present in the signal. This chapter successfully exhibits and exploits such DVS-resistant information.

Contribution. This chapter shows that, in addition to the signal's amplitude and spectrum, traditionally used for side-channel analysis, instantaneous frequency variations may also leak secret data. To the authors' best knowledge, "pure" frequency leakage has not been considered as a side-channel vector so far. Hence a re-assessment of several countermeasures, especially, these based on amplitude alterations, seems in order. As an example this chapter examines DVS, which makes AES implementation impervious to power and spectrum attacks while leaving it vulnerable to Correlation Instantaneous Frequency Analysis (CIFA), a new attack described in this chapter.

Organization. This chapter is organized as follows. Section 4.2 turns a signal processing algorithm called *Hilbert Huang Transform* (HHT) into an attack process. Section 4.3 illustrates an HHT performed on a simulated register switch and real power signal. Section 4.3 also motivates the exploration of instantaneous frequency as a side-channel carrier. Section 4.4 compares the cryptanalytic effectiveness of Correlation Instantaneous Frequency Analysis, Correlation Power Analysis and Correlation Spectrum Based Analysis on an unprotected AES FPGA implementation and on AES FPGA power traces with a simulated DVS. Section 4.5 concludes the chapter.

4.2 The Hilbert Huang Transform

The HHT represents the analysed signal in the time-frequency domain by combining the *Empirical Mode Decomposition* (EMD) with the *Discrete Hilbert Transform* (DHT).

DHT is a classical linear operator that transforms a signal $u(1), \ldots, u(N)$ into a time series $H_u(1), \ldots, H_u(N)$[1] as follows:

$$H_u(t) = \frac{2}{\pi} \sum_{k \neq t \bmod 2} \frac{u(k)}{t-k} \tag{4.1}$$

DHT can be used to derive an *analytical representation* $u_a(1), \ldots, u_a(N)$ of the real-valued signal $u(t)$:

$$u_a(t) = u(t) + iH_u(t) \text{ for } 1 \leq t \leq N \tag{4.2}$$

Equation (4.2) can be rewritten in polar coordinates as

$$u_a(t) = a(t)e^{i\phi(t)} \tag{4.3}$$

[1] Time series notation $H_u(t)$ should not be confused with the entropy notation $\mathrm{H}(X)$.

where

$$a(t) = \sqrt{(u^2(t) + H_u^2(t))} \text{ and } \phi(t) = \arctan\left(\frac{H_u(t)}{u(t)}\right) \tag{4.4}$$

represent the *instantaneous amplitude* and the *instantaneous phase* of the analytical signal, respectively.

The *phase change rate* $w(t)$ defined in equation (4.5) can be interpreted as an *instantaneous frequency* (IF):

$$w(t) = \phi'(t) = \frac{d}{dt}\phi(t) \tag{4.5}$$

For a real-valued time-series the definition of $w(t)$ becomes:

$$w(t) = \phi(t) - \phi(t-1) \tag{4.6}$$

The derivative must be well defined since physically there can be only one instantaneous frequency value $w(t)$ at any given time t. This is insured by the *narrow band condition*: the signal's frequency must be uniform [KM10]. Further, the physical meaningfulness of DHT's output is closely related to the input's fitness into a narrow frequency band [Boa92]. However, we wish to work with non-stationary signals having more than one frequency. This is achieved by de-composing these signals into several components, called *Intrinsic Mode Functions*, such that each component has nearly the same frequency.

Definition 23 [Intrinsic Mode Function] An Intrinsic Mode Function (IMF) is a function satisfying the following conditions:

1. the number of extrema and the number of zero crossings in the considered data set must be either equal or differ by at most one;
2. the mean value of the curve specified as a sum of the envelope defined by the local maxima and the envelope defined by the local minima is zero.

4.2.1 First Step: Empirical Mode Decomposition.

EMD, the HHT's first step, is a systematic way of extracting IMFs from a signal.

EMD involves approximation with splines. By Definition 23, EMD uses local maxima and minima separately. All the local signal's maxima are connected by a cubic spline to define an upper envelope. The same procedure is repeated for the local minima to yield a lower envelope. The first EMD component $h_{1,0}(t)$ is obtained by subtraction from $u(t)$ the envelopes' mean $m_{1,0}(t)$ (see Fig. 4.1):

$$h_{1,0}(t) = u(t) - m_{1,0}(t) \tag{4.7}$$

Ideally, $h_{1,0}(t)$ should be an IMF. In reality this is not always the case and EMD has to be applied to $h_{1,0}(t)$ as well:

$$h_{1,1}(t) = h_{1,0}(t) - m_{1,1}(t) \tag{4.8}$$

EMD is iterated k times, until an IMF $h_{1,k}(t)$ is reached, that is

$$h_{1,k}(t) = h_{1,k-1}(t) - m_{1,k}(t) \tag{4.9}$$

Then, $h_{1,k}(t)$ is defined as the first IMF component $c_1(t)$.

$$c_1(t) \stackrel{\text{def}}{=} h_{1,k}(t) \tag{4.10}$$

Next, the IMF component $c_1(t)$ is removed from $u(t)$

$$r_1(t) = u(t) - c_1(t) \tag{4.11}$$

and the procedure is iterated on all the subsequent residues, until the residue $r_n(t)$ becomes a monotonic function from which no further IMFs can be extracted.

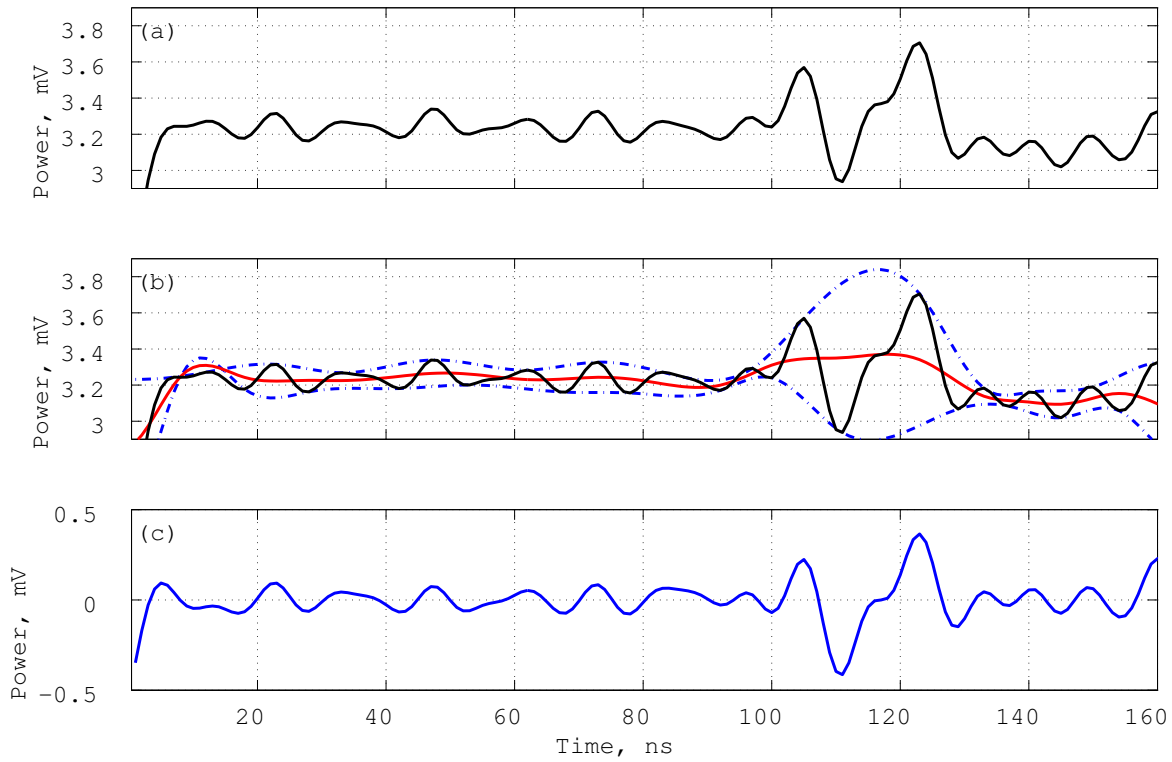


Figure 4.1 – Illustration of the EMD: (a) is the original signal $u(t)$; (b) $u(t)$ in thin solid black line, upper and lower envelopes are dot-dashed with their mean $m_{i,j}$ in thick solid red line; (c) shows the difference between $u(t)$ and the envelope's mean.

$$\begin{cases} r_2(t) = r_1(t) - c_2(t) \\ \dots \\ r_n(t) = r_{n-1}(t) - c_n(t) \end{cases} \tag{4.12}$$

Finally, the initial signal $u(t)$ is re-written as a sum:

$$u(t) = \sum_{j=1}^{n} c_j(t) + r_n(t), \quad \text{for} \quad 1 \leq t \leq N \tag{4.13}$$

where, $c_j(t)$ are IMFs and $r_n(t)$ is a constant or a monotonic residue.

4.2.2 Second Step: Representation.

The second HHT step is the representation of the initial signal in the time-frequency domain. All components $c_j(t)$, $j \in [1, n]$ obtained during the first step are transformed into analytical functions $c_j(t) + iH_{c_j}(t)$, allowing the computation of instantaneous frequencies by formula (4.6). The final transform $U(t, w)$ of $u(t)$ is:

$$U(t, w) = \sum_{j=1}^{n} a_j(t) \exp\left(i \sum_{\ell=1}^{t} w_j(\ell)\right) \tag{4.14}$$

where $j \in [1, n]$ is indexing components, $t \in [1, N]$ represents time and:

$a_j(t) = \sqrt{c_j^2(t) + H_{c_j}^2(t)}$ is the *instantaneous amplitude*;

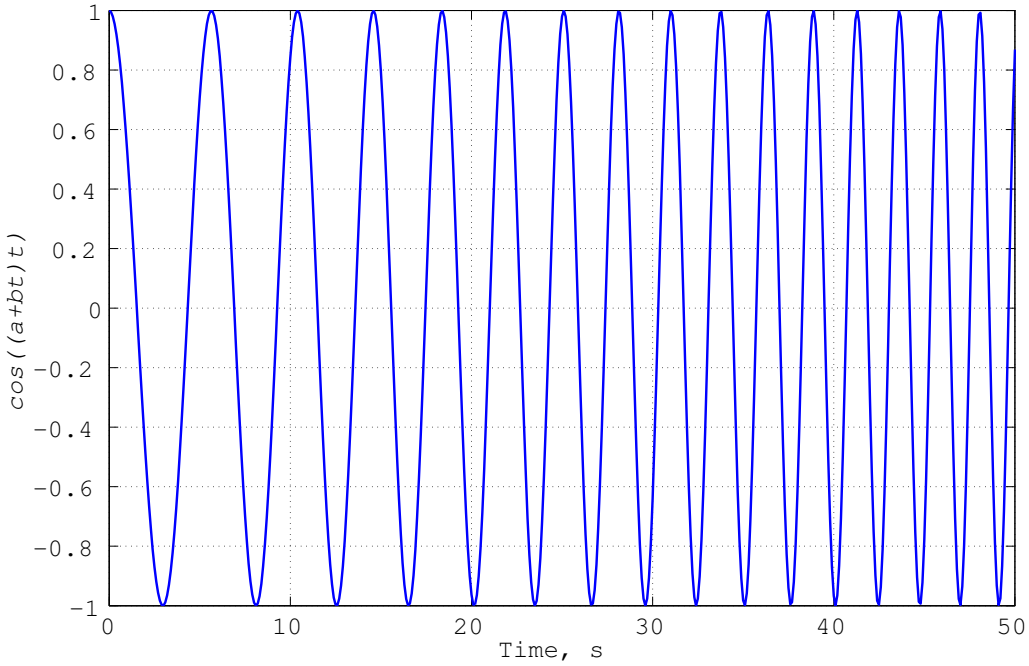
$w_j(t) = \arctan\left(\frac{H_{c_j}(t+1)}{c_j(t+1)}\right) - \arctan\left(\frac{H_{c_j}(t)}{c_j(t)}\right)$ is the *instantaneous frequency*;

Equation (4.14) represents the amplitude and the instantaneous frequency as a function of time in a three-dimensional plot, in which amplitude can be contoured on the frequency-time plane. This

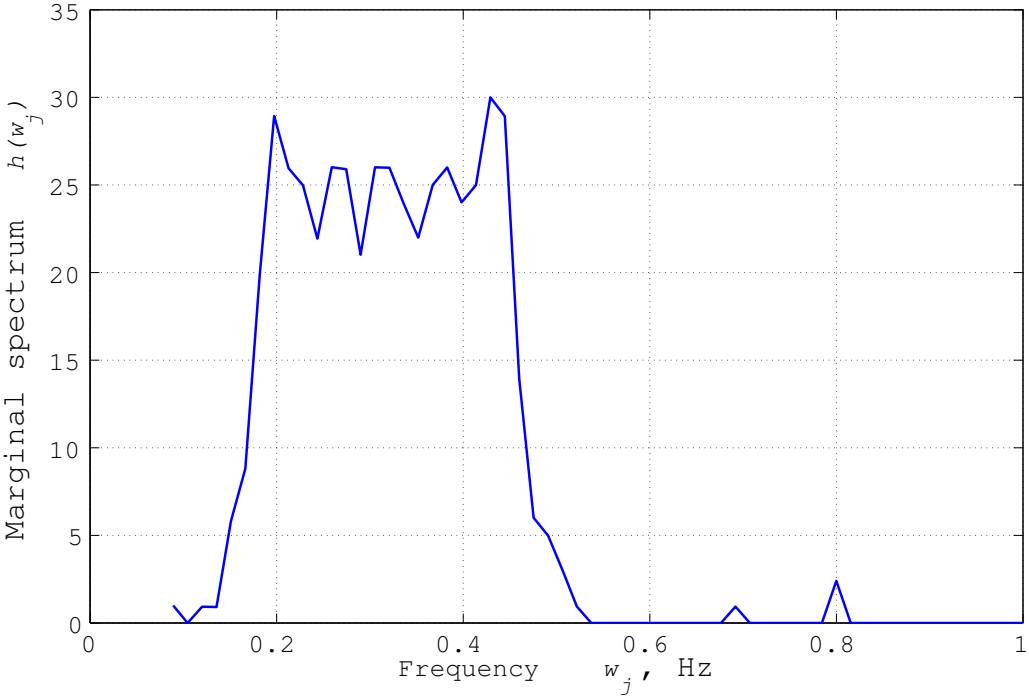
frequency-time amplitude distribution is called the *Hilbert amplitude spectrum* $U(t, w)$, or simply the *Hilbert spectrum* [HSL+98]. In addition to the Hilbert spectrum, we define the *marginal spectrum* or *HTT power spectral density* $h(w)$, as

$$h(w_j) = \sum_{t=1}^{T} U(t, w_j) \tag{4.15}$$

The marginal spectrum measures the total amplitude (or energy) contributed by each frequency value.

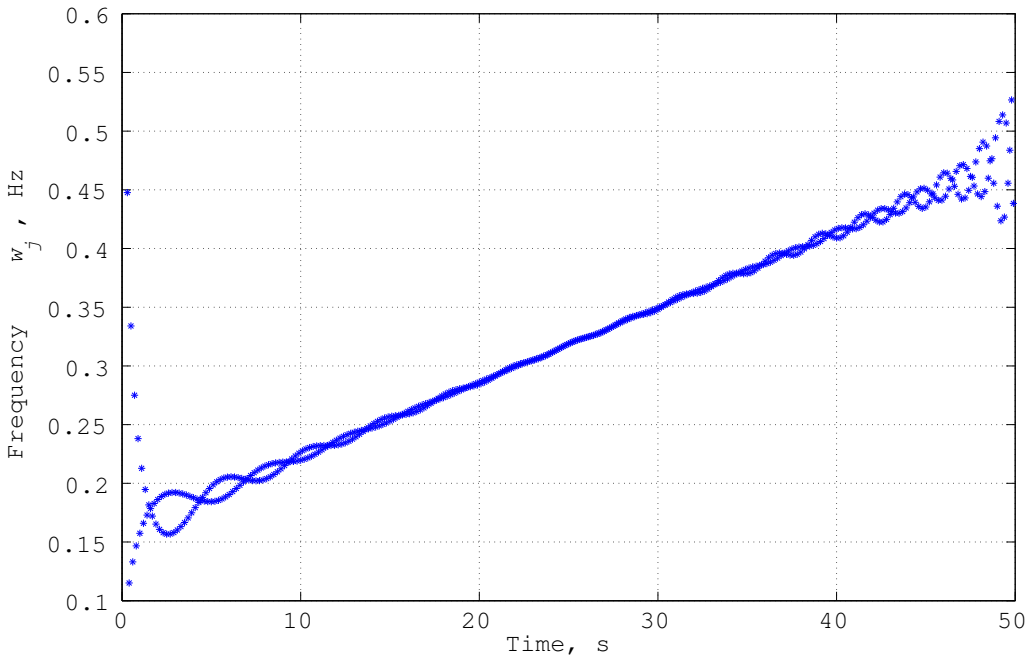


(a) The increasing frequency function $\cos((a + bt)t)$

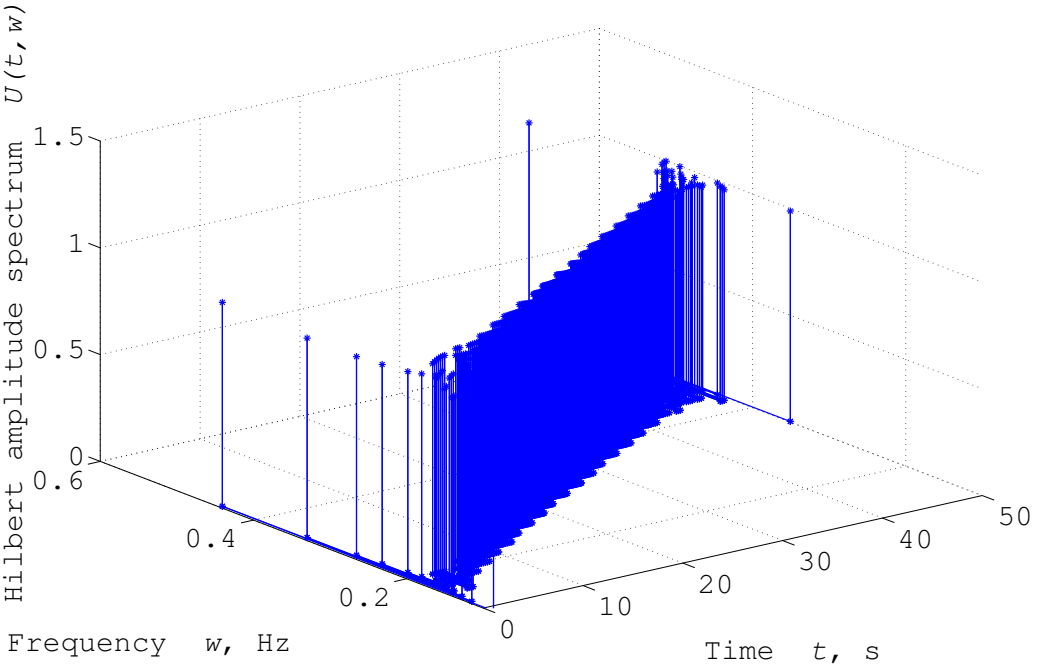


(b) Marginal Hilbert spectrum of Fig. 4.2a

Figure 4.2 – Marginal Hilbert spectrum of the function $\cos((a + bt)t)$.



(a) Hilbert's amplitude spectrum contour of Fig. 4.2a



(b) Hilbert's amplitude spectrum of Fig. 4.2a

Figure 4.3 – Hilbert amplitude spectrum of the function $\cos((a + bt)t)$.

To illustrate HHT decomposition consider the function $u(t) = \cos(t\,(a + bt))$. In Fig. 4.2a parameters a and b were arbitrarily set to $a = 1$ and $b = 0.02$. Fig. 4.2a shows that the cosine's frequency increases progressively. Fig. 4.2b presents the Hilbert marginal spectrum of the signal $u(t) = \cos((1 + 0.02t)t)$. Fig. 4.3a shows the contour of Hilbert's amplitude spectrum, *i.e.*, frequency evolution in time, and this evolution is indeed nearly linear. The 3D Hilbert amplitude spectrum is illustrated in Fig. 4.3b.

4.2.3 AES Hardware Implementation

The AES-128 implementation used for our experiments runs on an Altera Cyclone II FPGA development board clocked by an external 50MHz oscillator. The AES architecture uses a 128-bit datapath. Each AES

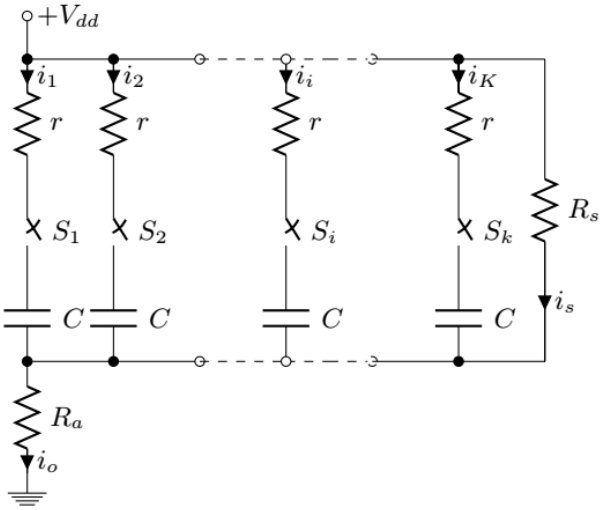


Figure 4.4 – Inverters switch simulation.

round is completed in one clock cycle and key schedule is performed during encryption. The S-box is described as a VHDL table mapped into combinational logic after FPGA synthesis. Encryption is triggered by a high `start` signal. After completing the rounds the device halts and drives a `done` signal high.

The implementation has no side-channel countermeasures. To simulate DVS, 200,000 physically acquired power consumption traces were processed by Algorithm 4. Algorithm 4 splits a time-series into segments and adds a uniformly distributed random voltage offset to each segment.

The rationale for simulating a DVS by processing a real signal (rather than adding a simple DVS module to the FPGA) is the desire to work with a rigorously modelled signal, free of the power consumption artefacts created by the DVS module itself.

4.3 Hilbert Huang Transform and Frequency Leakage

4.3.1 Why Should Instantaneous Frequency Variations Leak Information?

Most of the power consumed by a digital circuit is dissipated during rising or falling clock edges when registers are rewritten with new values. This activity is typically reflected in the power consumption trace as spikes occurring exactly during clock rising edges. Spike frequency, computed by the Fourier transform, is usually assumed to be constant because clock frequency is stable. In reality, this assumption is incorrect since each spike has its own duty cycle and consequently its own assortment of frequencies.

Differences in duty cycle come from the fact that the circuit's power supply must be restored to its nominal value after switching. Bigger amplitude spikes take more time to resorb than smaller amplitude ones.

To illustrate these spike differences, consider the simple circuit in Fig. 4.4. Each parallel branch has a resistor r, a switch S_i and a capacitor C that simulate a single inverter when switched from low to high. Resistor R_s and the current i_s represent the circuit's static current and R_a is the resistor used for acquisition. Initially all the switches $S_1 \dots S_k$ are open, so the current flowing through R_a is simply i_s.

Assume that at $t_0 = 0$ all the switches $S_1 \dots S_k$ are suddenly closed. All capacitors start charging and current flowing through R_a rises according to the following equation:

$$i_o(t) = i_s + k\left(\frac{V_{dd}}{r}e^{-\frac{t}{rC}}\right) \tag{4.16}$$

Equation (4.16) shows that current amplitude depends on the number of closed switches. However, there is one more parameter in the equation, namely the time t that characterizes the switching spike. The

current i_o needs some time to "practically" reach an asymptotic nominal value i_s and this time depends on the number of closed switches k. Consider the time T_k required by $i_o(t)$ to reach Γ% of its asymptotic value, *i.e.* $\frac{\Gamma}{100} i_s$:

$$i_o(T_k) = i_s - k\left(\frac{V_{dd}}{r} e^{-\frac{T_k}{rC}}\right) = \frac{\Gamma}{100} i_s \tag{4.17}$$

This is equivalent to:

$$T_k = rC \ln\left(\frac{100}{100-\Gamma}\frac{V_{dd}}{i_s r}\right) + rC\ln(k) = \alpha + \beta \ln(k) \tag{4.18}$$

Equation (4.18) shows that convergence time has a constant part α and a variable part $\beta \ln(k)$ that depends on the number of closed switches k. Equation (4.18) shows that both spike period and spike frequency depend on the processed data and could hence in principle be used as side-channel carriers. Nevertheless, power consumption is a non-stationary signal, which justifies the use of HHT.

The dependency between the number of switches and spike period in equation (4.18) is non-linear and hard to formalize as a simple formula for a real circuit. Section 4.3.2 shows that the standard Hamming distance model can be used in conjunction with instantaneous frequency.

4.3.2 Register Simulation

The relationship between processed binary values and power amplitude is a well understood phenomenon [AARR03, BCO04, GBTP08, KJJ99]. However, to the best of our knowledge the dependency of instantaneous frequency on processed data has not been explored so far. This may be partially explained by the fact that the Fourier Transform, previously examined in some papers, is not inherently adapted to non-stationary and non-linear signals. Fourier analysis cannot extract frequency variations from a signal because frequency is defined as a constant parameter of the underlying sine function spanning the whole data-set $u(t)$. By opposition, HHT allows extracting instantaneous frequencies and exploiting them for subsequent cryptanalytic purposes.

To illustrate information leakage through frequency variations, the power consumption of a 4-bit register was simulated using the Virtuoso toolkit. Power supply was set to 1.5V and the circuit was clocked by a 50 MHz oscillator (Fig. 4.5).

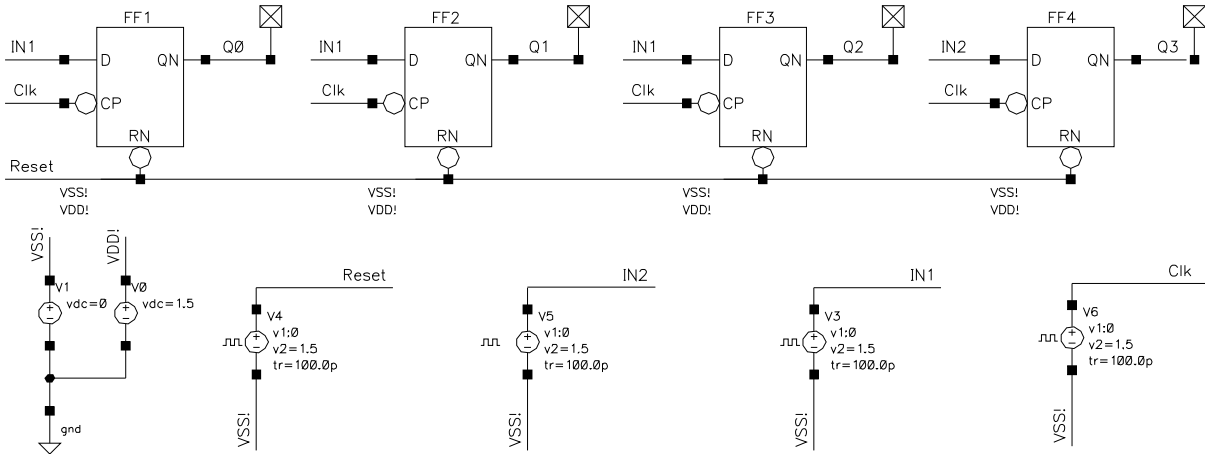


Figure 4.5 – Netlist of a 4-bit register.

Two scenarios were simulated under identical temperature and voltage conditions:

Single Latch: The register was reset. After a sufficiently long time a high input `IN2` was latched on flip-flop `FF4`. At the next clock rising edge the D-latch updated its state and transmitted a 1 to `Q3`. The simulation of the register's power consumption shown in Fig. 4.6 (blue signal).

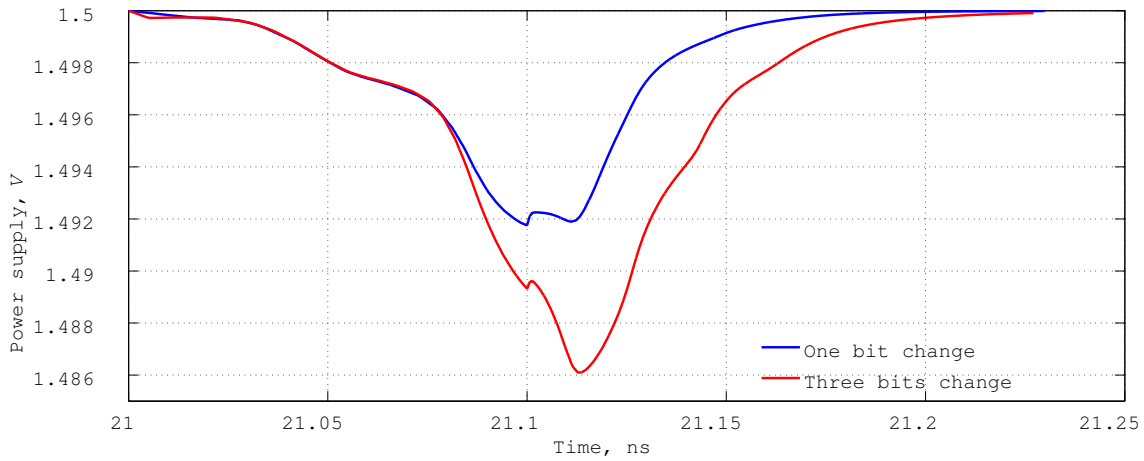
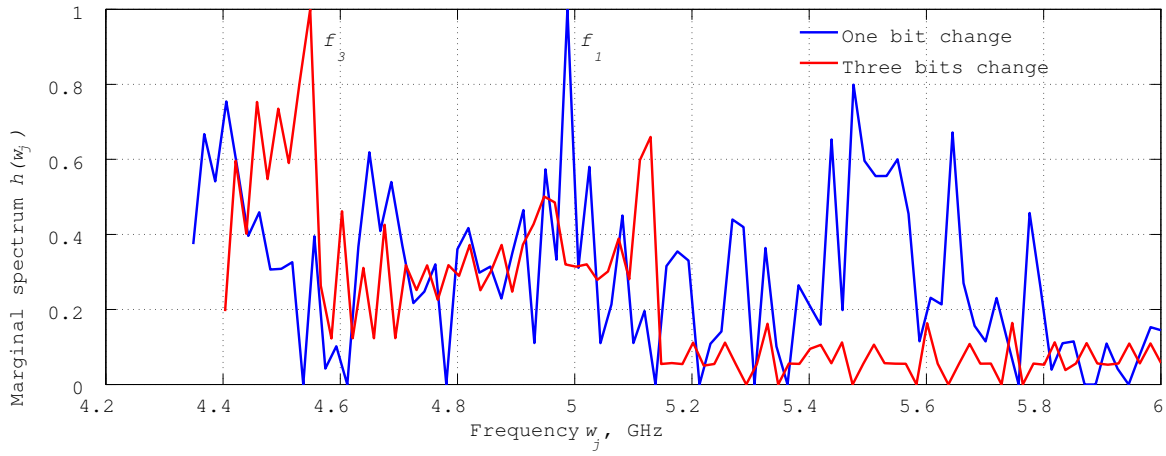
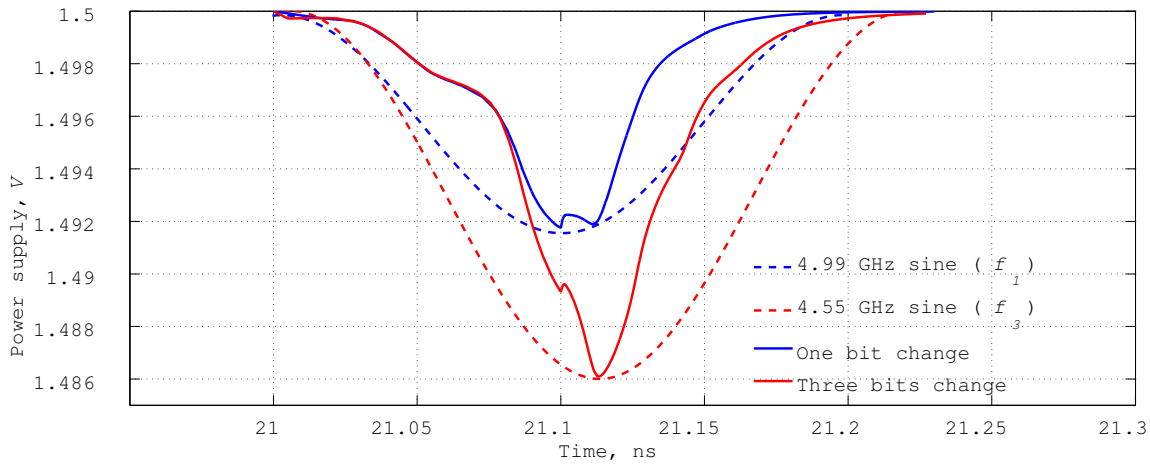


Figure 4.6 – Power consumption of register switch of 1 and 3 bits.

Triple Latch: The register was reset. After a sufficiently long time a high input `IN1` was latched on three the flip-flops `FF1`, `FF2`, `FF3`. At the next clock rising edge the three D-latches updated their state and transmitted `1`s to their outputs. Again, the power consumption's simulation is illustrated in Fig. 4.6 (red signal).



(a) Power spectrum



(b) Power consumption and sines with the frequency of the maximum spectral amplitude

Figure 4.7 – Register switch of 1 and 3 bits.

In classic side-channel models [BCO04], the energies consumed for flipping 1 bit and 3 bits differ. Fig. 4.6 shows that such is indeed the case. As per our assumption, the *frequency signatures of these two operations are also different*.

Fig. 4.6 shows that the recovery time following a 3 bits change is longer than the compensation time of 1 bit. This recovery time difference results in a frequency variation. Fig. 4.6 shows that the 3 bits' current spike has a longer pulse period than the 1 bit spike, therefore the 3 bits signal alteration presents a lower frequency. Intuition suggests (and experiments confirm) that this difference will be detected by the HHT.

To show that HHT can detect frequency differences consider the power spectral density (PSD) of both signals during 1 bit and 3 bits switch (Fig. 4.7a). The maximal spectral amplitude of the 1 bit change is located at 4.99 GHz (point f_1) while the maximal spectral amplitude of the 3 bits change (point f_3) is at 4.55 GHz which is supportive of the hypothesis that HHT can distinguish frequency variations even in non-stationary signals. As expected, Fig. 4.7b shows that two sine functions (4.55 GHz and 4.99 GHz) correspond well to the side-channels' shapes.

This shows that not only amplitude but also frequency varies during register switch. Logically, power consumption increases as more bits are flipped. However, simulation cannot prove that this variation is detectable in practice because frequency changes heavily depend on the Hamming weight of the data stored in the register. That is why the next section carefully examines the effect of register alteration on IF in a real AES FPGA implementation.

4.3.3 Power Consumption of One AES Round

To illustrate information leakage through frequency variation, the AES last rounds' power consumption was measured using a Picoscope 3207A with 250MHz bandwidth at 10G/s equivalent time sampling rate. Every signal had 1,000 samples and 100,000 traces were acquired for various input plaintexts. A power consumption example of the 4 last rounds is shown on the Fig. 4.8.

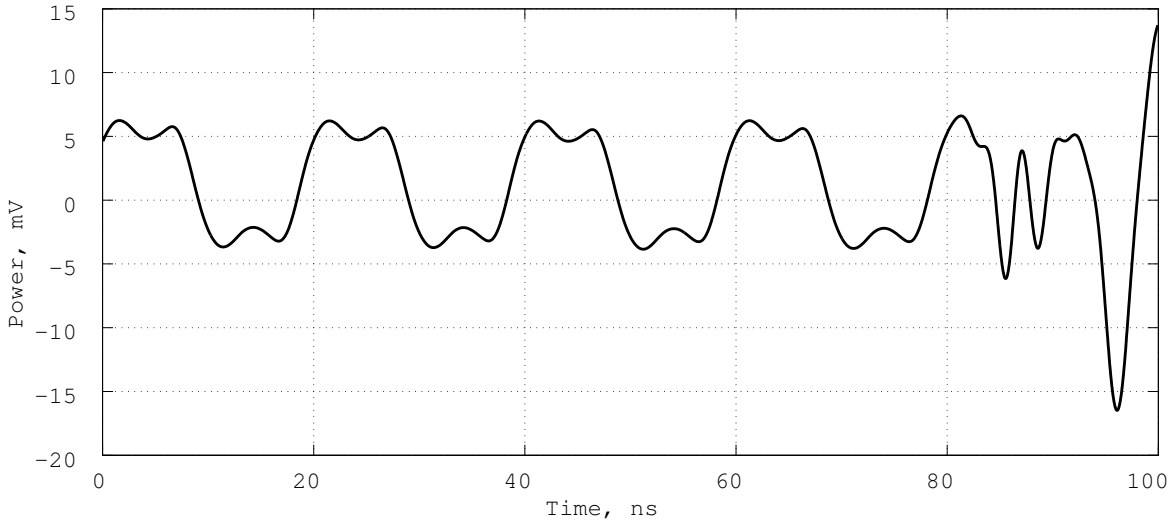
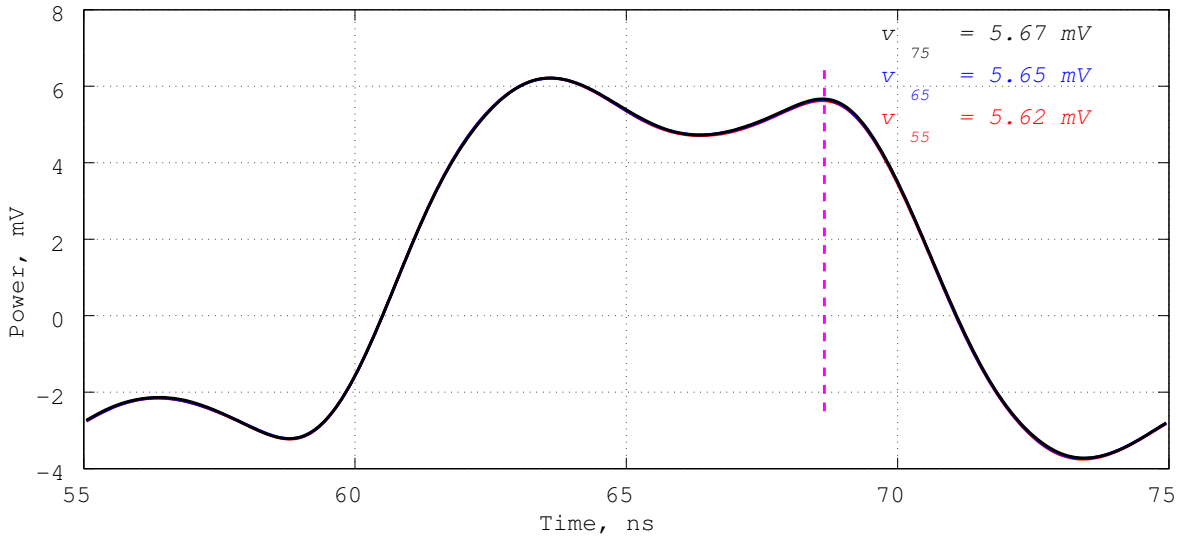


Figure 4.8 – Four AES last rounds.

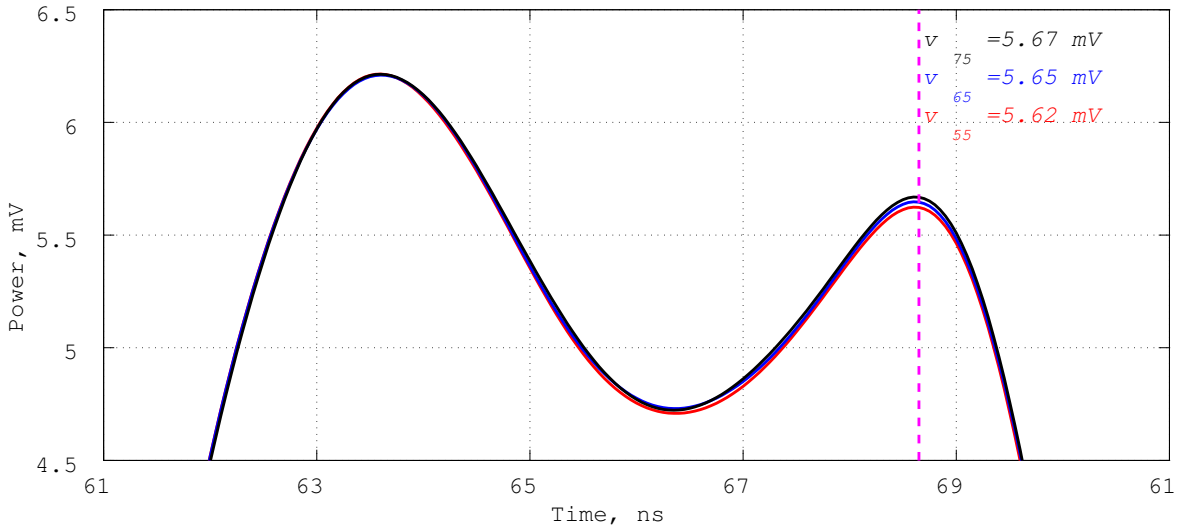
The AES last round was extracted from each power trace as shown on Fig.4.9a. The number of bits switches in the AES last round was computed with the known key. Afterwards the traces with the same number of bits switches were averaged.

In classic side-channel models [BCO04], flipping more bits would consume more energy. Fig.4.9 shows that such is indeed the case for power consumption of 55, 65 and 75 bit flips where $v_{75} > v_{65} > v_{55}$. As per our assumption, the *frequency signatures of these three operations are also different*.

To show that HHT can detect frequency differences consider the power spectral density (PSD) of signals during 55, 65 and 75 bits switchings (Fig. 4.10). The maximal spectral amplitude of the 55 bit change is located at 51.18 MHz (point f_{55}), that of the 65 bit change is at 51.12 MHz (point f_{65}) and that of the 75



(a) Full voltage range



(b) Zoomed voltage range

Figure 4.9 – AES last round power consumption for 55 (red), 65 (blue) and 75 (black) register's flip-flops.

bit change is at 50.73 MHz (point f_{75}) which is supportive of the hypothesis that HHT can distinguish frequency variations even in non-stationary signals because $f_{55} > f_{65} > f_{75}$.

This shows that not only amplitude but also frequency varies during register switch. Logically, power consumption increases as more bits are flipped. However, HHT was previously applied only for one AES round and HHT's applicability for the entire AES power traces must be verified. That is why the next section carefully examines the effect of register alteration on IF when AES FPGA implementation is sampled at a smaller rate.

4.3.4 Hilbert Huang Transform of an AES Power Consumption Signal

We start by performing a Hilbert Huang decomposition of a real signal. The analysis was performed on the power trace of the previously described AES-128 implementation. The acquisition was performed 1 G/s real time rate with 1 GHz differential probe. Signals were averaged 10 times and had 1,000 samples (Fig. 4.11).

EMD decomposed the power trace to five IMFs and a residue, shown in Fig. 4.12a. After decomposition, each IMF was Hilbert Transformed to derive the power signal's time-frequency representation. Fig. 4.12b

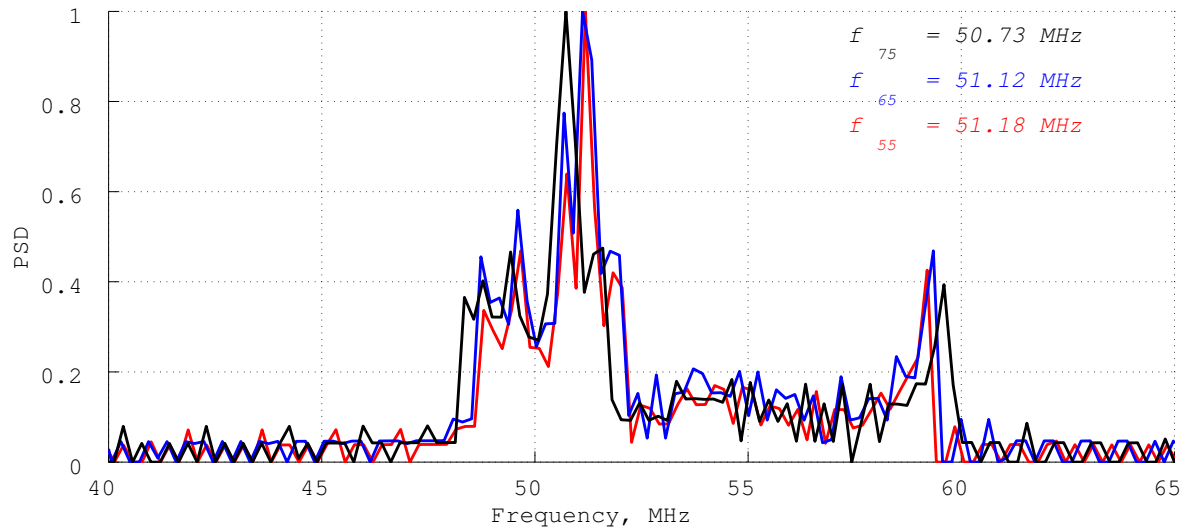


Figure 4.10 – Power spectra density for the signals shown on Fig.4.9a.

is an IF distribution of Fig. 4.11.

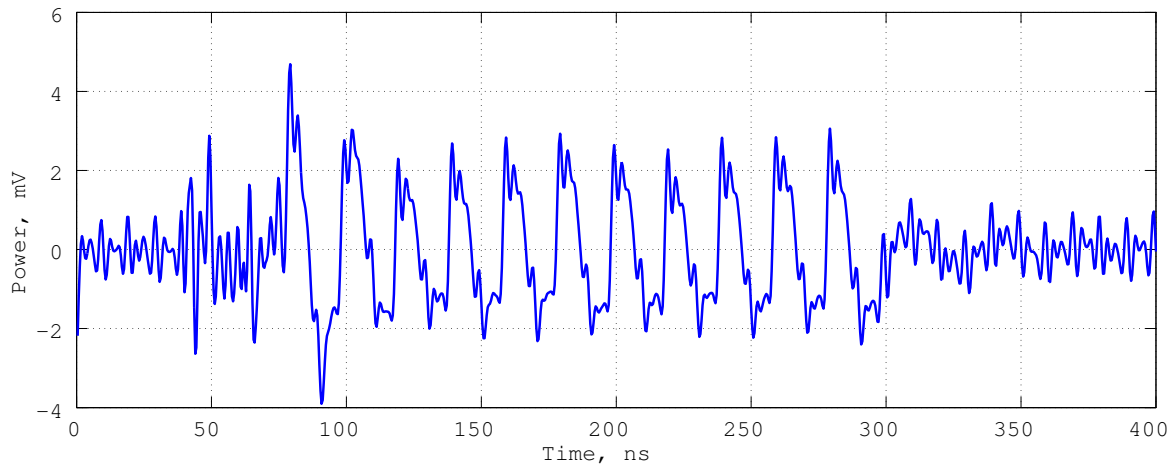


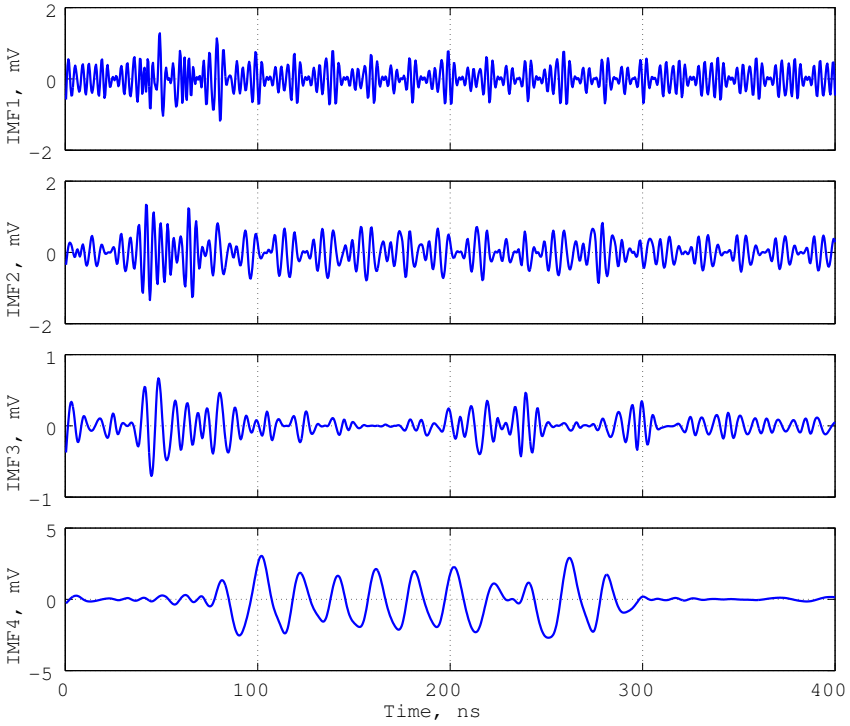
Figure 4.11 – Initial signal $u(t)$.

Amplitude combination over frequency gave the power spectral density plot shown in blue on Fig. 4.13. An important observation in Fig. 4.13 is that HHT spectrum shows the distribution of a periodic variable over the main peak frequencies. Notably, the peak near 50 MHz that corresponds to the board's oscillator is not represented by a single point, but by a set of points. This data scatter can be explained by the fact that the IF of AES rounds varies, and HHT distinguishes this variation.

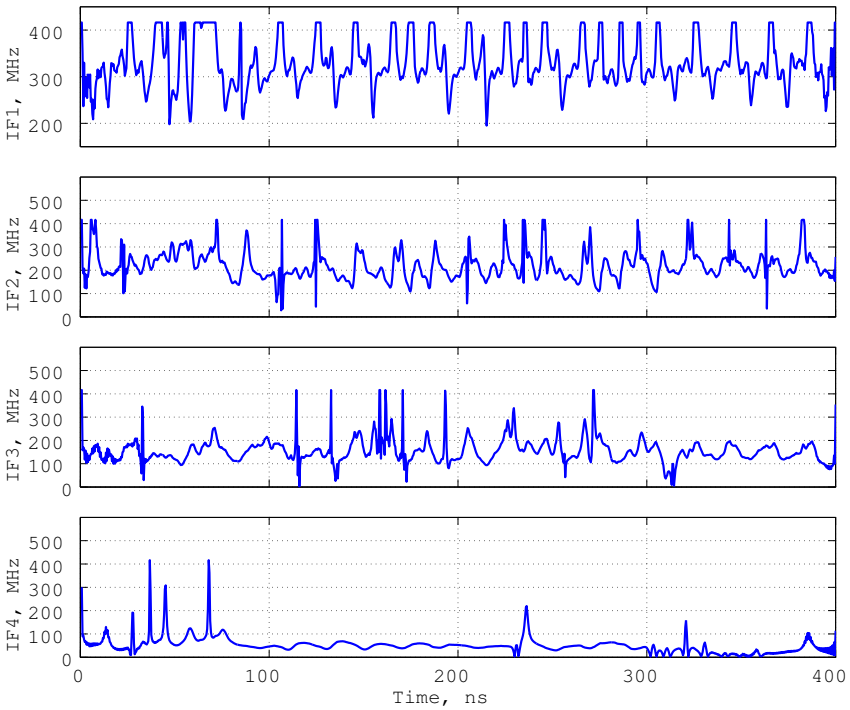
The main difference between HHT and FFT spectra (see plot shown in red on Fig. 4.13) is that HHT defines frequency as the speed of phase change and can hence detect intra-time-series deviations from the carrier's oscillation, whereas FFT frequency stems from the sine function, which is independent of the signals' shape.

So far, it was shown that IF varies for different rounds even within a given trace. However, an attack is only possible when IF depends on the data's Hamming weight.

The dependency is apparent in Fig. 4.14 showing the relationship between Hamming distance of the 9-th and 10-th AES round states and IF, taken from the first IMF component at the beginning of the 10-th round. Fig. 4.14 was drawn using 200,000 HHT-processed power traces. The thin solid line in Fig. 4.14 represents the mean IF value, obtained from the first IMF component, as a function of Hamming distance.



(a) The Empirical Mode Decomposition of signal $u(t)$



(b) IF distribution over time for the different IMFs of Fig. 4.12a.

Figure 4.12 – Power consumption of our experimental AES-128 implementation.

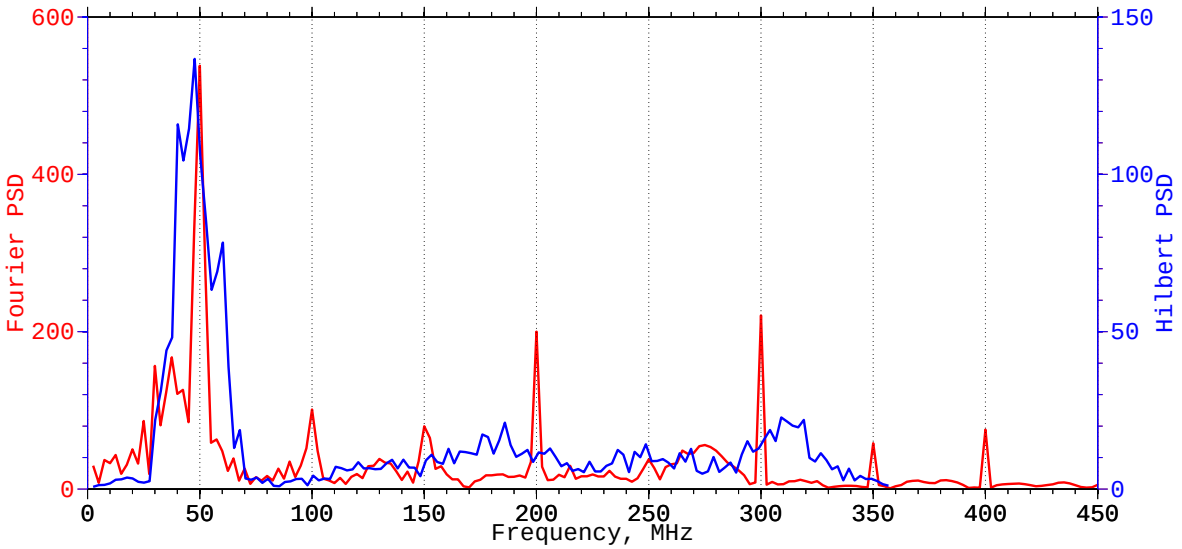


Figure 4.13 – Fourier and Hilbert power spectrum density of Fig. 4.11.

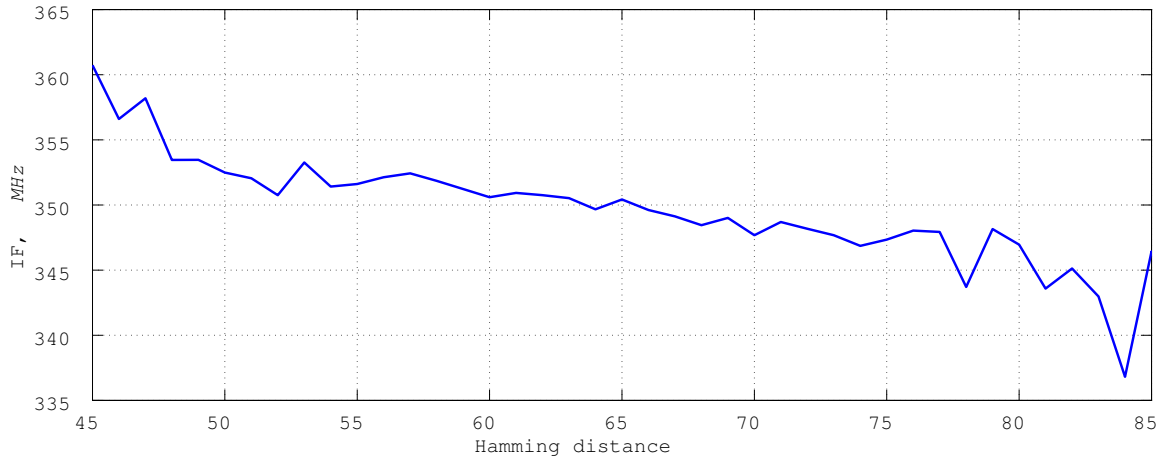


Figure 4.14 – Dependency between the Hamming distance of 9-th and 10-th AES round states and the IF of the first IMF component at time 276 ns (corresponding to the beginning of the last AES round).

The principal trend is the ascending line. Fig. 4.14 corresponds well to the simulation of a register's power consumption since frequency is decreasing due to the increase in Hamming distance. The relationship in Fig. 4.14 between Hamming distance and IF looks linear and therefore the Pearson correlation coefficient can be used as an SCA distinguisher.

IF adoption for side-channel attacks presents some particularities. The disadvantage of the method is that data scatter is higher than in usual DPA and hence the attack requires more power traces. Another issue is that each time-series will be decomposed into a set of IMFs, hence every sample will be wrapped-up with a set of IFs virtually multiplying the amount of data to be processed. However, the advantage is that because frequency-based analysis is independent of local amplitude, CIFA can still be attempted in the presence of certain countermeasures.

4.4 Correlation Instantaneous Frequency Analysis

This section introduces Correlation Instantaneous Frequency Analysis (CIFA) and compares its performance with Correlation Power Analysis (CPA) and to Correlation Spectral Based Analysis (CSBA).

4.4.1 Correlation Instantaneous Frequency Analysis on Unprotected Hardware

During the acquisition step 200,000 power traces were acquired at a sampling rate of 2.5 GS/s. Each power signal was averaged 10 times to reduce noise. All traces were HHT-processed using the Matlab HHT code of [BKMG07, BKMG12]. Most traces were decomposed into 6 components, but 5 and 7 IMFs occurred as well. To reduce the amount of processed information only the first four IMFs were used.

Generally, each higher rank IMF carries information present in smaller instantaneous frequencies (Fig. 4.12b), this is why IMFs from different power traces were aligned index-wise, *i.e.*, all first IMFs from every encryption were analyzed first, then all second IMFs and so on.

We chose the Hamming distance model and Pearson's correlation coefficient to investigate CIFA's properties and compare CIFA with other attacks. Applied SCA algorithm is given in Algorithm 1.

CPA.CPA applied to power traces produces Fig. 4.15(a). Clearly, CPA outperforms CIFA. CIFA's poorer performance can be partially attributed to the power model, because IF is not linearly dependent on the Hamming distance.

CSBA. Fig. 4.15(b) presents CSBA applied against Fourier power trace spectra with the same power power model and distinguisher. The correct key byte can be distinguished from 2000 power traces and on.

CIFA. The application of the selected power model and of the distinguisher to IFs yields Fig. 4.15(c) where the correct key byte emerges from 16,000 power traces and on.

The three experiments seem to suggest that CSBA is superior to CIFA but inferior to CPA. That is CIFA $<$ CSBA $<$ CPA.

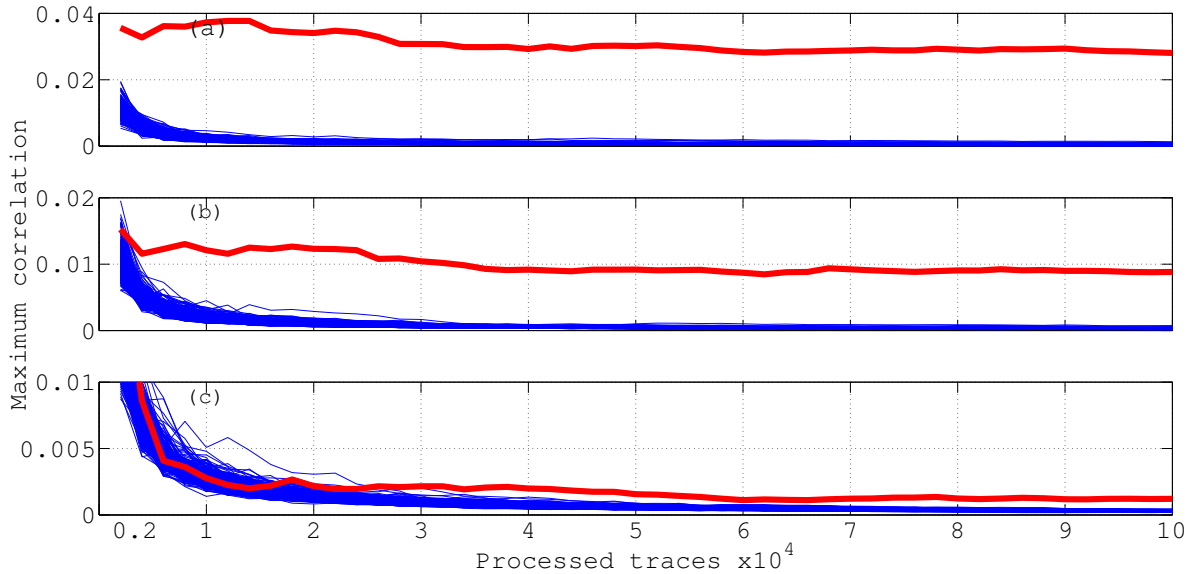


Figure 4.15 – Maximum correlation coefficients for a byte of the last round AES key in an unprotected implementation. Although the three attacks eventually succeed CPA>CSBA>CIFA. (a) CPA (b) CSBA (c) CIFA.

While it appears that CPA and CSBA outperform CIFA in the absence of countermeasures, we will now see that CIFA survives countermeasures that derail CPA and CSBA.

4.4.2 Correlation Instantaneous Frequency Analysis in the Presence of DVS

As mentioned previously DVS alters power supply to reduce dependency between data and consumed power. According to [BZ07, KGS+11] DVS is cheap in terms of area overhead since only a voltage controller and a random number generator must be added to the protected design.

Algorithm 4 Dynamic Voltage Scrambling (DVS) Simulator

Require:
A power trace $u(1), \ldots, u(N)$;
γ : the number of segments;
m : mean value of segment length $m \stackrel{\text{def}}{=} N/\gamma$;
σ : standard deviation of segment length;
D : maximum offset for segment lifting;
Ensure:
a DVS-protected power trace $u'(1), \ldots, u'(N)$;

▷ *Split a trace to a set of segments of normally distributed random length chunks*
$\tau_0 \leftarrow 1$
$\tau_\gamma \leftarrow N$
for $i = 1$ **to** $\gamma - 1$ **do**
  $\tau_i \leftarrow \tau_{i-1} + \mathcal{N}(m, \sigma)$
end for
▷ *Lift each segment by a uniformly distributed random offset* ℓ
for $s = 1$ **to** γ **do**
  $\ell_s \in_R [0, D]$
  for $t = \tau_{s-1}$ **to** τ_s **do**
    $u'(t) \leftarrow u(t) + \ell_s$
  end for
end for

To simulate DVS all the traces of the unprotected AES were modified by Algorithm 4. Each power trace was partitioned into γ segments of normally distributed lengths covering the whole dataset.[1]. Each segment was lifted by a uniformly distributed random offset ℓ that did not exceed a predetermined value D set to $D = 12$ mV.

A trace modification example is presented in Fig. 4.16, in which the trace of Fig. 4.11 was processed by Algorithm 4.

Logically, DVS decreases power analysis performance by reducing the attacker's SNR. We disposed of 200,000 DVS-modified power traces. All of which were used to mount power analysis attacks under the same conditions as before, *i.e.*, using Pearson's correlation coefficient and the Hamming distance model.

The same final round key byte used for attacks against the unprotected implementation was targeted. CPA and CSBA failed to detect the correct key byte even with 150,000 traces (Fig. 4.17(a),4.17(b). This confirms the intuition that DVS has a beneficial effect on the required number of power traces.

However CIFA was able to recover the byte from 60,000 traces and on (Fig. 4.17(c)). This illustrates that whilst CIFA is usually outperformed by CPA and CSBA, CIFA is much more resilient to DVS, to which CPA and CSBA are very sensitive.

4.5 Conclusions

This chapter investigated the use of instantaneous frequency instead of power amplitude and power spectrum in side-channel analysis. By opposition to the constant frequency used in Fourier Transform, instantaneous frequency reflects local phase differences and allows to detect frequency variations. These variations depend on the processed binary data and are hence cryptanalitically useful. The relationship stems from the fact that after higher power drops more time is required to restore power back to its nominal value.

IF analysis does not bring specific benefits when applied to unprotected designs on which CPA and CSBA yield better results. However, CIFA allows to discard the effect of amplitude modification countermeasures, *e.g.*, DVS, because CIFA extracts from signal features not exploited so far.

[1]The mean m and the standard deviation σ were arbitrary set to $m = 40$ ns and $\sigma = 5$ ns in our experiment.

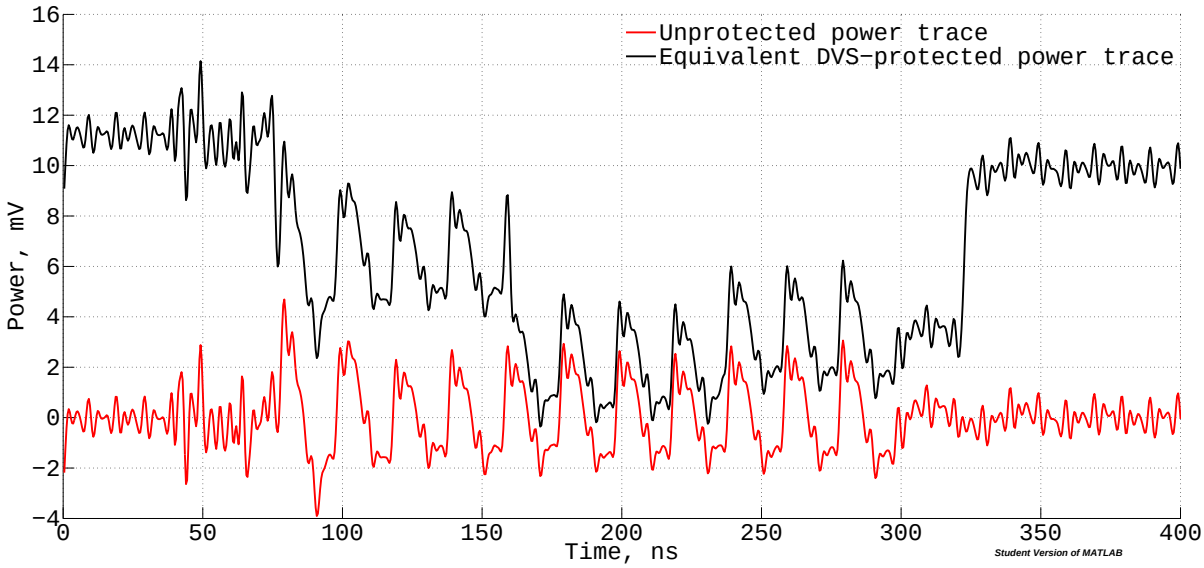


Figure 4.16 – Power traces of the FPGA AES implementation. The unprotected signal is shown in red. The DVS-protected signal is shown in black.

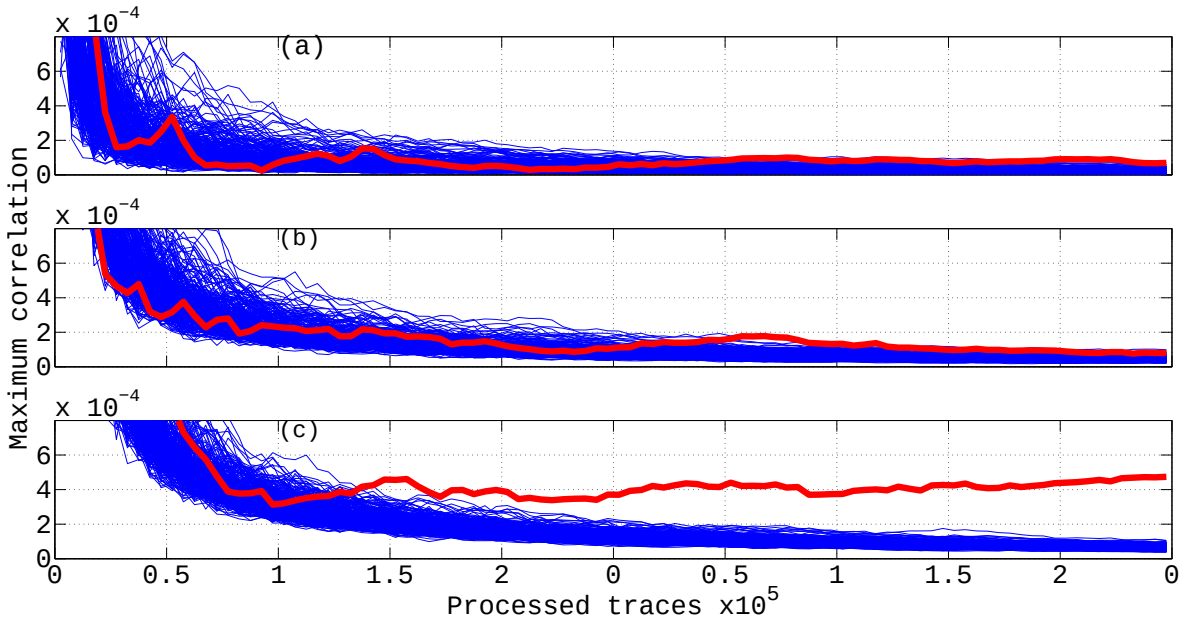


Figure 4.17 – Maximum correlation coefficient for a byte of the last round AES key with simulated DVS. (a) CPA (b) CSBA (c) CIFA.

CHAPTER 5

DISTINCT KEY DISTRIBUTION AND STATISTICAL INDISTINGUISHABILITY

Summary

This chapter introduces a novel attack approach based on distinct key distributions presented in block ciphers. Coupled with side-channel and fault information those unique key distributions prove to be a significant threat to block ciphers since the side-channel and fault attacks can be performed blindly, *i.e.*, without knowledge of ciphertext and plaintext.

This chapter presents a novel fault attack against SPN ciphers. The main advantage of the method is an absence of necessity to know the exact cipher's input and output values. The attack relies only on the number of faulty ciphertexts originating from the same unknown plaintext. The underlying model is a multiple bit-set or bit-reset faults injected several times at the same intermediate round state. This method can be applied against any round thus any round key can be extracted. The attack was shown to be efficient by simulation against several SPN block ciphers.

The new attack does not require a direct access to cipher's input and output. The attack assumes the following: (1) an attacker can encrypt several unknown plaintexts multiple times under the same key; (2) encryption results can be compared between themselves without disclosing their values (this is somewhat similar to the "generic model" often used in public-key cryptography [Mau05]); (3) a multiple random bit-reset or bit-set fault can be injected during the encryption rounds.

The method infers information from the relationship between the number of faulty ciphertexts originated from the same unknown plaintext and an intermediate state's Hamming weight. The attack is based on the fact that each SPN round comprises a key-involved operation that can reveal the round key if input and output Hamming weights of this operation are known. Simulations show that this attack is practical against LED, [GPPR11], AES [oSN01] and SAFER++ [MKK00] algorithms.

The attack is performed in two phases: fault injection and key search. The fault injection phase is used to determine the Hamming weights of intermediate states. When a number of fault injections is limited, we determine an occurrence probability for each possible Hamming weight value. The key search phase is done in two steps. The first step applies a finite field equation to filter-out key candidates. The second step assigns likelihood information to each key candidate which, in turn, reveals the correct key with high probability.

5.1 Statistical Indistinguishability

Statistical indistinguishability is a cornerstone of modern cryptography, underlying fundamental applications such as pseudorandom generators, secure encryptions, commitment schemes and much more [GMR89, Gol98].

Statistical indistinguishability captures a situation in which the *statistical distance* between two distributions X and Y tends to 0 faster that any inverse polynomial, that is, it is so-called *negligible function*.

Definition 24 [Statistical Distance Δ] Given two random variables X, Y taking values in a set V, the *statistical distance* is defined as:

$$\Delta(X,Y) = \frac{1}{2}\sum_{\forall v \in V} |\Pr[X = v] - \Pr[Y = v]|$$

Definition 25 [Negligible Function] A function $f : \mathbb{N} \to [0,1]$ is called *negligible*, denoted as $\mathrm{negl}(n)$, if for all $c \in \mathbb{N}$ there exists $n_c \in \mathbb{N}$ such that $f(n) \leq n^{-c}$ for all $n \geq n_c$.

Statistical distance can be defined in several manners, for example from the Kolmogorov-Smirnov test [Haz01], Hellinger distance [Nik01], *etc*.

In this thesis statistical distance between two distributions X and Y over a domain Θ is defined as Euclidean distance:

$$\Delta(X,Y) = \sqrt{\sum_{\theta \in \Theta} \left(\Pr[X = \theta] - \Pr[Y = \theta]\right)^2}$$

Definition 26 [Statistical Indistinguishability] Let $|X_n : x_i,\ x_i \in V_X, 1 \leq i \leq n|$ and $|Y_n : y_i,\ y_i \in V_Y, 1 \leq i \leq n|$ be two random sets. We say that X_n and Y_n are *statistically indistinguishable*, written $X_n \overset{s}{\approx} Y_n$, if their statistical distance is negligible:

$$\Delta(X_n, Y_n) = \mathrm{negl}(n)$$

Statistical indistinguishability allows defining *leakage-immune* functions F as in [CKN01].

Definition 27 [Leakage immune operation F] The operation $F(K,X)$ is *leakage immune* if $F(K,X) \overset{s}{\approx} F(K',X')$ for all distributions (K,X) and (K',X').

In the rest of this chapter $F()$ will denote leakage-immune operations whereas $\bar{F}()$ will denote non-leakage immune operations.

Leakage immunity is a fundamental principle used in many attacks including differential [BS91] and linear [Mat94] cryptanalysis. SCA also applies this principle. First-order methods collect leakage statistics of a single point per side channel measurement [KJJ99, BCO04]. This leakage usually stems from the non-leakage immunity of an S-box related operation. High-order methods combine several leakage points and build combined statistics to defeat countermeasures [Mes00, OMHT06].

Previous SCA operate with one-dimensional statistics, *i.e.*, either a single leakage point or a function of leakage points. This work shows that a subkey value can cause a unique distribution between several side-channel points. This distribution is built without plaintext data, *i.e.*, just with side-channel queries, thus an attack can be applied when both x_{in} and x_{out} values are unknown. The following section introduces unique subkey-dependent distributions which can be recovered with side-channel queries only.

5.2 Hamming Weight Probability Distributions

As explained in Section 2.2 Hamming weight is the maximal information which can be inferred from side-channel data, collected on modern CMOS devices. When the data itself is unknown the key might be found from the recovered Hamming weights.

Consider a set of functions $\bar{F}_1(k, x_{\text{in}}), \bar{F}_2(k, x_{\text{in}}), ..., \bar{F}_M(k, x_{\text{in}})$ defined for a given key $k \in \mathbb{F}_{2^m}$ and an input vector $X_{\text{in}} = \{x_{\text{in}}^i\}_{0 \leq i \leq 2^m - 1}$. Subkey recovery can be performed when each subkey k corresponds to a unique Hamming weight distribution, introduced as follows:

Definition 28 [Hamming Weight Probability Distribution (HWPD)] *Hamming Weight Probability Distribution* is a discrete probability distribution used to observe a set of Hamming weights:

$$\Pr_k[h_0, h_1, h_2, ..., h_M] = \left| \frac{\#x_{\text{in}}^i}{2^m} : \begin{cases} \text{HW}(x_{\text{in}}^i) = h_0 \\ \text{HW}(\bar{F}_1(k, x_{\text{in}}^i)) = h_1 \\ ... \\ \text{HW}(\bar{F}_M(k, x_{\text{in}}^i)) = h_M \end{cases} \quad k, x_{\text{in}}^i \in \mathbb{F}_{2^m} \right|$$

An HWPD can be computed for any set of operations $\bar{F}_1, \bar{F}_2, ..., \bar{F}_M$, however, only non leakage-immune operations would provide key uniqueness. Note, that

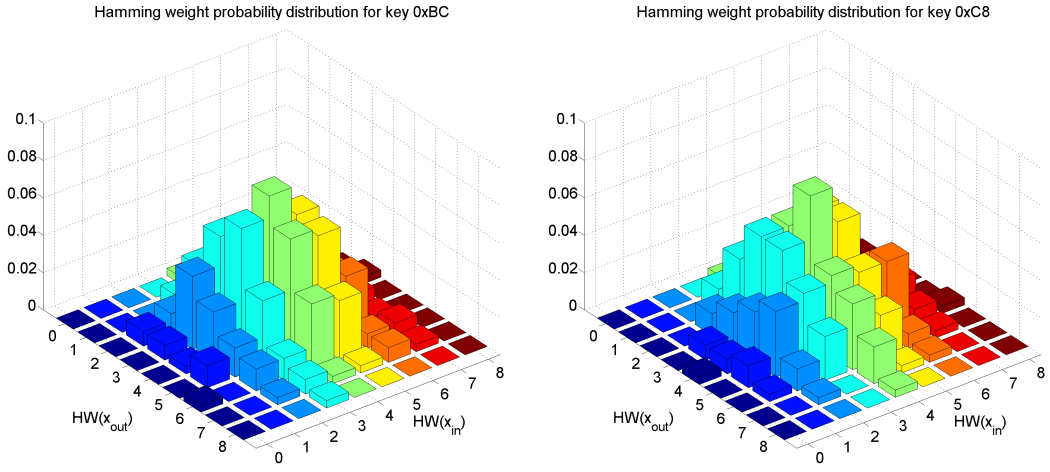
$$\sum_{h_0,h_1,h_2,...,h_M=0}^{m} \Pr_k[h_0, h_1, h_2, ..., h_M] = 1$$

To illustrate unique key distributions consider a typical block cipher operation $\bar{F}_1 = \mathbf{S}(k \oplus x_{\text{in}})$. HWPD can be computed by equation (5.1):

$$\Pr_k[\text{HW}(x_{\text{in}}), \text{HW}(\mathbf{S}(k \oplus x_{\text{in}}))] \tag{5.1}$$

Fig. 5.1 illustrates HWPDs (5.1) computed for AES using the two keys `0xBC` and `0xC8`. Clearly, the two HWPDs can be visually distinguished and can thus serve for key recovery. To verify the uniqueness of the HWPD a statistical distance between all subkeys $k_i, k_j \in \mathbb{F}_{2^m}$ can be computed as follows:

$$\begin{aligned} \Delta^2_{k_i,k_j} \quad = \quad \sum_{h_0=0}^{m} \sum_{h_1=0}^{n} \Big(& \Pr\big[\text{HW}(x_{\text{in}}) = h_0, \text{HW}(\mathbf{S}(k_i \oplus x_{\text{in}})) = h_1\big] - \\ & - \Pr\big[\text{HW}(x_{\text{in}}) = h_0, \text{HW}(\mathbf{S}[k_j \oplus x_{\text{in}}]) = h_1\big] \Big)^2 \end{aligned} \tag{5.2}$$



(a) HWPD for the key `0xBC` (b) HWPD for the key `0xC8`

Figure 5.1 – Hamming weight probability distribution $\Pr_k\big[\text{HW}(x_{\text{in}}), \text{HW}(\mathbf{S}(k \oplus x_{\text{in}}))\big]$ for AES.

A statistical distance between distributions (5.1) computed for all the subkeys $k_i, k_j \in \mathbb{F}_{2^8}$ using AES S-box is illustrated on Fig. 5.2a. Figure 5.2a shows that statistical distances computed for $(k_i, k_j),\ i \neq j$ are never equal to zero, thus all HWPDs are different. Therefore, a HWPD uniquely defines an AES subkey value. Similar statistical distances can be computed for other block ciphers as shown in Appendix A.

HWPD can be extended for several operations, for example $k \oplus x_{\text{in}}$ and $\mathbf{S}(k \oplus x_{\text{in}})$ as defined by equation (5.3). In that case the statistical distance is given by the equation (5.4). The statistical distance computed for all the AES subkey values is shown in Fig. 5.2b. Fig. 5.2a shows the statistical distance computed

for HWPDs with two components, *i.e.*, $\Pr_k\left[\mathrm{HW}(x_{\text{in}}), \mathrm{HW}(\mathbf{S}(k \oplus x_{\text{in}}))\right]$, Fig. 5.2b shows the statistical distance computed for HWPDs with three components $\Pr_k\left[\mathrm{HW}(x_{\text{in}}), \mathrm{HW}(k \oplus x_{\text{in}}), \mathrm{HW}(\mathbf{S}(k \oplus x_{\text{in}}))\right]$. The colour of each square indicate the statistical distance value.

Again, there is no key pair which has identical HWPDs. Comparison between Fig. 5.2a and Fig. 5.2b shows that the absolute statistical distance between key values became smaller when two $\bar{F}$ functions were applied.

$$\Pr_k\left[\mathrm{HW}(x_{\text{in}}), \mathrm{HW}(k \oplus x_{\text{in}}), \mathrm{HW}(\mathbf{S}(k \oplus x_{\text{in}}))\right] \tag{5.3}$$

$$\Delta^2_{k_i,k_j} \quad = \quad \sum_{h_0=0}^{m} \sum_{h_1=0}^{m} \sum_{h_2=0}^{n} \left(\Pr_{k_i}\left[h_0, h_1, h_2\right] - \Pr_{k_j}\left[h_0, h_1, h_2\right]\right)^2 \tag{5.4}$$

In the rest of the thesis a *real* HWPD will designate an HWPD constructed with side-channel measurements, while a *pattern* HWPD will mean a HWPD constructed by simulation for each key guess. An adversary may find a correct subkey by computing a statistical distance between a real HWPD and a pattern HWPD. The smallest statistical distance shall correspond to the correct subkey.

Side-channel measurements are imprecise thus a real HWPD might be recovered with errors. Key recovery rate depends on the percentage of erroneous Hamming weights used for HWPD computation. Error tolerance can be verified by simulation when a certain percentage of Hamming weights is replaced by wrong values. Key recovery success rates using HWPD (5.1) are illustrated in Fig. 5.3a and using HWPD (5.3) in Fig. 5.3b.

As illustrated by Fig. 5.3 key recovery can tolerate errors in Hamming weight detection. The threshold shown on the graphs indicates that up to 5 Hamming weight pairs for HWPD (5.1) and up to 33 Hamming weight triples for HWPD (5.3) can be wrongly detected before key recovery rate drops below 1. This tolerance allows applying HWPD key recovery in practice and the following section discusses the practical aspects of this attack.

5.3 Blind Fault Attacks Against SPNs

The new attack targets an SPN operation between two rounds, r and $r+1$, shown in Figure 5.4 in blue. This operation can be described by equation (5.5).

$$S_j^{[r+1]} = \mathbf{S}_j^{[r+1]} \circ \mathbf{A}\mid_{k_j^{[r]}} \left(S_j^{[r]}\right), \quad j \in [1, m] \tag{5.5}$$

where $S_j^{[r+1]} \in \mathbb{F}_{2^b}$ is an output of one S-box operation $\mathbf{S}_j^{[r+1]}$, $\mathbf{A}\mid_{k_j^{[r]}}$ mixes a key part $k_j^{[r]} \in \mathbb{F}_{2^b}$ with a state part $S_j^{[r]} \in \mathbb{F}_{2^b}$.

For simplicity this cryptographic operation is denoted as:

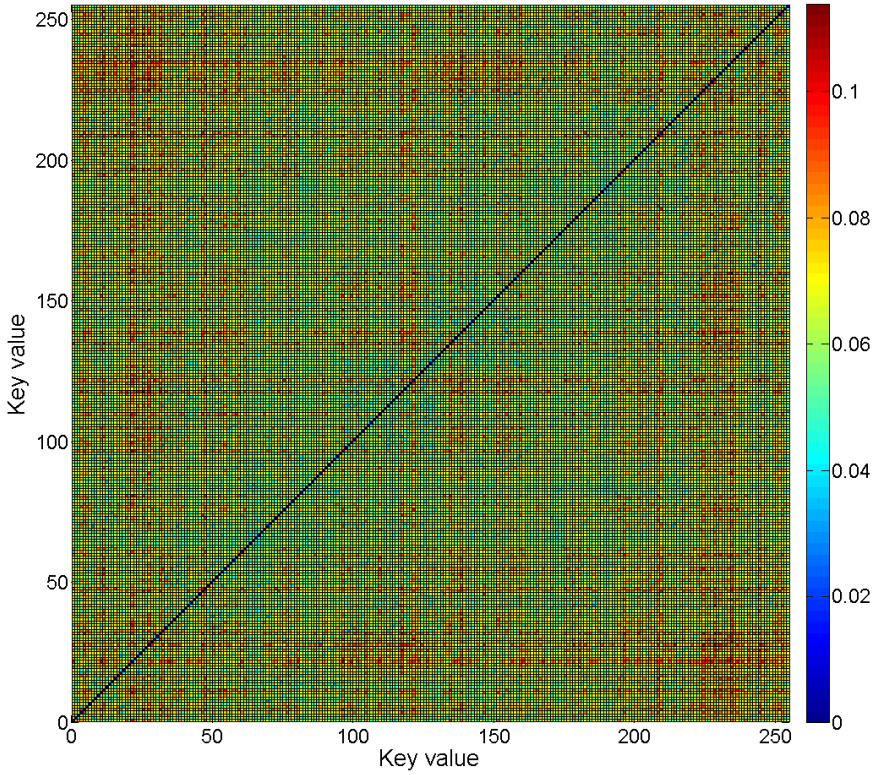
$$F(k_j^{[r]}, S_j^{[r]}) = \mathbf{S}_j^{[r+1]} \circ \mathbf{A}\mid_{k_j^{[r]}} \left(S_j^{[r]}\right) \tag{5.6}$$

The input $S_j^{[r]}$ and output $S_j^{[r+1]}$ size b is 4-bit for the ciphers LED [GPPR11] and KLEIN [GNL12] while 8-bit input and output variables are used in AES [AES01], PRESENT [BKL+07] and SAFER++ [MKK00]. However any other possible S-box input size can be considered.

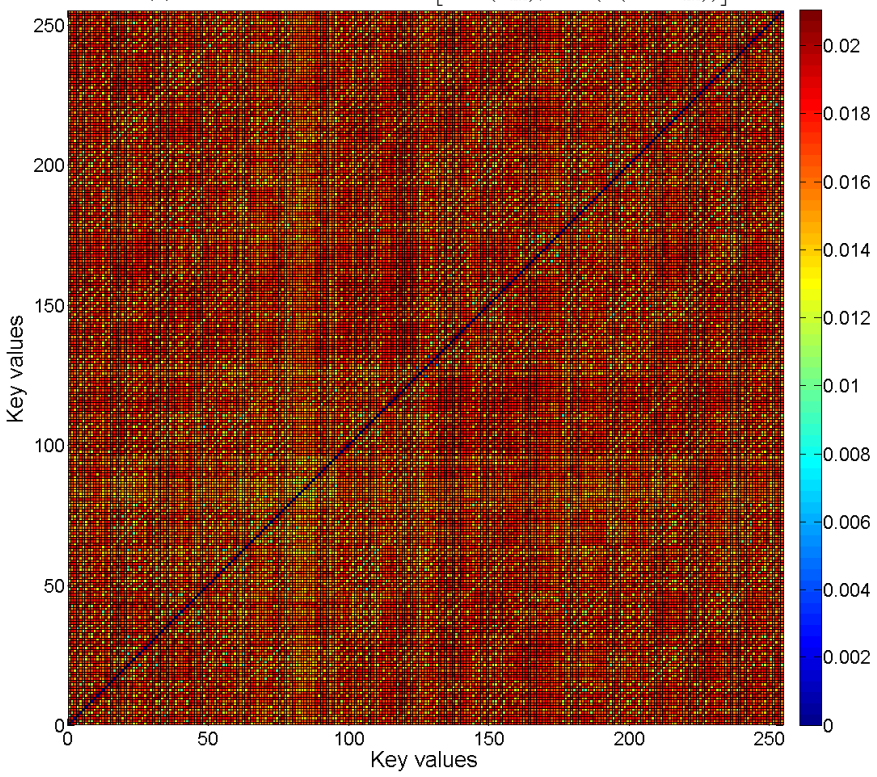
The main attacks idea is that a Hamming weight pair $\left(\mathrm{HW}(S_j^{[r]}), \mathrm{HW}\left(F(k_j^{[r]}, S_j^{[r]})\right)\right)$ can be used to distinguish key values $k_j^{[r]}$. This fact was demonstrated before in the section 5.2.

HWPD key dependency can be exploited when the Hamming weight of the operation's input and output are known to the attacker. So the first problem is to find the input and output Hamming weights for operation $F(k_j^{[r]}, S_j^{[r]})$. To obtain a Hamming weight of an intermediate state a multiple bit-reset fault model can be used:

$$\tilde{S}^{[r]} = S_j^{[r]} \wedge e \quad \text{for} \quad S_j^{[r]}, e \in \mathbb{F}_{2^b} \tag{5.7}$$



(a) Distance for HWPD $\Pr_k\left[\mathrm{HW}(x_{\mathrm{in}}), \mathrm{HW}(\mathbf{S}(k \oplus x_{\mathrm{in}}))\right]$

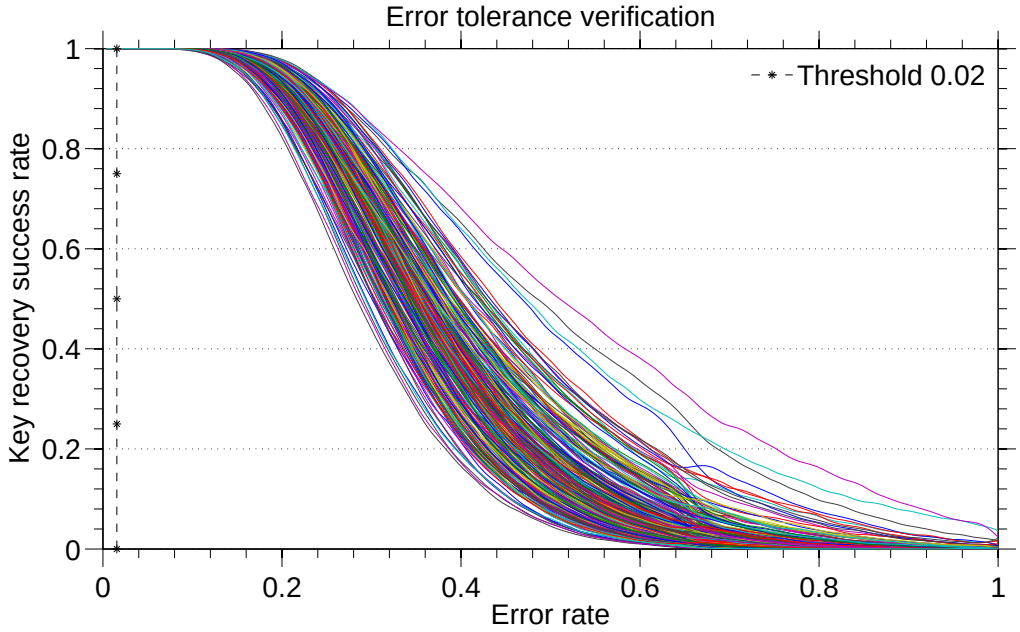


(b) Distance for HWPD $\Pr_k\left[\mathrm{HW}(x_{\mathrm{in}}), \mathrm{HW}(k \oplus x_{\mathrm{in}}), \mathrm{HW}(\mathbf{S}(k \oplus x_{\mathrm{in}}))\right]$

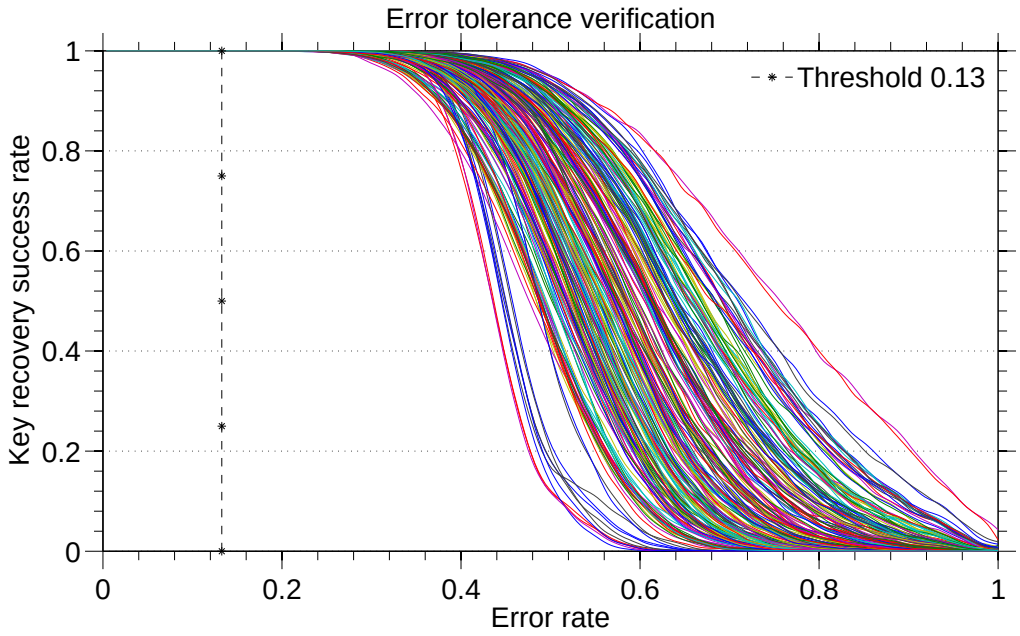
Figure 5.2 – Statistical distance for AES HWPDs.

where $S_j^{[r]}$ is a j-th part of the state in round r.

The maximum number of possible values for $\tilde{S}_j^{[r]}$, denoted by $\lambda = \#\tilde{S}_j^{[r]}$, is a function of the data block's



(a) HWPD $\Pr_k\left[\mathrm{HW}(x_{\text{in}}), \mathrm{HW}(\mathbf{S}(k \oplus x_{\text{in}}))\right]$



(b) HWPD $\Pr_k\left[\mathrm{HW}(x_{\text{in}}), \mathrm{HW}(k \oplus x_{\text{in}}), \mathrm{HW}(\mathbf{S}(k \oplus x_{\text{in}}))\right]$

Figure 5.3 – Key recovery success rate of AES operation.

Hamming weight $\mathrm{HW}\left(S_j^{[r]}\right) = h_j^{[r]}$, as shown by equation (5.8).

$$\lambda = \#\tilde{S}_j^{[r]} = 2^{h_j^{[r]}} \tag{5.8}$$

Since each value of $\tilde{S}_j^{[r]} = S_j^{[r]} \wedge e$ will lead to a different output, equation (5.8) also links the maximal number of observed ciphertexts and the Hamming weight of $S_j^{[r]}$. For byte values the maximal number of different ciphertexts is $\lambda \in \{1, 2, \ldots, 256\}$ while for nibbles $\lambda \in \{1, 2, \ldots, 16\}$.

In this method, fault injection is used to determine the Hamming weight of the input and output state-parts $S_j^{[r]}$ and $S_j^{[r+1]}$ used in equation (5.5). The manner in which Hamming weight is determined by fault

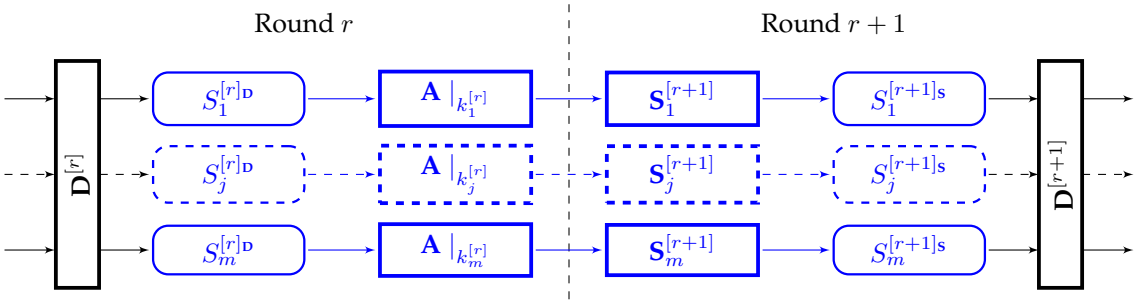


Figure 5.4 – Confusion operation at round $r+1$.

injection is illustrated in the next subsection 5.3.1. Once the Hamming weight of the pair $S_j^{[r]\mathrm{D}}, S_j^{[r+1]\mathrm{S}}$ ($S^{[r]\mathrm{D}}$ is state at round r after diffusion layer and $S^{[r+1]\mathrm{S}}$ is state at round $r+1$ after substitution layer) is found, the attacker can search the correct key value using key sifting and key likelihood information, as will be discussed in subsection 5.3.2.

5.3.1 Hamming Weight Computation

The attack assumes that a multiple bit-reset error[1] e can be invoked in a middle cipher round:

$$\tilde{S}_j^{[r]} = S_j^{[r]} \wedge e \quad \text{for} \quad S_j^{[r]}, e \in \mathbb{F}_{2^b} \tag{5.9}$$

The error e is assumed to be uniformly distributed over the finite field $\mathbb{F}_{2^b}$, hence all ciphertexts have equal appearance probabilities. The value $\tilde{S}_j^{[r]}$ cannot be directly accessed, so the attack's principal idea is to determine the Hamming weight of this variable by injecting N_ℓ random multiple bit-reset faults and observing the number of different outcoming ciphertexts.

Assume that N_c different ciphertexts are observed after N_ℓ fault injections. It is possible to compute the probability that faults injected into a variable with the Hamming weight $2^{h_j^{[r]}}$ could produce N_c different ciphertexts. This can be considered as an occupancy problem [Fel68] where N_c out of λ bins are occupied after throwing N_ℓ balls.

In the classical occupancy problem, the probability $\Pr(N_c = n_\kappa)$ can be computed using equation (5.10) given in [Har68].

$$\Pr(N_c = n_\kappa) = \begin{cases} \frac{\lambda!}{(\lambda - n_\kappa)!} \frac{\alpha(n_\kappa, N_\ell)}{\lambda^{N_\ell}} & \text{for } n_\kappa \in \{1, 2, ..., \min(\lambda, N_\ell)\} \\ 0 & \text{else} \end{cases} \tag{5.10}$$

where $\alpha(n_\kappa, N_\ell)$ is the Stirling number of the second kind *i.e.*:

$$\alpha(n_\kappa, N_\ell) = \frac{1}{n_\kappa!} \sum_{i=1}^{n_\kappa} (-1)^{n_\kappa - i} \binom{n_\kappa}{i} i^{N_\ell}$$

In the attack's case the values N_c and N_ℓ are known but λ must be determined. To estimate λ a maximum likelihood estimator $\hat{\lambda}$ is built as a function of n_κ and N_ℓ, *i.e.*, equation (5.10) is computed for all the values $\lambda_i \geq 2^{\lceil \log_2(n_\kappa) \rceil}$ and amongst them the λ_i with the maximum probability is assumed to be correct:

$$\hat{\lambda} = \arg\max_{\lambda_i} \Pr(N_c = n_\kappa | \lambda_i) \tag{5.11}$$

The above Hamming weight detection method was simulated for nibbles and bytes. A given number N_ℓ of randomly generated multiple bit-reset faults were injected into a randomly generated variable x and the number of different faulty values N_c were used to determine the Hamming weight of the variable using formula (5.11). The total number of successfully determined Hamming weights was recorded for 10^5 trials and the success rate was computed for each number of faults N_ℓ as shown on Figure 5.5. On the

[1] The attack is also working with the multiple bit-set fault model.

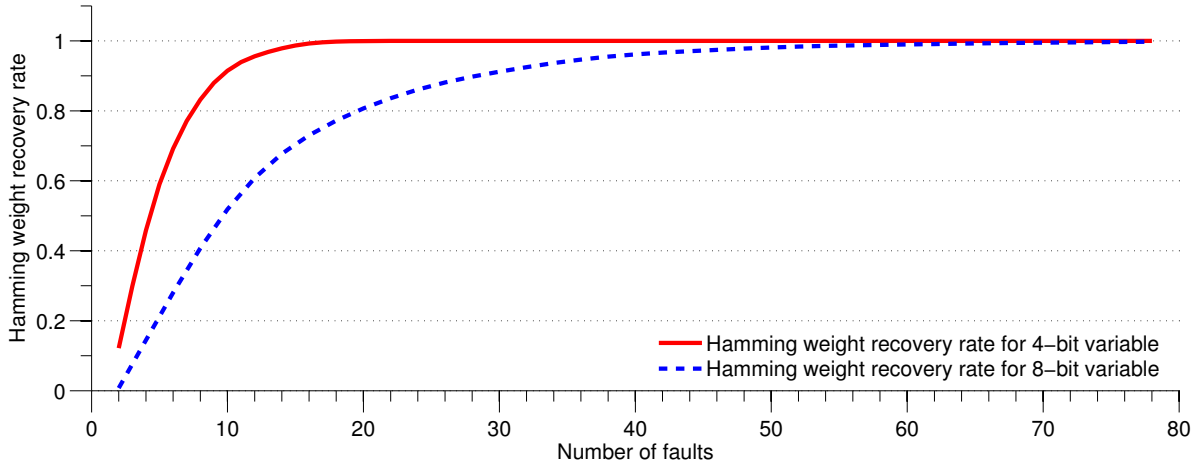


Figure 5.5 – Results of Hamming weight computation by fault injection.

average 15 faults sufficed to detect the Hamming weight of a nibble with a 99% probability. Determining bytes with the same probability required 62 faults.

The occupancy problem and various estimators were previously discussed in [BF93], however the maximum likelihood estimator was chosen due to the limited number of possible bins and computational simplicity.

5.3.2 Key Search

An attacker has M pairs of Hamming weights $\left(\mathrm{HW}(S_{j,i}^{[r]\mathrm{D}}), \mathrm{HW}(S_{j,i}^{[r+1]\mathrm{S}})\right), i \in [1, M]$ (i indicates an encryption index) received for the same operation $S_{j,i}^{[r+1]\mathrm{S}} = \mathbf{S}_j^{[r+1]} \circ \mathbf{A} \mid_{k_j^{[r]}} \left(S_j^{[r]\mathrm{D}}\right)$.

In the following part, the notations $\left(h_i^{[r]}, h_i^{[r+1]}\right)$ and $\left(\mathrm{HW}(S_{j,i}^{[r]\mathrm{D}}), \mathrm{HW}(S_{j,i}^{[r+1]\mathrm{S}})\right), i \in [1, M]$ are interchangeable.

The key search process is performed in two steps. The first step, called *key sifting*, is a typical equation-based approach when the key has to satisfy a set of finite field equations. The present attack uses M pairs $\left(h_i^{[r]}, h_i^{[r+1]}\right)$ to find the key candidates that satisfy the following constraint:

$$\mathcal{L} = \left\{ k \in \mathbb{F}_{2^b} | \forall i \in [1, M] \; \exists x \in \mathbb{F}_{2^b} : \mathrm{HW}(x) = h_i^{[r]}, \; \mathrm{HW}(\mathbf{S}_j^{[r+1]} \circ \mathbf{A} \mid_k (x)) = h_i^{[r+1]} \right\} \tag{5.12}$$

Reducing $|\mathcal{L}|$ requires a significantly higher number of pairs than ciphertext- or plaintext-based attacks.

To perform second step, called *key likelihood estimation*, the HWPD

$$\mathrm{Pr}_k \left[\mathrm{HW}(x), \mathrm{HW}\left(\mathbf{S}_j^{[r+1]} \circ \mathbf{A} \mid_k (x)\right) \right]$$

is pre-computed for each key value $k \in \mathbb{F}_{2^b}$ and uniformly distributed $x \in \mathbb{F}_{2^b}$. During this step the probability distribution function is computed for the list of obtained Hamming weight pairs $\mathrm{Pr}_r \left[h_i^{[r]}, h_i^{[r+1]} \right]$. Then the Euclidean distance between the Pr_r and Pr_k is computed for each key from the list $k \in \mathcal{L}$:

$$\Delta(\mathrm{Pr}_r, \mathrm{Pr}_k) = \sqrt{\sum_{\forall h_i, h_j} \left(\mathrm{Pr}_r \left[h_i, h_j\right] - \mathrm{Pr}_k \left[h_i, h_j\right]\right)^2} \tag{5.13}$$

and the key candidate with the minimum distance is betted as correct:

$$\hat{k} = \arg\min_k \Delta(\mathrm{Pr}_r, \mathrm{Pr}_k) \tag{5.14}$$

Table 5.1 – Specification of the operation $\mathbf{S}_j^{[r+1]} \circ \mathbf{A} \mid_{k_j^{[r]}} \left(S_j^{[r]}\right)$ for different ciphers.

Cipher	Exact operation	Size of $S_j^{[r]\mathrm{D}}$, $S_j^{[r+1]\mathrm{S}}$, and $k_j^{[r]}$	Number of elements in the S-box
LED	$S_j^{[r+1]\mathrm{S}} = \mathbf{S}\left(k_j^{[r]} \oplus S_j^{[r]\mathrm{D}})\right)$	4-bit	16
AES	$S_j^{[r+1]\mathrm{S}} = \mathbf{S}\left(k_j^{[r]} \oplus S_j^{[r]\mathrm{D}})\right)$	8-bit	256
SAFER++	$S_j^{[r+1]\mathrm{S}} = \mathbf{S}\left(k_j^{[r]} + S_j^{[r]\mathrm{D}})\right)$	8-bit	256

Table 5.2 – Number of faults used to recover a key from the operation $\mathbf{S}_j^{[r+1]} \circ \mathbf{A} \mid_{k_j^{[r]}} \left(S_j^{[r]}\right)$ for different ciphers.

Cipher	Average number of plaintexts	Average number of faults per plaintext	Total number of faults
LED	50	40	2,000
AES	250	120	30,000
SAFER++	200	120	24,000

5.3.3 Simulations

The key search algorithm was simulated for LED, AES and SAFER++. Operation (5.5) of each cipher is described in the Table 5.1. Note that LED and AES use bit-wise exclusive or as key mixing operation, while SAFER++ applies byte addition. SAFER++ uses two kinds of S-boxes based on discrete logarithms and exponentiation. In the present research work the logarithm-based S-box with 256 elements was tested.

Key search was performed with known Hamming weights $\left(\mathrm{HW}(S_{j,i}^{[r]\mathrm{D}}), \mathrm{HW}(S_{j,i}^{[r+1]\mathrm{S}})\right)$ that could have been recovered at an earlier step by fault injection. The successful key recovery was recorded after key sifting and key likelihood estimation and shown in the Figure 5.6 for the various ciphers considered.

As illustrated on Figure 5.6, key likelihood estimation significantly improves key recovery success rate. Moreover, key sifting does not converge to 1, which justifies the usage of the key likelihood estimation step. The average number of Hamming weight pairs needed to recover the correct key with 99% confidence is 50 for LED, 250 for AES and 200 for SAFER++.

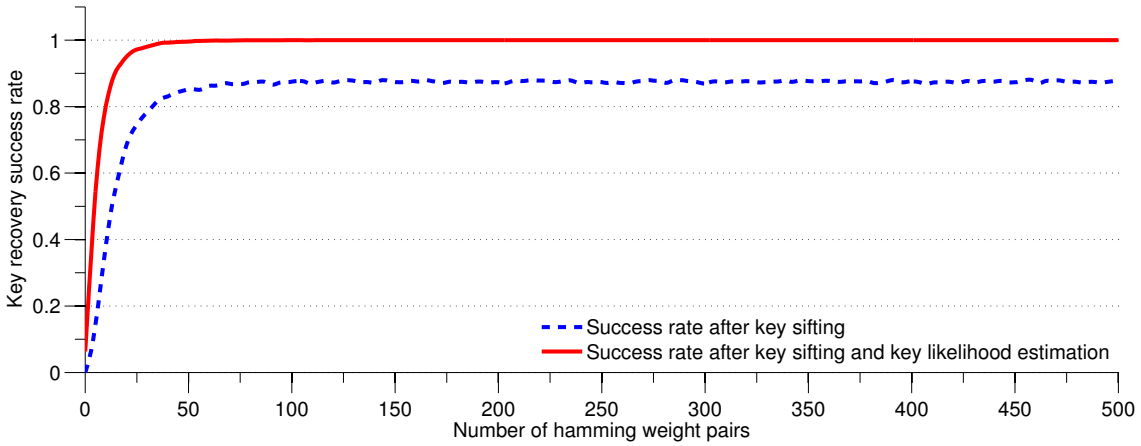
Simulations reveal that our attack can be used to recover round keys. The total number of required faults, given in Table 5.2, depends on the cipher's S-box input size, key mixing operation and the number of elements in S-box.

5.4 Substitution Layer Leakage

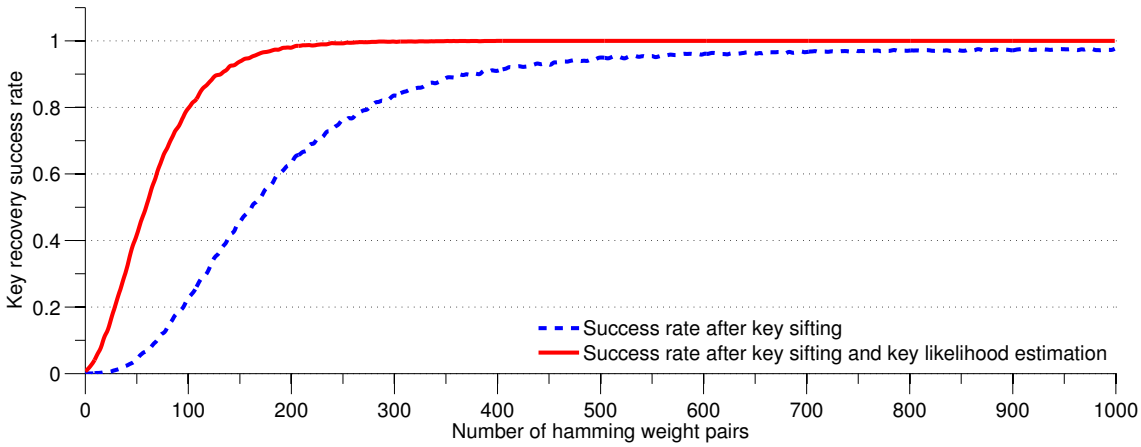
This section discusses practical aspects of the proposed fault attack, namely multiple bit-set or bit-reset fault models, precise fault injection time and the required number of faulty ciphertexts.

Multiple Bit-Set or Bit-Reset Fault Model.

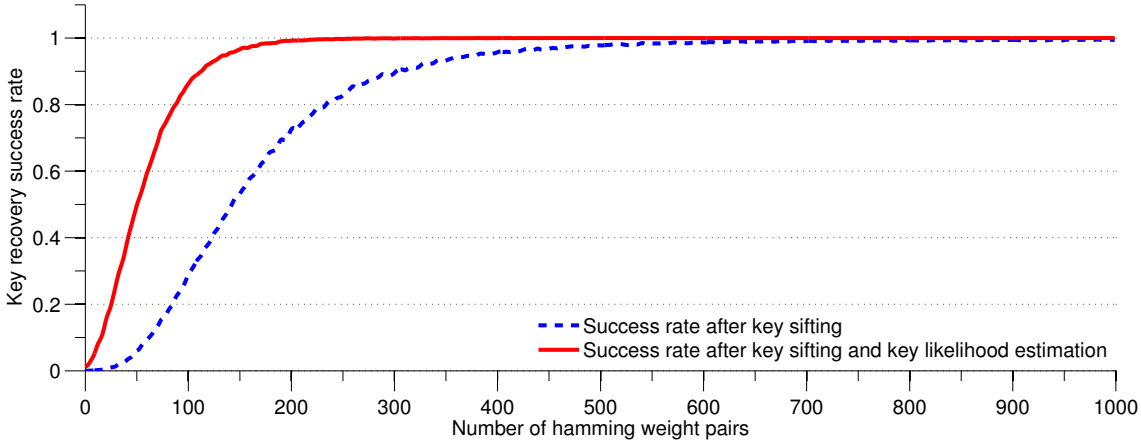
One of the attack's main assumptions is that a multiple bit-reset (or multiple bit-set) can be caused by fault injection. Previously, these fault models have been applied in various papers [BS03, FJLT13]. The practical feasibility of bit-reset (or bit-set) fault injection was shown in a set of experiments. The multiple bit-set fault model was observed during EM-glitch fault injection as described in [MDH+13]. [RSDT13] reports that during laser fault injection to SRAM, bit-flip fault model is irrelevant, only bit-set (or bit-reset) errors are feasible.



(a) LED



(b) AES



(c) SAFER++

Figure 5.6 – Key recovery success rate for different S-boxes $\mathbf{S}_{r+1,j} \circ \mathbf{A} \mid_{K_{r,j}} \left(X_{r,j}^{SP}\right)$.

Precise Fault Injection Time and Space.

A second attack requirement is precise fault injection, *i.e.*, the time and the location of a fault must be well specified. This is a single fault attack since only one fault has to be injected during an encryption. The identification of fault injection time and place can be done during the characterization phase when the adversary has the full control over the device. To identify the processing time of the state value $S_j^{[r]}$ an adversary may use side-channel based reverse engineering techniques as shown in [JT12, Nov03, GP08]. Once the time is identified the adversary can search a location for EM or laser fault injection. To speed up the identification phase the number of cipher rounds can be reduced. Note that during the characterization phase an adversary may have access to the cipher's input and output but during the attack this information is not accessible; hence, standard fault injection or side-channel attacks cannot be applied.

The Number of Faults.

Simulation shows that approximately 120 fault injections are needed before the Hamming weights of the 8-bit input and output state values can be identified. This number of faults has to be multiplied by the number of plaintexts required to recover the key value, *i.e.*, the number of Hamming weight pairs needed for key byte recovery. In total approximately 30,000 faults have to be injected before the correct key byte value can be found for the AES and 24,000 faults for SAFER++. This number of faults is significantly higher than for other FAs. However, our attack targets scenarios where other FA methods cannot be applied. The increased number of faults seems a reasonable price to pay. Once the time and location of fault injections are identified, it is just a matter of time to create this number of errors. In addition, our attack targets individual nibbles or bytes depending the S-box sizes. After recovering several bytes with our method, the rest of the key can be brute-forced.

Countermeasures.

Our attack is feasible against implementations where the number of different ciphertexts can be counted while their actual values remain inaccessible. The most straightforward countermeasures against the presented method are based on randomization. Infective countermeasures that replace the ciphertext by a random number after a fault detection is one of them [LRT12]. This countermeasure outputs the correct ciphertext C if no fault happens and when a fault is injected the output is masked with random data $\check{C} = C \oplus ((C \oplus \tilde{C}) \cdot \eta)$ where η is a random number. With this countermeasure, for the same fault injected, there are multiple different faulty ciphertexts and thus Hamming weight computation cannot be applied.

Another type of countermeasures, which is often used to defeat side channel analysis, is masking [KHL11]. In this case, the operation $F = \mathbf{S}_j^{[r+1]} \circ \mathbf{A} \mid_{k_j^{[r]}}$ is changed to $F_\eta = \mathbf{A} \mid_{f(\eta)} \mathbf{S'}_j^{[r+1]} \circ \mathbf{A} \mid_{k_{j\oplus\eta}^{[r]}}$ where η is the randomly generated mask, **S'** is the new layer computed as a function of the mask and $\mathbf{A} \mid_{f(\eta)}$ is the unmasking operation. However, the states $S_j^{[r]\mathrm{D}}$ and $S_j^{[r+1]\mathrm{S}}$ change masks for each encryption, thus the Hamming weight can not be determined.

5.5 Conclusions

In this chapter a new fault attack on SPN ciphers was described. This attack has conservative preliminary assumptions. Namely, the adversary:

- does not know plaintext and ciphertext values.
- can encrypt several times a set of unknown plaintexts.
- knows the number of tampered encryptions performed for the same plaintext.
- can induce a multiple bit-reset (or bit-set) fault in a middle SPN round.

It is shown that under these assumptions the adversary can derive the Hamming weight of an internal round state. When a Hamming weight of the state before key mixing operation and a Hamming weight of the state after confusion operation are known the round key can be recovered. To the authors' best knowledge this is the first fault attack that can be used to derive any round key. Also this is the first attack based on Hamming weight of the internal state values.

Simulations confirm that our attack works in practice against AES, LED and SAFER++.

CHAPTER 6

MULTI FAULT ATTACKS ON PROTECTED CRT-RSA

Summary

Since the first publication of a successful and practical two-fault attack on protected CRT-RSA surprisingly little attention was devoted by the research community to an ensuing new challenge of multiple fault injection. The reason for it seems to be two-fold. One is that generic higher order fault attacks are very difficult to model and thus finding robust countermeasures is also difficult. Another reason may be that the published experiment was carried out on an outdated 8-bit microcontroller and was thus not perceived as a threat serious enough to create a sense of urgency in addressing this new menace.

This chapter presents two-fault attacks on protected CRT-RSA implementations running on an advanced 32-bit ARM Cortex M3 core. To the author's best knowledge, at the time of publication this was the first practical result of two fault laser attacks on a protected cryptographic application. Considering that laser attacks are much more accurate in targeting a particular variable, our result suggest that for protecting security-crucial applications software-only countermeasures may need to be complemented by some hardware-based protection techniques such as shields, sensors, error correction methods, etc.

6.1 The State-of-The-Art

Most papers develop theoretical attacks: typically they start by specifying a fault model and go on describing how it can be used to break a cryptographic primitive. Sometimes simulations are used to verify the success of the attack model, but practical experiments are rarely reported. Papers that describe breaking real-life cryptosystem implementations exist, and they often play the role of a catalyst - for both, attacks and countermeasures. We hope that this thesis falls into this category. We describe practical multi-fault laser attacks on protected against single fault cryptographic implementations, which are running on a powerful general purpose ARM-based microcontroller.

The only practically implemented second-order fault attack published in the open literature [KQ07a] was carried out on a simple 8-bit microcontroller. Faults were generated by means of power glitches. Considering that laser attacks are much more accurate in targeting a particular variable, the significance of our result in attacking general-purpose 32-bit microprocessors cannot be overlooked. We describe attack techniques to provide a reader with useful insights for understanding of threats of multi-fault attacks and their impact on the implementation of cryptographic algorithms.

Until now the publication of Kim and Quisquater [KQ07a] is the only work describing a practical implementation of a multi-fault attack on CRT-RSA in which they were able to break first order countermeasures [CJ05]. The attack was implemented on Atmel's 8-bit AVR microcontroller ATMega. The first countermeasure is based on the Montgomery powering ladder which works on a pair of intermediate results of the form $(m^{[k-1]}, m^{[k]})$ and uses a relationship between them to perform a coherence check on the computed value. The second countermeasure is an "infective" generalization of Shamir's trick initially presented in [Sha97]. In the generalized countermeasure CRT computations are carried out modulo $p \times r_1$ and $q \times r_2$ (r_i being a small random value), which allows to verify the result modulo r_1 or r_2 afterwards, and, if an error is detected, to infect it using a specially computed random value [JPY01].

$$\begin{aligned} S_p^* &= m^{d_p} \mod (r_1 \cdot p) & S_1 &= m^{d_p} \mod \phi(r1) \mod r_1 \\ S_q^* &= m^{d_q} \mod (r_2 \cdot q) & S_2 &= m^{d_q} \mod \phi(r2) \mod r_2 \end{aligned}$$

Both half exponentiations are check separately before the CRT recombination:

$$\text{If } S_1 = S_p^* \mod r_1 \text{ and } S_2 = S_q^* \mod r_2 \text{ return } S = CRT(S_p^*, S_q^*)$$

After the feasibility and the danger of the new class of attacks was demonstrated, there were very few papers dedicated to their theoretical models and countermeasures. Notably, the two new second-order countermeasures for CRT-RSA proposed in the original paper [KQ07a] were found to be weak and were subsequently improved in [KQ07b]. In [BDL97] a second-order resistant infective method for CRT-RSA was suggested, but as shown later, it can be broken with a single fault. The most serious analysis of a second-order fault attack resistance for CRT-RSA was published recently in [DGRS09] where the countermeasures of [KQ07b] were analyzed from an implementation standpoint, revealing their potential vulnerability and, more importantly, a claim was made on a general second-order countermeasure. This will be addressed at the end of this chapter.

The question that begs to be asked is – *are second-order fault attacks relevant in real life? Can they be implemented in practice on modern high performance devices?* Our work allows us to answer this question with full confidence – *yes, they can*.

The rest of this chapter is organized as follows. Section 6.1 recalls the notion of fault models. Section 6.2 and 6.3 detail the setting of a multi-fault attack: the device architecture and a laser bench, respectively. Section 6.4 briefly describes a preparation process for a front side laser attack. Section 6.5 recalls two new countermeasures for CRT-RSA from [BHT10]: a very efficient signature verification method for protection against single-fault attacks and its extension to a second-order countermeasure by combining it with infective computations. Implementations of these algorithms on a 32-bit ARM Cortex M3 microcontroller were used as a test bench for our practical multi-fault laser attacks described in details in Section 6.6. In conclusion we summarize an impact of this experiment on security engineering.

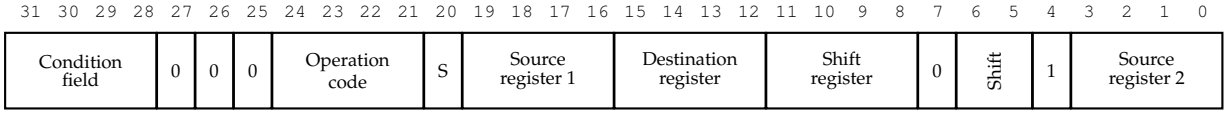
6.2 Device Architecture

Feasibility of multi-fault attacks on cryptographic algorithms running on modern high performance architecture is a challenging problem with a serious practical impact. For our experiments we had chosen a general purpose microcontroller based on a 32-bit ARM Cortex M3 core – a CPU specially developed for embedded applications [ARM] and widely used in medical equipment, PC peripherals, industrial applications, alarm systems, etc.

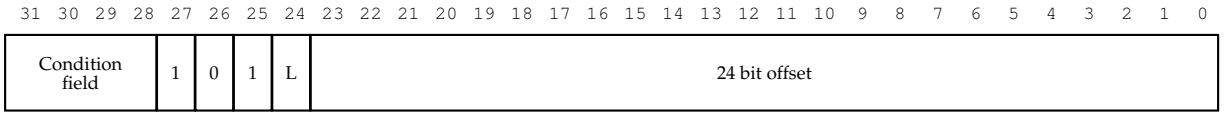
The microcontroller is implemented as a System on Chip (SoC) and includes 512 KB of embedded flash and 64 KB of embedded SRAM, both memories being accessed in one clock cycle. Many peripherals, analog and digital, are present on the chip including I2C and USB interfaces, DMA, ADC, PLL, an internal RC oscillator, independent watchdogs and timers. The SoC is manufactured using 130nm feature size and six metal layers. All digital components are implemented in glue logic.

The device has a number of features intended for robustness in adversarial environmental conditions, *e.g.*, temperature sensors, embedded voltage regulator, etc. A Programmable Voltage Detector generates an interrupt if the voltage drops below a predetermined threshold. In this case, it is up to the firmware to implement a safe shutdown before the reset. The clock safety system generates an interrupt if the main external clock is disconnected or broken and the microcontroller is automatically clocked with an internal safe clock so the system can perform shutdown or a reset operation. Thus some low cost attacks such as setup time violation [BBPP09] or clock glitches [ADN+10] were not possible.

The ARM architecture is based on a (modified) Harvard model [ARM] which separates code and data and thus can simultaneously read them both from (distinct) memory address spaces. It uses different buses for instruction and data signals. The Cortex M3 is characterized by a three stage pipeline unit (instruction fetch, decode, execute/address compute) and normally does not include caches. In the User mode, Cortex M3, which is a load-store machine and does not support operations directly from the main memory, can access 16 general purpose 32-bit registers. Three of them have specific roles, namely: r_{13} is dedicated to a stack pointer, r_{14} is a link register, and r_{15} is a program counter (PC). There is also a set of registers representing a program status. These registers, called xPSR, store all the flags which are used for conditional statements and exception checking. The relevant flags stored in these registers, as far as control flow management goes, are: `Z` (zero result from ALU flag), `N` (negative result from ALU flag), `C` (ALU operation carried out), and `V` (ALU operation overflowed).



(a) Data processing and shift instruction



(b) Brunch and branch-link instruction

Figure 6.1 – Instruction encoding in ARM architecture.

Fig. 6.1 is an excerpt from the ARM reference manual [ARM, Yiu] and depicts two most common instructions: a general purpose data processing instruction and a branch. The instruction encoding in ARM architecture uses different fields to distinguish among possible instructions. The first field (bits 27 to 25) is used in order to split an instruction set into macro groups (data processing, load/store, branches and co-processor dedicated instructions). Data processing instructions are further characterized by a 4-bit opcode (bits 24-21) which specifies the actual instruction to be performed.

Most of instructions can be executed conditionally, *i.e.*, the instruction is committed only if a particular condition is met, for example if the result of the previous instruction is zero or negative. As one can see in Fig. 6.1 instructions have a 4-bit wide condition field which is checked at the time of execution. If the condition is unmatched, the instruction has the same effect as `NOP` without any rollback penalty. Conditional branches are thus implemented as simple conditional instructions distinguished by the 3-bit instruction class code 101 (bits 27-25) and nothing more. A special condition code (1110) is the one

matching the "Always" condition and allows the normal execution of instructions. Thus, due to the nature of the ARM architecture by injecting faults in registers we may:

- Modify the condition field of the instruction which may result in an instruction skip or a mistaken branch;
- Modify the instruction itself, *e.g.*, `ADD` instead of `SUB`;
- Alter a value on which the condition is checked: this will result in a wrong branch evaluation;
- Modify the program counter register r_{15};
- Change a destination address or a link register of a branch thus altering program flow.

The Cortex M3 processor supports several exception fault handlers. These handlers can detect faults resulting from an error condition in instruction execution such as:

- Memory management fault, generated when program jumps to memory area where there is no executable code;
- Accessing privileged operating system functions;
- An attempt by the processor to execute an undefined instruction;
- An important exception type is a hard fault which occurs because of an error in exception processing or because an exception cannot be managed by any other exception mechanism. Hard faults were regularly observed during our experiments.

6.3 The Laser Equipment

A laser platform which we used in our experiments was equipped with the YAG laser shown on Fig. 6.2, a driving board on which a chip is mounted, a synchronization board which controls timing of the laser shots, two cameras with separate monitors, a cooling/heating system, a LeCroy oscilloscope with 10 GHz bandwidth, and a personal computer. Two wavelengths, 532 nm green and 1064 nm infrared, can be generated by the laser.

For the green laser $5\times$ and $20\times$ magnification lenses are available, and for the infrared $50\times$ and $100\times$ magnification lenses can be installed. The laser aperture is about 35×35 μm, and it can be reduced by shutters to 1% of this area. The duration of the laser shot is fixed at 5ns. The laser's aperture, energy, displacement step and other parameters can be set up either manually or using the LabView interface available on the computer.

The target chip is mounted on a driving board used to relocate its position for shooting. The step of the driving table for both Y and X axes is equal 1μm. To target different timings of a program execution, a synchroboard delays a trigger signal from the chip. The delay ranging from 10ns to hundreds of ms, can be set via the LabView interface which also allows handling the start and end points of the driving board and a displacement step as well as the number of shots in each position.

A driving table, a synchroboard and a chip are connected to a computer. It is also possible to connect the laser terminal and an oscilloscope to the computer, but during our experiments these connections were not established. The bench is controlled via a custom-designed LabView project. This software runs under Windows XP.

To establish proper communication with a chip, one needs a special DLL library which opens the connection on a serial port, an initialization file which specifies all parameters of the connection, such as a port number and time before closing, and a scenario file, which determines the commands sent to a chip. All projects for the chip (including crypto-libraries) were developed using KEIL tool-chain. We also used special firmware containing handlers, exceptions and other functions.

To reduce the communication time between the computer and a chip, initially all data for attacked algorithms (input parameters, cipher keys, plaintexts, etc.) were hard-coded. A program installed on a chip was running in an infinite loop, and when the specific command from a computer arrived, a cryptographic algorithm was launched. After finishing the algorithm, the program returned to the initial infinite loop waiting for the next command.

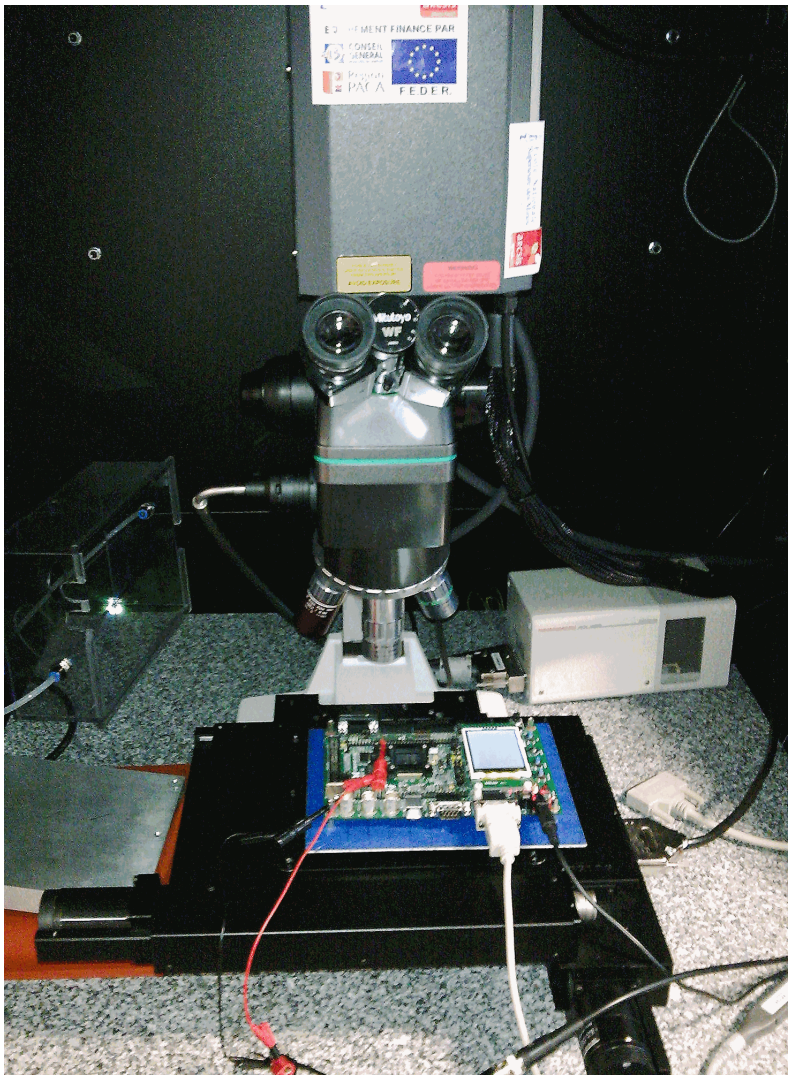


Figure 6.2 – Laser platform.

An algorithm according to which the bench is typically run is as follows:

1. The synchroboard is ready, the laser is charged, the chip is running and waiting for a command from the PC.
2. The PC sends a command (*e.g.*, start the execution of cryptographic algorithm) to the chip.
3. The chip starts execution and at some moment of time it raises up a trigger.
4. The trigger is recognized by the synchroboard.
5. The synchroboard generates another trigger for the laser with a predefined delay.
6. The laser receives the trigger and after the delay it shoots.
7. The chip outputs the result to a serial port.
8. This result is recorded by the computer.
9. The computer maintains the current state of the experiment, *i.e.*, it moves the driving board, changes the delay of the synchroboard if necessary and does other routines specified in the attack scenario.
10. While the driving board did not reach the end point, go to step 1.

This algorithm provides a good flexibility and allows the user to master a shot location and its time. The user can easily change an attack scenario or bench parameters via a LabView interface. The bench records all erroneous results together with exact bench settings when the error occurred; thus the attacker can

infer for example timing information, type of errors, etc. which in turn allows her to repeat or fine-tune an experiment.

6.4 Preparatory Steps

For modern SoC front side laser attacks face severe limitations with the growing complexity and use of multi-level metals. In recent publications dedicated to front side laser attacks authors attacked experimental ASIC containing simple IP, often just one crypto core [LPM+06, LAM+07, MRL+06]. To the best of the author's knowledge, practical laser attacks against modern multi-layered SoC with embedded memories and powerful processors had not been presented in the open literature before this result was published in 2010 [TK10].

The front side laser attack is usually performed with a green laser, and the back side with an infrared laser. Given a particular wavelength, selecting an energy level and a size of the aperture depend on the chip and attack scenario. An important parameter is focus. When the laser is focused, all its energy is concentrated at a very small spot, typically covering few gates [GPLL09]. Depending on the goal, an attacker can use either focused or defocused laser. With the focused laser the fault injection can be done very precisely, while an advantage of the defocused laser is that it covers a larger area, so it can be used for a preliminary scan of a SoC to identify vulnerable spots.

We describe an attack on a front side of the chip because some of the devices come in a BGA package which make back side attacks impossible without device modifications. To perform a laser attack an attacker needs to de-package the chip which can be done using, *e.g.*, JetEtch II decapsulation system.

After having done it, we discovered the first obstacle: the SRAM and flash areas were completely covered by metal tiles (see Fig. 6.3), thus no direct access to memory cells was possible. To investigate a possibility to induce any fault at all, the overall chip area was scanned many times with a minimal displacement step, and at every position several shoots were made. The microscope lens, its focus and energy levels varied from one scan to the next because these parameters have an influence on the laser's effect on the IC.

Initially a digital logic area of the SoC was selected for an attack. The assumption was that it could be possible to create a fault in the CPU logic or in registers. The scan has proven to be unprofitable. Higher energy levels burned the chip, so we had to use the minimum energy. We run experiments with both, focused and defocused laser. The aperture varied from 2% to 25%. But the result was the same – whenever the laser shot, the chip either crashed or did not return any error. Five chips were destroyed while shooting logic. The conclusion from this experiment was that inducing useful faults by shooting this area is unfeasible.

SRAM and Flash were selected as the next targets. As discussed earlier, these areas are covered by metal tiles. Although the opacity of these zones is very low, we hoped that it would be possible to penetrate the gap between the tiles. However no experiments with shooting memory cells caused errors for all tried energy levels and aperture sizes, so the conclusion was similar: inducing errors by targeting SRAM and Flash areas is unfeasible.

The only remaining part was an area between the SRAM and Flash. This zone was not covered with metal tiles; instead it was densely populated with connections highlighted by the red rectangle on Fig. 6.3. As previously, we scanned the whole zone with an defocused laser and discovered that different exceptions were raised during shooting. Hence we concluded that the laser can cause perturbations in the microcontroller's functionality. The next step was to find laser parameters causing computational errors that remain undetected by the microcontroller.

After few days of experiments such parameters were found. We do not know the exact functionality of the attacked area; one of the hypotheses is that these are buses and power tracks on the top layer, registers, buffers, and decoders beneath. In any case the attack proved effective: once we mastered the location and the timing, it became possible to achieve exploitable faults with very high repetition rate.

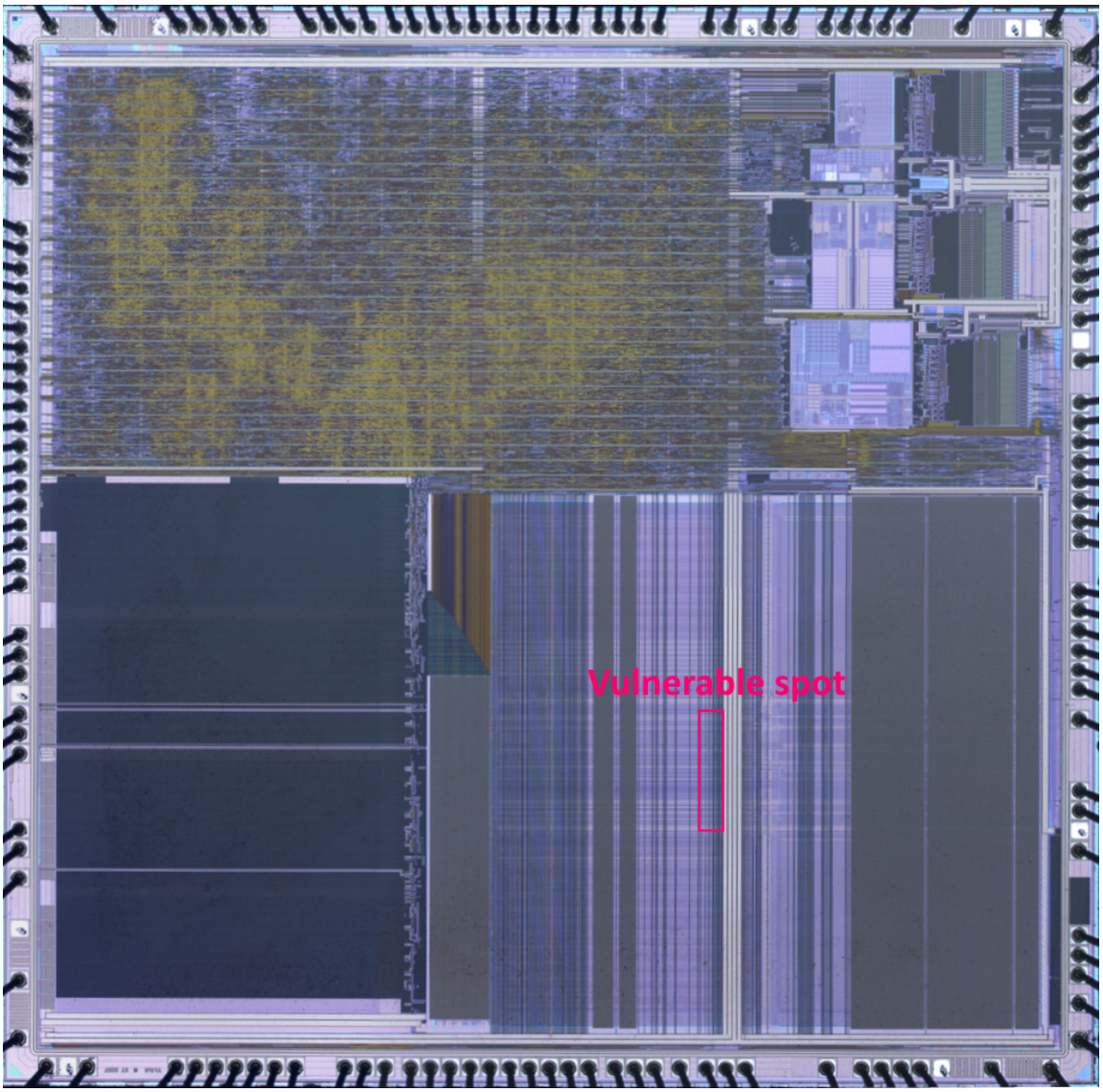


Figure 6.3 – Top layer of the chipset.

6.5 Two Fault Attacks Against CRT-RSA

The first fault attack on CRT-RSA was presented in the seminal paper [BL96] where it was pointed out that by injecting a single random fault into the computation of either s_p or s_q, a secret exponent can be calculated by subtracting faulty and correct signatures and then computing the GCD of the result and the modulus N. Some RSA implementations do not allow signing the same message twice; nevertheless, as shown by [Len96] an enhancement of the attack described in [BL96], these implementation can be still broken as an attacker needs only an incorrect signature and an initial message.

Due to performance advantages of CRT-RSA and its high vulnerability to fault attacks, securing its implementation is an important and challenging task, attracting a lot of attention. We can distinguish three main approaches. The first is an introduction of some redundant computations to perform internal coherence checks aiming at verifying the result before returning it. The most known countermeasure of thus type is proposed by Shamir and adapted for a CRT-RSA in many papers, *e.g.*, [ABF+03, KQ07a]. The second approach consists in performing a consistency check directly at the exponentiation algorithm itself, using Montgomery powering ladder; this method was suggested by Giraud [Gir05b] and enhanced in [BHT10].

A general feature of these countermeasures is checking if some relation exists between two values. To avoid the if-branching inherent to comparisons, Yen et al [SMKLM02] introduced a concept of *infective*

Algorithm 5 Protected CRT-RSA signature generation

Require:

p : first secret prime number;
q : second secret prime number;
e : public key component, such that GCD $(e, (p-1)(q-1)) = 1$;
d : private key component $d = e^{-1} \mod (p-1)(q-1)$;
i_q : modular inverse of q, *i.e.*, $i_q = q^{-1} \mod p$;
$m \in \mathcal{M}$: a padded message to be signed;

Ensure:

A signature $s \in \mathcal{S}$;

1: $d_p = d \mod p-1$
2: $d_q = d \mod q-1$

3: $s_p = m^{d_p} \mod p$
4: $s_q = m^{d_q} \mod q$
5: $s = s_q + q\left(q^{-1}(s_p - s_q) \mod p\right)$ ▷ Signature s of the message m

6: $e_p = d_p^{-1} \mod p-1$
7: $e_q = d_q^{-1} \mod q-1$
8: $m_p = s^{e_q} \mod p$
9: $m_q = s^{e_p} \mod q$
10: $m' = q\left(i_q(m_p - m_q) \mod p\right) + m_q$ ▷ Re-computation of the message m for verification

11: **if** $m' == m$ **then** ▷ Comparison of the initial and recomputed messages
12: return s
13: **else**
14: return error
15: **end if**

computations, which can be applied for both aforementioned methods.

The third, and simplest, countermeasure against fault attacks for any signature scheme is signature verification following the signature generation. To implement this countermeasure For CRT-RSA, a public exponent e must be accessible during computations, which may not be the case in some implementations. Therefore, more sophisticated methods were developed which do not require the knowledge of e; instead its value can be obtained from the available parameters d_p and d_q. For our attacks we had chosen a new algorithm published in [BHT10] and presented below as Algorithm 5.

Algorithm 5 is much more efficient than other verification methods where a public exponent e is assumed to be known or embedded as in [Joy09] or computed as in [ABF+03] because it is carried out on half-sized data. For a short public key, the values e_p and e_q are usually equal to e hence the countermeasure does not significantly increase the overall time of a normal signature operation. But as other first order countermeasures broken in [KQ07a], it is based on a comparison of two values and thus does not protect the system against second order fault attacks.

Algorithm 6, given in [BHT10], combines the conditional check and the infective method. The authors claim that with this algorithm if the conditional check at Step 9 is bypassed, the output value will be random and useless for recovering private exponent.

This countermeasure, however, can be broken with a single fault. Indeed, consider a situation when an attacker can skip just the computations of the sum $s + s_p$ at the step 12. All other operations have been performed correctly, so the resulted signature is $s' = s - (s \mod p) + s_q - (s \mod q)$. In this case the difference between the correct and the faulty answers will be the following:

$$\Delta = s - s' = s - s + (s \mod p) - s_q + (s \mod q) = s_p$$

Algorithm 6 Second order resistant CRT-RSA signature generation with infective method

Require:

p : first secret prime number;
q : second secret prime number;
e : public key component, such that GCD $(e, (p-1)(q-1)) = 1$;
d : private key component $d = e^{-1} \mod (p-1)(q-1)$;
i_q : modular inverse of q, *i.e.*, $i_q = q^{-1} \mod p$;
$m \in \mathcal{M}$: a padded message to be signed;

Ensure:

A signature $s \in \mathcal{S}$;

1: $d_p = d \mod p-1$
2: $d_q = d \mod q-1$

3: $s_p = m^{d_p} \mod p$
4: $s_q = m^{d_q} \mod q$
5: $s = s_q + q\left(q^{-1}(s_p - s_q) \mod p\right)$ ▷ Signature s of the message m

6: $e_p = d_p^{-1} \mod p-1$
7: $e_q = d_q^{-1} \mod q-1$
8: $m_p = s^{e_q} \mod p$
9: $m_q = s^{e_p} \mod q$
10: $m' = q\left(i_q(m_p - m_q) \mod p\right) + m_q$ ▷ Re-computation of the message m for verification

11: **if** $m' == m$ **then** ▷ In case the comparison is skipped infect the signature s with redundant data
12: return $s + s_p - (s \mod p) + s_q - (s \mod q)$
13: **else**
14: return error
15: **end if**

Now the value of p can be found:

$$\begin{aligned} \Delta &= s_p \mod p \\ s_p &= \Delta + kp \\ s_p - \Delta &= kp \\ \text{GCD}\,(s_p - \Delta, N) &= \text{GCD}\,(kp, pq) = p \end{aligned}$$

An attack on Algorithm 6 requires skipping an operation. As we discovered, this is not difficult to achieve on Cortex M3 architecture. RSA deals with 512, 1024, and 2048 bit numbers. There is no processor capable of working with such scalability, thus different data structures, such as arrays of integers, are used to handle large numbers, and subroutines are developed to perform operations such as add, compare, etc. on these structures. Hence operations in lines 11-15 of Algorithm 6 are implemented as function calls. In ARM architecture a single assembler instruction corresponding to a function call can be bypassed by a laser shot as described in Section 6.6.

6.6 Practical Issues of Multi Fault Attacks

At first, the attack described in [BDL97] was performed in a "grey box" setting: we augmented the initial cryptographic code with instructions that made an attack easier. For example, to find a time for a fault injection, a trigger was raised at the beginning of the s_p computations and dropped at its end. Using an oscilloscope we observed that the overall computation took about 200 ms when run with an internal clock (for this we simply removed a crystal oscillator from the device board). With an external clock the device runs approximately 9 times faster. Once we knew the timing, we calculated the delay between raising a trigger and the laser shot. It was even possible to use an external trigger (and thus conduct a real black box attack). At one position we fired 20 shots. It took one day to produce about 40 incorrect

results, all of which allowed recovering the prime p hence the attack against an unprotected CRT-RSA was easy.

The next step was to test the ability to induce multiple faults into the protected CRT-RSA (described by Algorithm 5), but the attack was not trivial.

An attack against Algorithm 5 assumes that one fault is induced while computing s_p or s_q and another fault either skips the conditional check in line 11 or flips its outcome. The example below is the snapshot of the assembler code corresponding to lines 11-15 of Algorithm 5.

```
0x08001160 A9FD      ADD      r1,sp,#0x3F4
0x08001162 A82B      ADD      r0,sp,#0xAC
0x08001164 F000F881  BL.W     CompareBig (0x0800126A)
509:                 if (i == 0)     {
510:                  /* write to vector of char */
0x08001168 B930      CBNZ   r0,0x08001178
511:                 W32_to_W8(m.w,P_pOutput,P_pPrivCRTKey->modulus_size);
512:                 } else {
0x0800116A 4639      MOV      r1,r7
0x0800116C F1060008  ADD      r0,r6,#0x08
0x08001170 6822      LDR      r2,[r4,#0x00]
0x08001172 F000FCF8  BL.W     W32_to_W8 (0x08001B66)
0x08001176 E003      B        0x08001180
513:                        printf("Fault␣%d\n",i);
514:                 }
515:          }
0x08001178 4601      MOV      r1,r0
0x0800117A A004      ADR      r0,{pc}+2  ; @0x0800118C
0x0800117C F002FC24  BL.W     __1printf (0x080039C8)
```

Register `r0` keeps the result of the comparison between two large numbers. If these two numbers are equal a zero is returned. The syntax of the Compare and Branch on Non Zero (`CBNZ`) instruction is:

```
CBNZ Rn, label
```

This means that whenever `Rn`$\neq 0$, the program jumps to `label`. This instruction does not change the conditional flags, and is, in other words, equivalent to:

```
CMP Rn, #0
BNE label
```

This observation is important because previous attacks assumed that it is possible to flip the conditional flag bit; however in the ARM architecture this flag bit is not often used as comparisons can be done without it. There are several ways to skip the test in line 11 of Algorithm 5:

- Skip the execution of `CompareBig`; in that case register `r0` may not keep any useful value;
- Force register `r0` to 0 or induce a fault in such a way that the conditional check returns zero in all cases;
- Skip the execution of `CBNZ`; in which case the microcontroller will just increase the program counter thus going to the if-yes branch.

Among the three options, the last one is the most promising. To evaluate the feasibility of condition skipping, `CompareBig` was modified to never return `0` as the result of comparison. Due to this modification, the program's normal behaviour was to print only zeros if $m = m'$ and the only way to get the correct result was to skip the `CBNZ`.

Let us describe in detail how this has been achieved. The laser's advantage is its ability to target a spatial location, as small as a few gates, very precisely. This turns into a disadvantage when an attacker does not know exactly where a targeted cell or a register is located. We started by scanning the whole vulnerable area by the laser with a time step of 10 ns and the minimal displacement step of 100 nm while the chip was running our specially designed assembly code that used the `CBNZ` instruction with the same registers; the objective was to skip at least one `CBNZ`. To pinpoint the exact location where and when `CBNZ` was executed we used a trigger signal. The laser needs a trigger commanding it to shoot

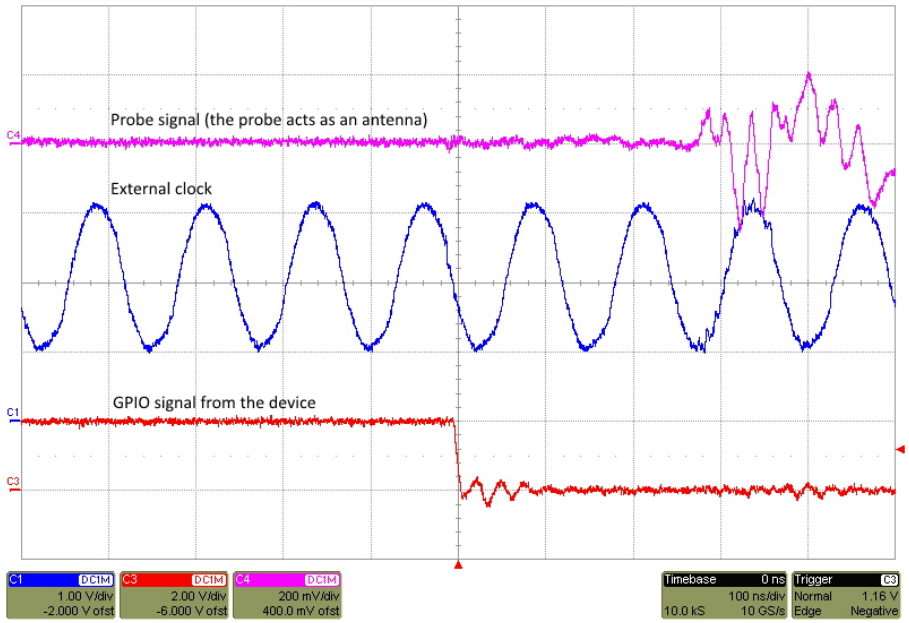


Figure 6.4 – Oscilloscope snapshot taken during instruction skipping.

at a certain delay. A shooting must occur when the targeted instruction is executed, thus we need to find this time precisely. By raising a trigger up somewhere in the program and dropping it down just before the if-statement, so that the targeted `CBNZ` is executed in a clock cycle right after dropping the trigger, we approximated a required delay for a laser shot. If the laser shot skipped the `CBNZ` instruction, the program printed out the correct result, thus we had a means to know if instruction skipping was successful. The laser shot can be seen on the oscilloscope by means of the differential probe which acts as an antenna as illustrated on Fig. 6.4. Hence by correlating program outputs with oscilloscope traces we were able to find the exact spatial positions of the laser shot which resulted in the required fault. It took about one day to find this position.

The next step was to find a precise shot time for the non-customized code. The synchronization was done again by correlating the oscilloscopes traces with the results of computations recorded by the computer. By repeatedly adjusting the trigger delay with a 10ns step, we eventually found the exact shot time targeting the `CBNZ` execution in the non-customized code. The time window for a shot was 190 ns; it is slightly more than one clock cycle, which is equal to 125 ns. During this time window, 9 shots were fired. Three out of the nine shots resulted in instruction skipping. Fig. 6.4 presents an oscilloscope snapshot taken during a successful instruction skipping.

Thus the first objective was achieved: the location and the proper time window allowing the skipping of the `CBNZ` consistently and with high probability was found. The next step was to synchronize a shot with s_p computations.

Unexpectedly a new problem appeared. The laser bench does not provide a possibility to set up different delays for different triggers, *i.e.*, if two triggers are raised up during one program execution then the delays between the time of raising the triggers and the time of shots are exactly identical. Another problem is the need to recharge the YAG laser between two shots. According to the laser's documentation, the minimal delay between shots is 200 ms for the laser to fully recharge. Actually the laser can shoot earlier, but with a lesser energy. Fortunately, shooting with a minimal energy level was sufficient and the interval between shots could be shortened.

Initially we checked that the delay used to skip the `CBNZ` instruction could also cause an error during s_p computations. To do that the chip was programmed with the CRT-RSA without countermeasures, so if this delay allowed inducing such an error the erroneous output would lead to the seminal attack described in [BDL97]. The result was successful; hence it had been proven that with the same delay it was possible to skip the if-condition and to corrupt an exponentiation separately and independently from the other exponentiation.

The final preparation step was to assemble code with all the triggers to attack a protected CRT-RSA. It

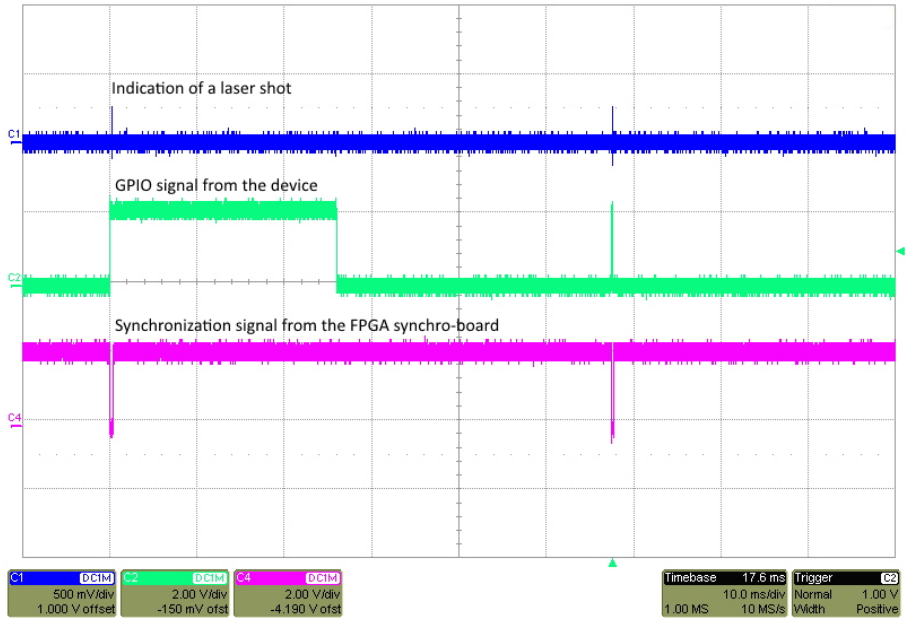


Figure 6.5 – Snapshot taken during two fault attacks.

was done, but unexpectedly this setup did not produce the desired result: it was possible to observe that an error occurred during the computation of s_p because the comparison result was different (sometimes it returned -1, sometimes 1,2 or -2), but it was impossible to skip the `CBNZ` instruction. Several days were spent unsuccessfully searching for other delays and checking all parameters. Finally, we discovered the reason: the time of the shot had been found when there were no errors in the s_p computation. When an error was injected, the computation time of functions such as `CompareBig` changed. This meant that the time of raising the second trigger also changed and thus the shot did not coincide with the time of execution of the `CBNZ` instruction.

To overcome this difficulty, the laser needs a "fixed point" for a trigger. First we changed `CompareBig`'s code so that it runs in constant time, but that did not help. Apparently the execution time of functions, such as long number multiplication, addition, etc., also slightly depends on the processed values. Hence we used a workaround: just before checking the if-condition we inserted some useless but constant-time computations. The trigger was raised at the beginning and dropped at the end of these computations used as a fixed point for the trigger for the second shot.

The new timing was found much faster because the exact location of shooting had been already known. Finally, for this setup it was possible to corrupt the value of s_p and skip the `CBNZ` instruction. We obtained several faulty results needed to recover private exponents. Fig. 6.5 presents a snapshot of the oscilloscope during one of the successful attacks.

Yet another interesting observation: it is assumed that it is difficult to induce exactly the same fault during two experiments. In our case the reality is opposite: because the time window for a two fault attack (which is determined by the delay required for skipping the `CBNZ` instruction) is very small, it is difficult to get different faulty results from the same input. Usually we got the same faulty outputs if we run experiments sequentially.

During the previous experiments one significant power trace was found several times, see Fig. 6.6. On that power trace one can clearly see that the trigger was dropped and immediately raised up again, which should not have happened. Despite a strange trigger behaviour the program terminated without any error. Initially we thought that it was one of unpredictable chip's reactions, but after several such patterns we decided to look at it closely and discovered that the occurrence of this pattern was strongly dependent on the clock cycle, hence strongly dependent on the instruction.

The program's logic is as follows. After the trigger is dropped, the if-condition has to be executed. Resetting a trigger is implemented as a separate function `GPIO_ResetBit` and the last instruction in its code is `bx lr` which returns the program to the main flow. This "branch indirect" instruction uses an `lr` register as an indicator where to branch to. What happens if this instruction is skipped? Logically, the

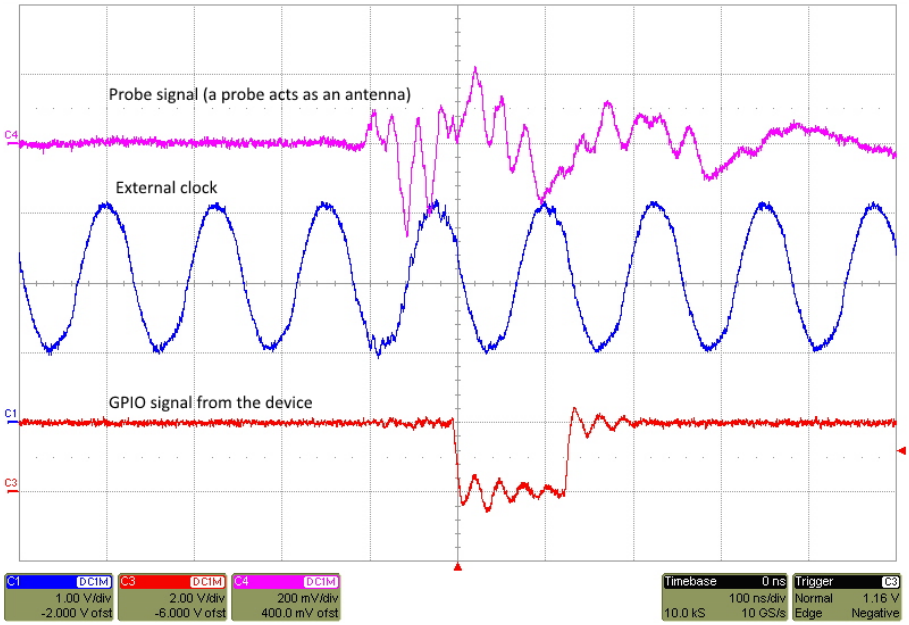


Figure 6.6 – GPIO corruption during laser shots.

program counter is incremented and the program enters into the subsequent function. At the end of this function there is another `bx lr` instruction which should return the program to the main flow.

Assembler code examination revealed that it was compiled in such a way that the function raising the trigger, `GPIO_WriteBit`, was located right after the function `GPIO_ResetBit`. It means that if the return statement in `GPIO_ResetBit` is missed, then the program enters `GPIO_WriteBit` and starts executing its instructions, until new `bx lr` is reached. At one point the program raises the trigger again; that is what was observed on the oscilloscope.

To check this assumption, some additional functionality was added to `GPIO_WriteBit` so that just before the end it printed a small message. We checked that this function was not called from anywhere in the program, thus the only possibility for printing a message was to skip the return statement in `GPIO_ResetBits`. This experiment was repeated several times and the message was often observed confirming that the `bx lr` instruction was consistently skipped.

It is a particularly nice error because it can be used for interesting attacks. Suppose there is a print instruction in a function located next to a key handling procedure. In that case if this procedure's return address is skipped, it is possible to get registers values. By changing a program flow one may skip some critical part or activate other functionalities. This error has been successfully applied to break Algorithm 6 in practice. Indeed, the attack model described in section 6 assumes that an attacker can skip a function call. The following C code is a straightforward implementation of lines 11-15 of Algorithm 6:

```
if (i == 0)
{
  /*temp - is an initial plaintext
  mp and mq - parts of the RSA recombination for the plaintext*/

  GPIO_ResetBits(GPIO_SMART, GPIO_Pin_11);

  /*s = s + sp*/
  AddBig(&s,&sp,&s);

  /*s = s + sq = s + sp + sq*/
  AddBig(&s,&sq,&s);

  /*result = c mod p*/
  ModularReduction(&c,&result,&p);

  /*temp = m - result = m + mp + mq - cp*/
```

```
  SubBig(&m,&result,&temp);

  /*result = c mod q*/
  ModularReduction(&c,&result,&q);

  /*m = temp - result = m + mp + mq - cp - cq*/
  SubBig(&temp,&result,&m);

  W32_to_W8(m.w,P_pOutput,P_pPrivCRTKey->modulus_size);
}else {
  printf("Fault␣%d\n",i);
}
```

Using techniques discussed previously it was not difficult to locate and then skip `AddBig(&m, &mp,&m)` function, because it is called by a simple instruction `BL.W` and instruction skipping was mastered in a previous example. The program then returned a faulty result which was sufficient for recovering the value of the RSA prime p.

Note that the countermeasure infects only half of the result-s bytes. In the example above comparing the correct and faulty outputs shows that the first 32 bytes are identical, while the last 32 bytes are different which makes it pretty easy to understand that a required error was injected during the infective recombination.

6.7 Conclusions

We made the following conclusions from our experiment. The first is that even with a multi-layer metal technology, front side laser attacks on complex SoCs are possible although we do believe that they are approaching their limit. The second conclusion is that even for very big and complex chips a black box approach to finding vulnerable locations works. By scanning the chip with the laser we found a potentially vulnerable area of approximately 80×270 μm. The spot where instruction skipping was possible is about 80×40 μm. The size of the chip is 4000×4000 μm, so a vulnerable spot constitutes only 0,135% of the SoC. We described only a front side experiment, but with a modification of the board, a back side attack was also conducted.

After the vulnerability is found, one can run a number of experiments to understand the effects of the injected errors on executed code. In our case it was proven that the laser shot's main effect was skipping an instruction: it seems that it was possible to corrupt values of the program registers.

The ability to skip an instruction provides an attacker with a very powerful tool: program flow can be altered giving an opportunity to execute functions which do not have to be executed at this time or skip functions that must be executed. This opens a possibility to attack countermeasures based on conditional checks. It also allows an attacker to bypass some of the infective countermeasures, such as those suggested in [KQ07a] modified (Ciet-Joye scheme), [KQ07b] (modified Giraud scheme), etc. In other words, all theoretical attacks described in [DGRS09] can be successfully carried out in practice.

The complexity of specifying a formal attack model prevents developing generic countermeasures with formal security proofs. Typically, countermeasures are dealt with on a case-by-case basis and apart from the attack model they depend on the algorithm and its implementation.

The experiment's most important outcome is that it proved in practice that multi-fault attacks on cryptographic algorithms running on a powerful 32-bit ARM Cortex M3 processor are feasible. Considering that many applications require device or user authentication, signature generation and public-key-based credentials exchange, this may have an important impact on security engineering practices. The significance of this result is that it forces the cryptographic community to re-consider what constitutes a feasible attack model and also to critically analyze existing countermeasures with respect to this new practical attack.

CHAPTER 7

DEFENSIVE LEAKAGE CAMOUFLAGE

Summary

Similar to the Shannon's model described in the Section 1.1 this chapter considers the transfer of digital data over *leaky and noisy* communication channels. The new defensive strategies proposed in this chapter are based on the fact that noise prevents the attacker from accurately measuring leakage.

The defence strategy described in this chapter pairs each useful data element k with a camouflage value v and simultaneously transmits both k and v over the channel. This releases an emission $e(k, v)$. The camouflage values $v(k)$ are selected as a function of k in a way that makes the quantities $e(k, v(k))$ as *indistinguishable* as possible from each other.

This chapter presents a model, which shows that optimal camouflage values can be computed from side-channels under very weak physical assumptions. The proposed technique is hence applicable to a wide range of readily available technologies.

We propose algorithms for computing optimal camouflage values when the number of samples per trace is moderate (typically ≤ 6) and justify our models by a statistical analysis.

We also provide experimental results obtained using FPGAs.

7.1 Introduction

A number of authors, *e.g.*, [BCF03], rely on the isotropic switching-model in which all bits dissipate identical switching energies. This thesis does not assume any *a priori* side-channel model and totally relies on the analysis of actually measured[1] emissions.

While most previous works analyzed leakage from complex cryptographic computations, we focus on one of the simplest forms of leakage: the emanations of a bus through which bits are being sent. We make only two physical assumptions:

- Emanations can be measured with equal (in)accuracy by both the attacker and the defender.
- Leakage is a global function of data plus noise. The proposed methods are thus unadapted to settings in which individual channel bits are probed with precision.

The proposed methodology is hence applicable to a wide range of circuits having *leaky* buses.

The proposed countermeasure pairs each useful data element k with a camouflage value v and simultaneously transmits k and v through the channel. This releases a physical side-channel emanation $e(k, v)$ that can be measured by both the attacker and the defender.

We address the following question:

> How can a defender pair each value of k with a corresponding value $v(k)$ that makes the $e(k, v(k))$ as *indistinguishable* as possible from each other?

The crux of this chapter is the definition of *indistinguishability* given the measured emissions.

Section 7.2 introduces algorithms for computing optimal camouflage values from actual power traces. These algorithms are efficient when each trace contains a few samples (typically ≤ 6). Section 7.3 presents a statistical analysis justifying the intuition that the best v values are those concentrating the $e(k, v)$ into the smallest possible sphere containing representatives of all k values. Section 7.4 provides experimental results.

In a way, this work achieves some sort of cryptographic key exchange based on the existence of ambient noise and on a gap in measurement accuracy between the legitimate receiver and the attacker.

7.2 Models and Algorithms

Let $e(d)$ represent the side-channel (*e.g.*, power consumption) resulting from the transfer of an n-bit data element d over an n-bit channel (*e.g.*, a bus). $e(d)$ can be measured with equal precision by both the attacker and the defender.

The defender builds a set of 2^n side-channel measurements $\mathcal{E}$. Each $e(d) \in \mathcal{E}$ is generated by transmitting an n-bit data element d. The defender assigns s channel bits to the useful information k, and devotes the remaining $n - s$ bits to the transmission of $(n - s)$-bit camouflage values $v(k)$. We denote $d = k|v$ and call the k's *"keys"* or *"colors"*. Note that key bits and camouflage bits are not necessarily adjacent and might be interleaved.

Let $e(k, v) = e(d)$ be the emanation released by transmitting $d = k|v$.

The vector $V = [v(0), \ldots, v(2^s - 1)]$ of all camouflage values must be chosen to make all emanations $e(k, v(k))$ look "as similar as possible". Our goal is to infer V from $\mathcal{E}$.

We assign a unique *color* $k = \text{color}(e(k|v))$ to each $e(i) \in \mathcal{E}$. $\mathcal{E}$ is hence analogous to a multidimensional cloud of 2^n colored points (*i.e.*, 2^s sets of colored points; each of these 2^s sets contains 2^{n-s} identically colored points).

A *color-spanning sphere* is a subset $\mathcal{B} \subset \mathcal{E}$ containing at least one emission of each color.

[1] potentially anisotropic

The defender will use the 2^n elements of $\mathcal{E}$ to select 2^s transmittable $k|v(k)$ values forming a color-spanning sphere $\mathcal{A}(V) \subset \mathcal{E}$. The attacker will only get to see traces belonging to $\mathcal{A}(V)$:

$$\mathcal{A}(V) = \bigcup_{k=1,\ldots,2^s-1} \{e \in \mathcal{E} : \text{color}(e) = k\}$$

The defender's goal is to minimize the *size* of the color-spanning sphere $\mathcal{A}(V)$ exposed to the attacker, *i.e.*, infer from $\mathcal{E}$ a *smallest color-spanning sphere* $\mathcal{A}_{\text{optimal}}$ such that

$$\|\mathcal{A}_{\text{optimal}}\| = \min_V \|\mathcal{A}(V)\|$$

$\mathcal{A}_{\text{optimal}}$ has thus the least size for all choices of V.

The next section considers the simplest setting where emanations are scalars[2]. In that case the difference $|e - e'|$ between two scalars $e, e' \in \mathcal{E}$ can be used as a similarity measure for constructing $\mathcal{A}_{\text{optimal}}$ efficiently.

7.2.1 One Dimension

Assume that the $e(d)$ are scalars (*e.g.*, execution times or a unique power measurement per trace). Acquire the 2^n reference traces:

$$\mathcal{E} = \{e(0), \ldots, e(2^n - 1)\}$$

A given choice of $V = [v(0), \ldots, v(2^s - 1)]$ restricts the attacker's information to

$$\mathcal{A}(V) = \{e(0, v(0)), \ldots, e(2^{s-1}, v(2^{s-1}))\}$$

The defender's goal is to minimize:

$$\|\mathcal{A}(V)\| = \max \mathcal{A}(V) - \min \mathcal{A}(V) = \max_k(e(k, v(k))) - \min_k(e(k, v(k)))$$

Let $\mathcal{P} = [p_0 \leq p_1 \leq \cdots \leq p_{2^n-1}]$ be the $e(i) \in \mathcal{E}$ sorted (with repetitions) by increasing scalar values. A *color-spanning segment* is an interval of $\mathcal{P}$ containing at least one p_i of each color.

A straightforward algorithm for finding $\mathcal{A}_{\text{optimal}}$ consists in working with two pointers start and end representing the beginning and the end of the segment under evaluation. When execution begins, start and end point at p_0. While [start,end] is not a color-spanning segment end is moved to the right. When end reaches p_{2^n-1} start is moved by one position to the right (*i.e.*, from p_i to p_{i+1}) and end is moved back to start. Throughout this process, whenever a shorter color-spanning segment is found, it is recorded. The complexity of this algorithm is quadratic in the cardinality of $\mathcal{E}$, *i.e.*, $O(2^{2n})$.

More clever approaches allow to solve the problem in $\tilde{O}(2^n)$. To do so build the 2^s sorted sequences (with repetitions) of emissions for each color:

$$\mathcal{P}^k = [p_0^k \leq \ldots \leq p_{2^{n-s}-1}^k] \text{ for } k = 1, \ldots, 2^s - 1$$

Represent the *color-spanning* segments by a binary search tree $\mathcal{T}$ of size 2^s.

At step 0, initialize the tree to $\mathcal{T}_0 = \{p_0^0, \ldots, p_0^{2^s-1}\}$ and proceed by 2^s-way merging.

At stage t, the color-spanning tree is

$$\mathcal{T}_t = \left\{p_{\lambda_t^0}^0, \ldots, p_{\lambda_t^{2^s-1}}^{2^s-1}\right\}$$

[2] *e.g.*, execution times or a unique power measurement per trace.

where the λ_t^k denote the merge pointers.

Let $\underline{m}$ and $\overline{m}$ denote (respectively) the minimal and maximal scalars in $\mathcal{T}_t$. We denote by ϕ_t the minimal (*i.e.*, best) segment length found at step t.

If $t = 0$ or $\overline{m} - \underline{m} < \phi_{t-1}$, then update $\phi_t = \overline{m} - \underline{m}$ else $\phi_t = \phi_{t-1}$.

Let $\underline{m} = p^c_{\lambda_t^c}$ and let $m = p^c_{\lambda_t^c+1}$ be the next emission of the same color. The next tree $\mathcal{T}_{t+1}$ is obtained by replacing $\underline{m}$ by m in $\mathcal{T}_t$, *i.e.*, we increase $\lambda^c_{t+1} = \lambda^c_t + 1$ and stall all other merge pointers $\lambda^k_{t+1} = \lambda^k_t$ for $k \neq c$.

The algorithm terminates (at some step $\tau < 2^n$) when it fails to find a successor m to $\underline{m}$. The length of the minimal color-spanning segment is then ϕ_τ.

Complexity:

Partitioning $\mathcal{E}$ to 2^s color subsets and sorting these subsets to get the $\mathcal{P}^k$ costs $O(n2^n)$.

Binary search trees [Knu98] support the operations (insert, find-min, extract-min and find-max) required by the structure $\mathcal{T}$, each of these operations requires $O(s)$ time. It follows that the 2^s-way merge runs in $O(s2^n)$ and hence the above algorithm has an overall complexity of $\tilde{O}(2^n)$.

7.2.2 Higher Dimensions

We now consider the general case where e is a T-dimensional vector, *e.g.*, a power consumption sampled at T different instants. $\mathcal{E}$ is now a T-dimensional cloud of colored points (Fig. 7.1) and the color spanning interval is a T-dimensional sphere. We need to determine the smallest sphere containing at least one point of each color, *i.e.*, the smallest color-spanning sphere $\mathcal{A}_{\text{optimal}}$ (Fig. 7.2, right).

The cloud of points is contained in some minimal enclosing T-dimensional rectangle $\mathcal{R}$, whose sides are parallel to the hyperspace's T axes (Fig. 7.3, right).

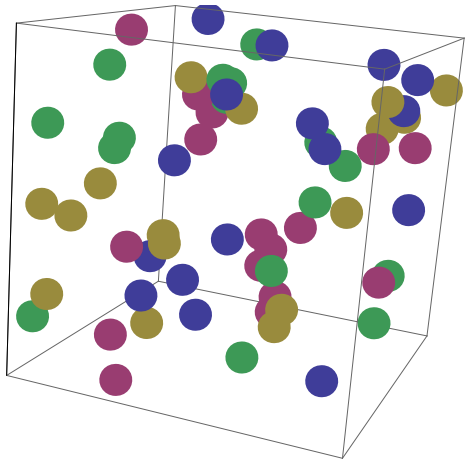


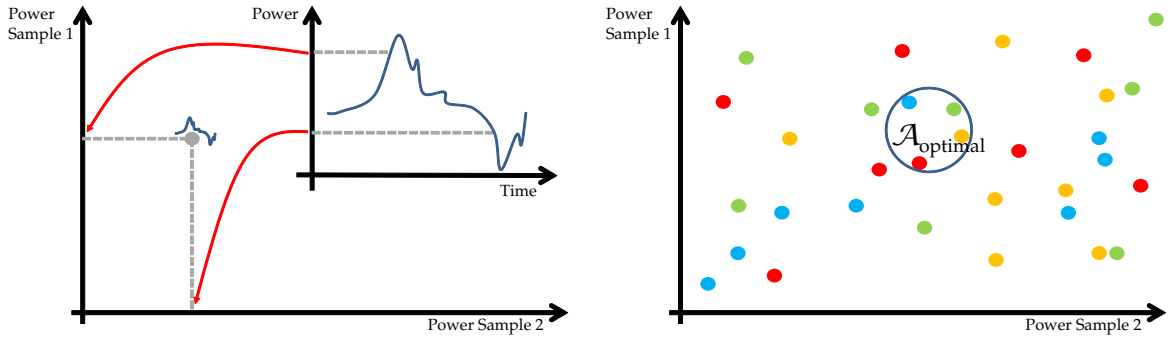
Figure 7.1 – 3D power trace representation.

Divide and Conquer

This problem lends itself to divide and conquer resolution.

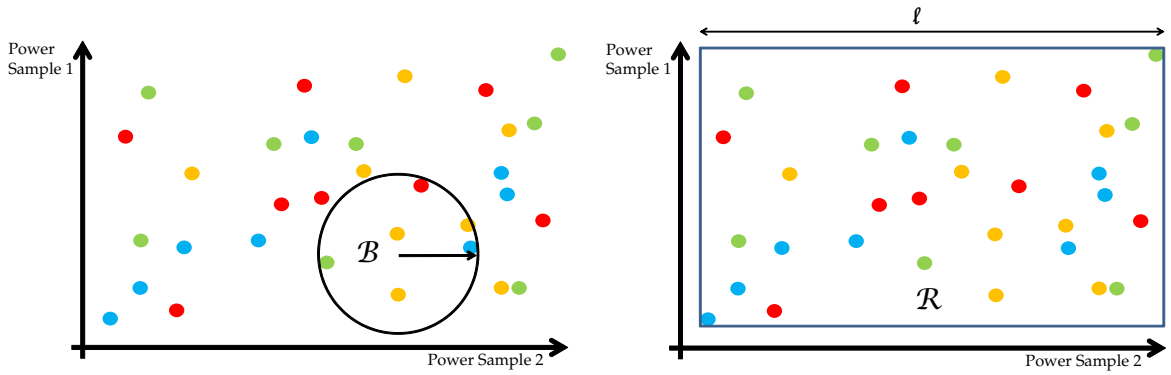
Let $\mathcal{B}$ be some[3] initial color spanning sphere of radius r. Let ℓ denote the length of the rectangle $\mathcal{R}$ along some dimension x. Split $\mathcal{R}$ along the x axis into two overlapping sub-rectangles of lengths $\frac{\ell}{2} + r$ as

[3] not necessarily optimal, *cf.* Fig. 7.3, left.



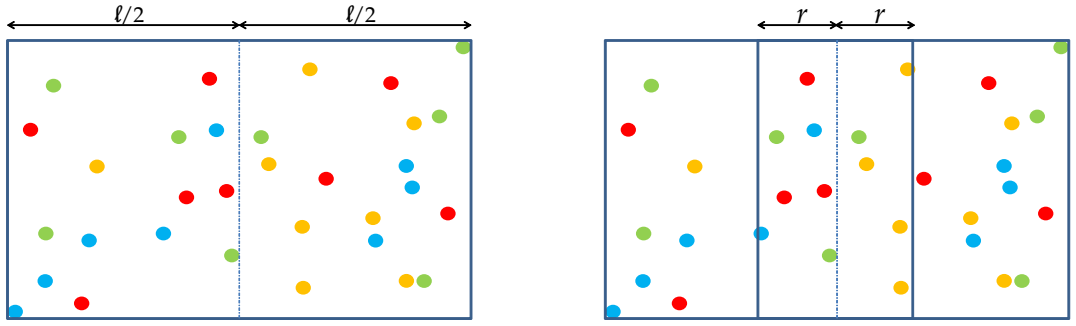
(a) Mapping curves into points (b) Problem instance and its optimal solution $\mathcal{A}_{\text{optimal}}$

Figure 7.2 – Determining the smallest sphere containing at least one point of each color.



(a) Step 1, find any color spanning sphere $\mathcal{B}$ (b) Step 2, define the rectangle $\mathcal{R}$

Figure 7.3 – First two steps of the 2D color-spanning algorithm.



(a) Rectangles of size $\frac{\ell}{2}$ (b) Rectangles of size $\frac{\ell}{2} + r$

Figure 7.4 – Step 3, split $\mathcal{R}$ into two overlapping rectangles $\mathcal{R}_{\text{right}}$ and R_{left} of length $\frac{\ell}{2} + r$.

shown by Figure 7.4. Let $\mathcal{R}_{\text{right}}$ and $\mathcal{R}_{\text{left}}$ be the two equally sized sub-rectangles obtained that way (Fig. 7.5).

By construction, $\mathcal{A}_{\text{optimal}}$ is fully contained in either $\mathcal{R}_{\text{right}}$ or $\mathcal{R}_{\text{left}}$. So, we recursively apply the process to $\mathcal{R}_{\text{right}}$ and $\mathcal{R}_{\text{left}}$ until splitting diminishes the rectangles' sizes only negligibly[4]. At that point we solve each of the smaller instances (by any chosen method) and output the smallest solution of all, which is

[4]After the w-th splitting the rectangles' sides are of size $\ell_w = (\ell - 2r)2^{-w} + 2r$. Hence splitting can last forever. We suggest to stop splitting when $\ell_w < 3r$, *i.e.*, after $\lfloor \log_2(\ell/r - 2) \rfloor$ iterations.

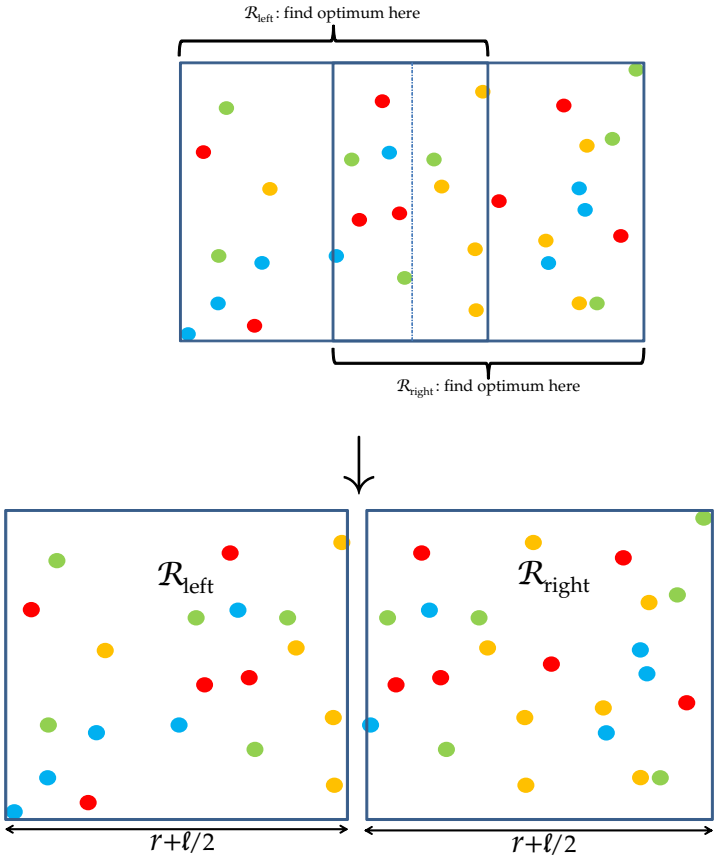


Figure 7.5 – Recursive problem size reduction.

indeed the smallest color-spanning sphere in $\mathcal{R}$, *i.e.*, the smallest color-spanning sphere $\mathcal{A}_{\text{optimal}}$ of the original problem.

Note that splitting can take place along several orthogonal axes simultaneously.

While practically very useful, this algorithm fails in a number of pathological cases (*e.g.*, when $\mathcal{B}$ is too large to split $\mathcal{R}$). Luckily this is a well-studied problem: [DBVKOS00] describes a simple linear-time algorithm in two dimensions and Welzl [Wel91] shows how to solve the problem in linear time for all dimensions, considering that the number of dimensions is a fixed problem parameter. Complexity is however exponential in the number of dimensions.

A key choice is the initial sphere $\mathcal{B}$: we want $\mathcal{B}$ to be small enough to significantly reduce the divide and conquer's search space. Yet, we want $\mathcal{B}$ to remain easy to compute.

Heuristics:

In our implementation we used the following method to construct $\mathcal{B}$: let p_0 be a point (for example the closest point to the center of $\mathcal{R}$) of color 0. After computing $p_1, \ldots, p_k$, we select as p_{k+1} the point of color $k+1$ at minimal distance from the barycenter of the cloud $p_1 \cdots p_k$. The resulting $\mathcal{B}$ is not necessarily optimal, (*cf.* Figure 7.7) but turns out to be much better than selecting any random color-spanning sphere.

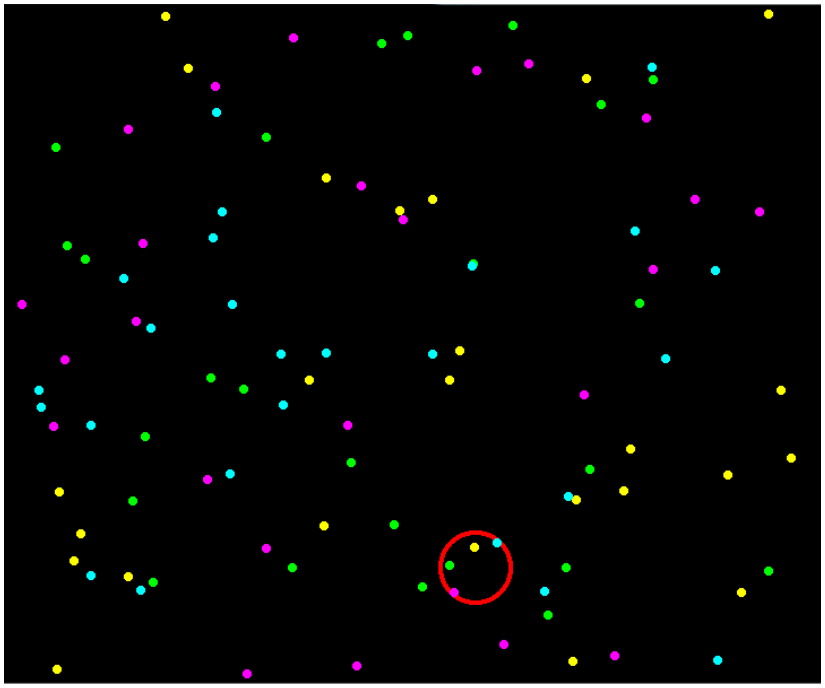


Figure 7.6 – Program output example in 2 dimensions.

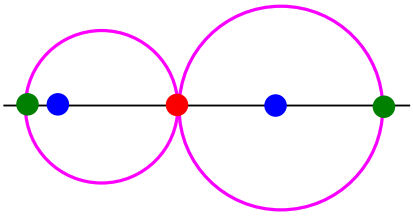


Figure 7.7 – The optimal sphere (left) is different from the sphere found by the barycenter heuristic (right) if the heuristic considers first the red, then the blue and finally the green points.

7.2.3 Implementations

Algorithms were implemented in C++[5] in a straightforward manner. A function

```
bool smallest_ball(points, space, output)
```

splits space and points as explained above (using a sphere found by `find_ball_barycenter`) and calls recursively `smallest_ball` on the smaller spaces, until this process stops to significantly decrease the problem size. We then use Miniball[6], a C++ software for computing smallest enclosing spheres of points in arbitrary dimensions (without requiring spheres to be color spanning) using brute force. The description of Miniball can be found in [Gar99, Wel91].

Timings were measured on a Dell Inspiron 1520[7]. Code was compiled using Visual C++ 2008 with all optimization flags set for maximal speed.

Experimental running times seem to confirm that the algorithm is linear in the number of points and exponential in the number of colors.

[5] the code is available at http://perso.ens-lyon.fr/quentin.fortier/color_ball.html
[6] http://www.inf.ethz.ch/personal/gaertner/miniball.html
[7] Intel Core 2 Duo T7300 processor, 2.0GHz, 4MB L2 cache, 2Go memory.

Total number of points	2 colors	3 colors	4 colors	5 colors
10^2	8 ms	11 ms	43 ms	211 ms
10^3	96 ms	221 ms	833 ms	7 s
10^4	946 ms	3 s	11 s	81 s
10^5	10 s	31 s	145 s	953 s
10^6	109 s	327 s		

Table 7.1 – Running time for points randomly chosen in the 3-dimensional unit cube, averaged over 10 runs.

Total number of points	2 colors	3 colors	4 colors	5 colors
10^2	11 ms	39 ms	309 ms	2 ms
10^3	164 ms	1 s	10 s	147 s
10^4	2 s	16 s	160 s	
10^5	27 s	188 s	37 min	
10^6	287 s	32 min	>1 hour	>1 hour

Table 7.2 – Running time for points randomly chosen in the 4-dimensional unit cube, averaged over 10 runs.

7.3 Why Euclidean Distances?

Let $\{m_{0,t}, \ldots, m_{n-1,t}\}$ be a database of n reference power consumption traces measured over some discrete time interval $t \in [0; T-1]$. Sample $m_{i,t}$ corresponds to the power consumption caused by the manipulation of data element i at instant t. Let μ_t be the average power consumption at time t and σ_t the standard deviation at time t:

$$\mu_t = \frac{1}{n}\sum_{i<n} m_{i,t} \qquad\qquad \sigma_t = \sqrt{\frac{1}{n}\sum_{i<n}(m_{i,t}-\mu_t)^2}.$$

Let a_t be an unidentified power measurement made by an attacker. The attacker's problem consists in finding the $m_{k,t}$ that *best reassembles* a_t. This section justifies why for doing so, an attacker would naturally compute for $i < n$ the quantities:

$$\text{score}(i) = \sum_{t<T} \frac{(a_t - m_{i,t})^2}{\sigma_t^2}, \tag{7.1}$$

and output the guess k corresponding to the $m_{k,t}$ whose score is the lowest *i.e.*:

$$\text{score}(k) = \min_{i<n}(\text{score}(i)).$$

This formula is justified in the next section for t-wise independent $m_{i,t}$'s.

In general, samples may be correlated, for instance when the same secret bit is manipulated at two different instants. We analyze this general case later and propose an explicit score minimization formula (7.2) taking into account intra-sample correlations.

7.3.1 Multivariate Normal Distributions

Equation (7.1) stems from the assumption that, for any fixed i, successive measurements of $m_{i,t}$ follow an independent normal distribution with mean μ_t and standard deviation σ_t, and hence abide by the probability density function:

$$f_{m_t}(x) = \frac{1}{\sigma_t\sqrt{2\pi}} \exp\Big(-\frac{(x-\mu_t)^2}{2\sigma_t^2}\Big)$$

When the $m_{i,t}$'s are independent, the probability density of all measurements $t < T$ can be expressed, for $\vec{x} = [x_0 \cdots x_{T-1}]$ as a T-dimensional multivariate distribution:

$$f_{\vec{m}}(\vec{x}) = \prod_{t<T} f_{m_t}(x_t) = \frac{1}{(2\pi)^{T/2} \prod\limits_{t<T} \sigma_t} \exp\Big(-\sum_{t<T} \frac{(x_t-\mu_t)^2}{2\sigma_t^2}\Big).$$

Note that in the previous equation μ_t and σ_t are the expected value and standard deviation of $m_{i,t}$ over *all* data elements i. For a measurement $m_{i,t}$ corresponding to a specific data element i, in addition, we also assume that $m_{i,t}$ follows a normal distribution with mean $\tilde{\mu}_t = m_{i,t}$ and standard deviation $\tilde{\sigma}_t$; we also assume that the standard deviation $\tilde{\sigma}_t$ around $m_{i,t}$ is the same for all data elements. In this case, the measurement m_t corresponding to data element i has the following distribution:

$$f_{\vec{m}}(\vec{x}) = \frac{1}{(2\pi)^{T/2} \prod\limits_{t<T} \tilde{\sigma}_t} \exp\Big(-\sum_{t<T} \frac{(x_t-m_{i,t})^2}{2\tilde{\sigma}_t^2}\Big)$$

Additionally, we assume that the standard deviation $\tilde{\sigma}_t$ of m_t around $m_{i,t}$ is proportional to the standard deviation σ_t of m_t when all data values are considered, *i.e.*, we assume $\tilde{\sigma}_t = \alpha \cdot \sigma_t$ for all $0 \le t \le T-1$ for some $\alpha \in \mathbb{R}$. In this case, the probability density function of the m_t's for data i can be written as:

$$f_i(\vec{m}) = \frac{1}{(2\pi)^{T/2}\alpha^T \prod\limits_{t<T} \sigma_t} \exp\Big(-\sum_{t<T} \frac{(m_t-m_{i,t})^2}{2\alpha^2\sigma_t^2}\Big) \propto \exp\Big(-\frac{\text{score}(i)}{2\alpha^2}\Big)$$

where score(i) is given by equation (7.1). The probability to obtain measurements m_t from data i is thus a decreasing function of score(i). Given measurement $\vec{m}$, the most probable candidate is therefore the one with the lowest score.

7.3.2 Multivariate Normal Distribution: Taking Correlation into Account

We denote by Σ the covariance matrix of the measurements, defined as follows:

$$\Sigma = \text{var}(\vec{m}) = \text{var}\begin{pmatrix} m_1 \\ \vdots \\ m_T \end{pmatrix} = \begin{pmatrix} \text{var}(m_1) & \text{cov}(m_1m_2) & \cdots & \text{cov}(m_1m_T) \\ \text{cov}(m_1m_2) & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(m_1m_T) & \cdots & \cdots & \text{var}(m_T) \end{pmatrix}$$

where $\text{cov}(X,Y) = \text{E}(XY) - \text{E}(X)\text{E}(Y)$ and $\text{var}(X) = \text{cov}(X,X) = \text{E}(X^2) - \text{E}(X)^2$.

We assume that the measurements follow a T-dimensional multivariate distribution with mean $\vec{\mu}$ and covariance matrix Σ. The probability density function can then be expressed as:

$$f_{\vec{m}}(\vec{x}) = \frac{1}{(2\pi)^{T/2}|\Sigma|^{1/2}} \exp\Big(-\tfrac{1}{2}(\vec{x}-\vec{\mu})^{\text{tr}}\Sigma^{-1}(\vec{x}-\vec{\mu})\Big).$$

where $|\Sigma|$ is the determinant of Σ and M^{tr} is the transposed of matrix M. The mean $\vec{\mu}$ is a T-vector and Σ is a $T \times T$-matrix.

Note that in the previous equation $\vec{\mu}$ and Σ are the expected value and covariance matrix of measurements for *all* data elements i. As previously for measurements corresponding to a specific data element i, we assume that these measurements follow a T-multivariate normal distribution with mean $\tilde{\mu}_t = m_{i,t}$ and covariance matrix $\tilde{\Sigma}$.

If we further assume that matrix $\tilde{\Sigma}$ is identical for all data elements, the measurement $\vec{m}$ for data i then obeys the multivariate distribution:

$$f_{\vec{m}}(\vec{x}) = \frac{1}{(2\pi)^{T/2}|\tilde{\Sigma}|^{1/2}} \exp\Big(-\tfrac{1}{2}(\vec{x} - \vec{m_{i,.}})^{\mathrm{tr}}\tilde{\Sigma}^{-1}(\vec{x} - \vec{m_{i,.}})\Big).$$

As previously, let us additionally assume that the covariance matrix satisfies $\tilde{\Sigma} = \alpha \cdot \Sigma$ for some $\alpha \in \mathbb{R}$. In this case, the probability density function is expressed by:

$$f_{\vec{m}}(\vec{x}) = \frac{1}{(2\pi\alpha)^{T/2}|\Sigma|^{1/2}} \exp\Big(-\tfrac{1}{2\alpha}(\vec{x} - \vec{m_{i,.}})^{\mathrm{tr}}\Sigma^{-1}(\vec{x} - \vec{m_{i,.}})\Big).$$

This can finally be written as

$$f_{\vec{m}}(x) = \frac{1}{(2\pi\alpha)^{T/2}|\Sigma|^{1/2}} \exp\Big(-\frac{\mathrm{score}(i)}{2\alpha}\Big)$$

where

$$\mathrm{score}(i) = (\vec{m} - \vec{m_{i,.}})^{\mathrm{tr}}\Sigma^{-1}(\vec{m} - \vec{m_{i,.}}) \tag{7.2}$$

It follows that equation (7.2) is a generalization of equation (7.1) where correlations are taken into account. In other words, to take correlations into account acquire a_t and compute for every i the score as per equation (7.2), sort the scores by increasing values and bet on the smallest.

Examples

To illustrate the procedure, we consider the bivariate case where the covariance matrix between variables X and Y is:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$$

where $\mathrm{var}(X) = \sigma_x^2$, $\mathrm{var}(Y) = \sigma_y^2$, $\mathrm{cov}(X,Y) = \rho\sigma_x\sigma_y$ and ρ is the correlation between X and Y. In this case, we find:

$$\Sigma^{-1} = \frac{1}{1-\rho^2}\begin{bmatrix} \frac{1}{\sigma_x^2} & \frac{-\rho}{\sigma_x\sigma_y} \\ \frac{-\rho}{\sigma_x\sigma_y} & \frac{1}{\sigma_y^2} \end{bmatrix}$$

and the probability density function can be written as

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right]\right).$$

In this case, equation (7.2) gets simplified as follows:

$$s_i = \frac{(a_1 - m_{i,1})^2}{\sigma_1^2} + \frac{(a_2 - m_{i,2})^2}{\sigma_2^2} - \frac{2\rho(a_1 - m_{i,1})(a_2 - m_{i,2})}{\sigma_1\sigma_2}$$

where $\sigma_1 = \mathrm{var}(m_1)$, $\sigma_2 = \mathrm{var}(m_2)$ and ρ is the correlation between m_1 and m_2.

7.4 Experiments

7.4.1 Measurements

This section describes our experimental results using the Altera EP2C20F484C7N FPGA present on the Cyclone II Starter Development Kit (SDK). Fig.7.8 shows the circuit used to measure the power consumption of a memory read + register store operation. The circuit consisted of a RAM, a multiplexer, eight registers, slide switches (`DIP`) and buttons. Identical data was simultaneously written into eight identical registers to increase power signature.

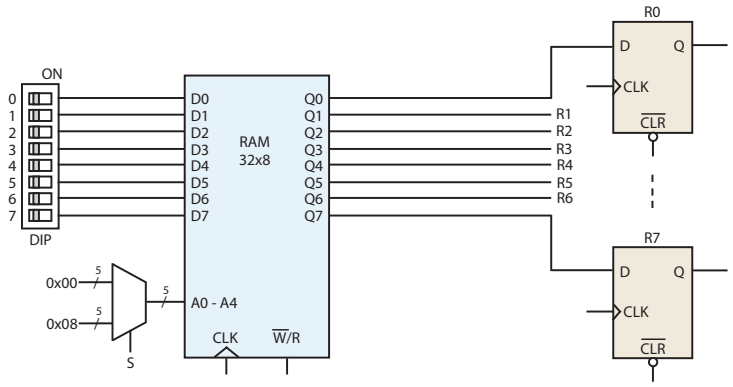


Figure 7.8 – The experimental circuit used for power consumption measurements.

Power was measured using a 1GHz oscilloscope (TDS 684B) and a Tektronix P6247 differential probe (1GHz bandwidth). The SDK's two GPIO pins (`power` and `ground`) were connected via the differential probe. Apart from DC signal rejection no filtering or power trace post processing was done.

The experimental protocol was defined as follows:

- The `DIP`'s eight slide switches were manually set to `0x00`.
- Address `0x00` was latched on address bus `A=[A0,...,A4]` using the multiplexer's control bit `S`. This caused the value `0x00` to be written into RAM address `0x00`.
- For $d = 0$ to 255:
  - The `DIP`'s eight slide switches were manually set to d.
  - Pressing the board's `KEY0` button triggered the following sequence of events 1000 times (averaged to remove noise):
    1. RAM write ($\overline{\text{W}}$) was activated and bit `S` was used to latch address `0x08` on bus `A`. This caused d to be written to RAM address `0x08` (1 cycle).
    2. RAM read was activated (`R`) and bit `S` was used to latch address `0x00` on bus `A`. This caused `0x00` to be read-out of RAM and clear all data previously present on the bus and in the registers (3 cycles).
    3. The RAM's `CLK` signal was disabled.
    4. Bit `S` was used to latch address `0x08` on bus `A`.
    5. The oscilloscope was triggered.
    6. The RAM's `CLK` signal was enabled for one cycle only causing d to appear on bus `[R0,...,R7]`. The RAM's `CLK` signal was immediately re-disabled to avoid a double-reads and freeze d on bus `[R0,...,R7]`.
    7. At the next clock cycle, d appeared at the `Q` output pins of the eight registers.
    8. The clock was left running for one more cycle to acquire any signal tails due to capacitive discharges.
  - A 2500-sample averaged power measurement $e'(d)$ was recorded.
  - Three samples corresponding to instants t_0, t_1, t_2 were extracted from $e'(d)$ to form $e(d)$. $e(d)$ was recorded[8] as a file `trace_d.d` used for camouflage calculations.

The described state-flow could only be interrupted by power-off or by pressing `KEY0`. A finite state-machine (FSM) diagram will appear in the final version of this paper. A characteristic power trace is shown in Figure 7.9.

[8] 3 big-endian values stored in ASCII in decimal format. Each sample is represented by two bytes (oscilloscope's precision).

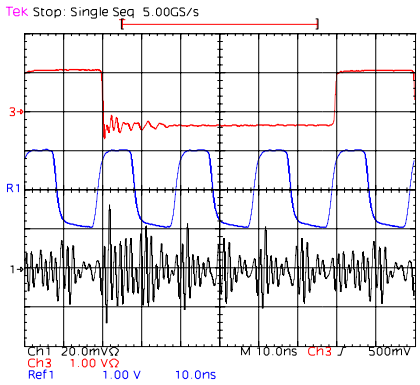


Figure 7.9 – Power trace of the circuit of Fig.7.8.

The obtained results confirm very wall both our analysis and intuition. However, for various technical reasons, we are not entirely satisfied with this first measurement campaign. We thus plan to refine our setting and provide new experimental results in the final paper.

7.4.2 Analysis

Figure 7.10 represents the 256 values ($n = 8$) obtained experimentally as 8 color families (*i.e.*, 32 points per family). The experimental data is available upon request.

Our goal is to consider this data as 2^i colors $\times$ 2^{8-i} points for $i = 1, \ldots, 7$, select the optimal bus bits on which k should be encoded, compute the $v(k)$ in all cases and check if the results indicate, as we conjecture, that similar Hamming weight words yield the best encoding.

For two colors (*i.e.*, a 1-bit k) the two most similar bus values are `0x7F` and `0xF9` for which:

$$\begin{aligned} \text{distance}(e(\texttt{0x7F}), e(\texttt{0xF9})) &= \\ &= \text{distance}(\{28601, 28795, 28794\}, \{29115, 28789, 28876\}) \\ &= 26.94 \end{aligned}$$

For four colors (*i.e.*, a 2-bit k) we get:

binary value of k	**optimal** k **and** $v(k)$	**observed side channel** $e(k, v(k))$
`00`	`0xB4=10110100`	$e(\texttt{0xB4}) = \{28704, 28232, 28278\}$
`01`	`0xD9=11011001`	$e(\texttt{0xD9}) = \{28652, 28107, 28315\}$
`10`	`0x96=10010110`	$e(\texttt{0x96}) = \{28716, 28159, 28293\}$
`11`	`0x6B=01101011`	$e(\texttt{0x6B}) = \{28670, 28280, 28380\}$

$e(\texttt{0xB4}), e(\texttt{0xD9}), e(\texttt{0x96}), e(\texttt{0x6B})$ are contained in a sphere of radius $\sqrt{\frac{17239}{2}} \cong 92.84$ centered at $c = \{28661, 28193.5, 28347.5\}$ where:

$$\begin{aligned} \text{distance}(c, e(\texttt{0xB4})) &= 90.34 & \text{distance}(c, e(\texttt{0xD9})) &= 92.84 \\ \text{distance}(c, e(\texttt{0x96})) &= 84.77 & \text{distance}(c, e(\texttt{0x6B})) &= 92.84 \end{aligned}$$

The positions are illustrated in Figure 7.11 where points were re-scaled to $[0, 1]$ using the affine transform $\text{rescale}\{x, y, z\} = \{u(x), u(y), u(z)\}$ where $h(u) = (u - 28107)/609$:

$$\begin{aligned} \text{rescale}(e(\texttt{0xB4})) &= \{0.98, 0.21, 0.28\} & \text{rescale}(e(\texttt{0xD9})) &= \{0.89, 0.00, 0.34\} \\ \text{rescale}(e(\texttt{0x96})) &= \{1.00, 0.09, 0.31\} & \text{rescale}(e(\texttt{0x6B})) &= \{0.92, 0.28, 0.45\} \end{aligned}$$

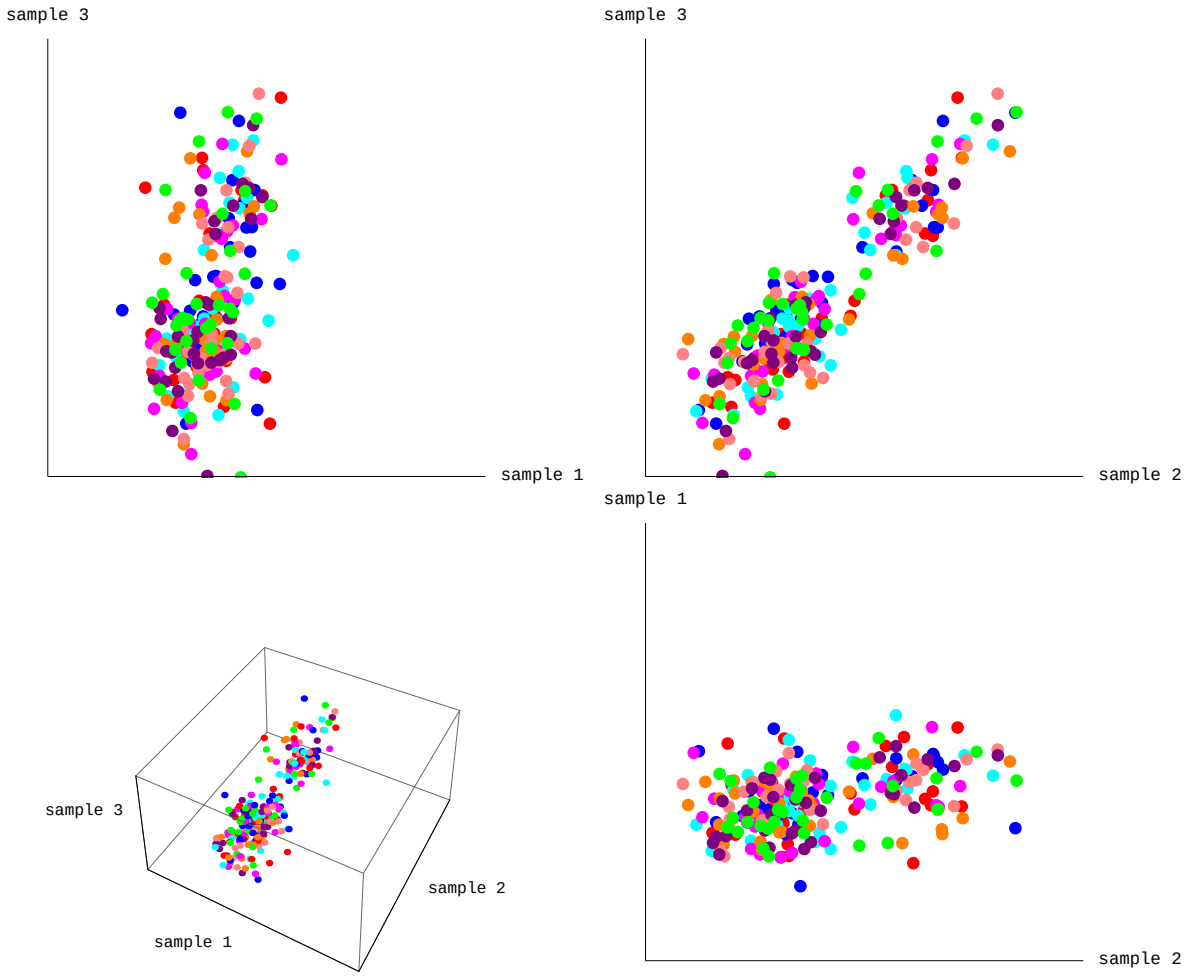


Figure 7.10 – Experimental results for $n = 8$. 3D and projected representations of the 256 experimental measurements (represented as 8 color families of 32 points).

7.5 Conclusions and Further Research

This works raises a number of interesting questions. A first natural generalization is the translation of our analysis to an infinite number of dimensions (in terms of metrics on function spaces and distances between functions).

A second line of research consists in introducing more complex information encoding schemes. Here the defender detects the 2^s most similar traces in $\mathcal{E} = \{e(0), \ldots, e(2^n - 1)\}$, *e.g.*, using clustering. Let $\mathcal{L}$ be the subset (cluster) of these most similar traces:

$$\mathcal{L} = \{e(\ell(1)), \ldots, e(\ell(2^s - 1))\} \subset \mathcal{E}$$

The communicating parties assign[9] to the transmitted information the encoding:

$$\ell(k) = \text{encode}(k) \qquad k = \text{decode}(\ell(k))$$

Along the same line of ideas, a further refinement consists in buying an easier computation of camouflage values at the cost of extra assumptions on the power consumption model. Assume for instance an isotropic consumption model where emanations are proportional to the Hamming weight of the transmitted data. Here all $\binom{n}{w}$ emissions of weight w cause identical emanations. The largest binomial has weight $w = n/2$, and it is bounded by

[9] *e.g.*, using a lookup table.

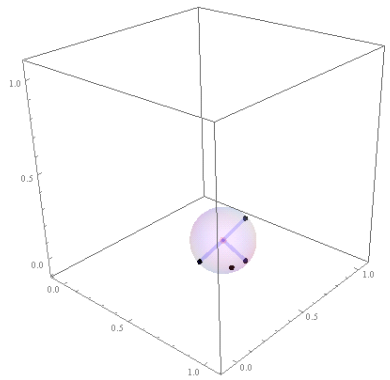


Figure 7.11 – Display of the rescaled solution.

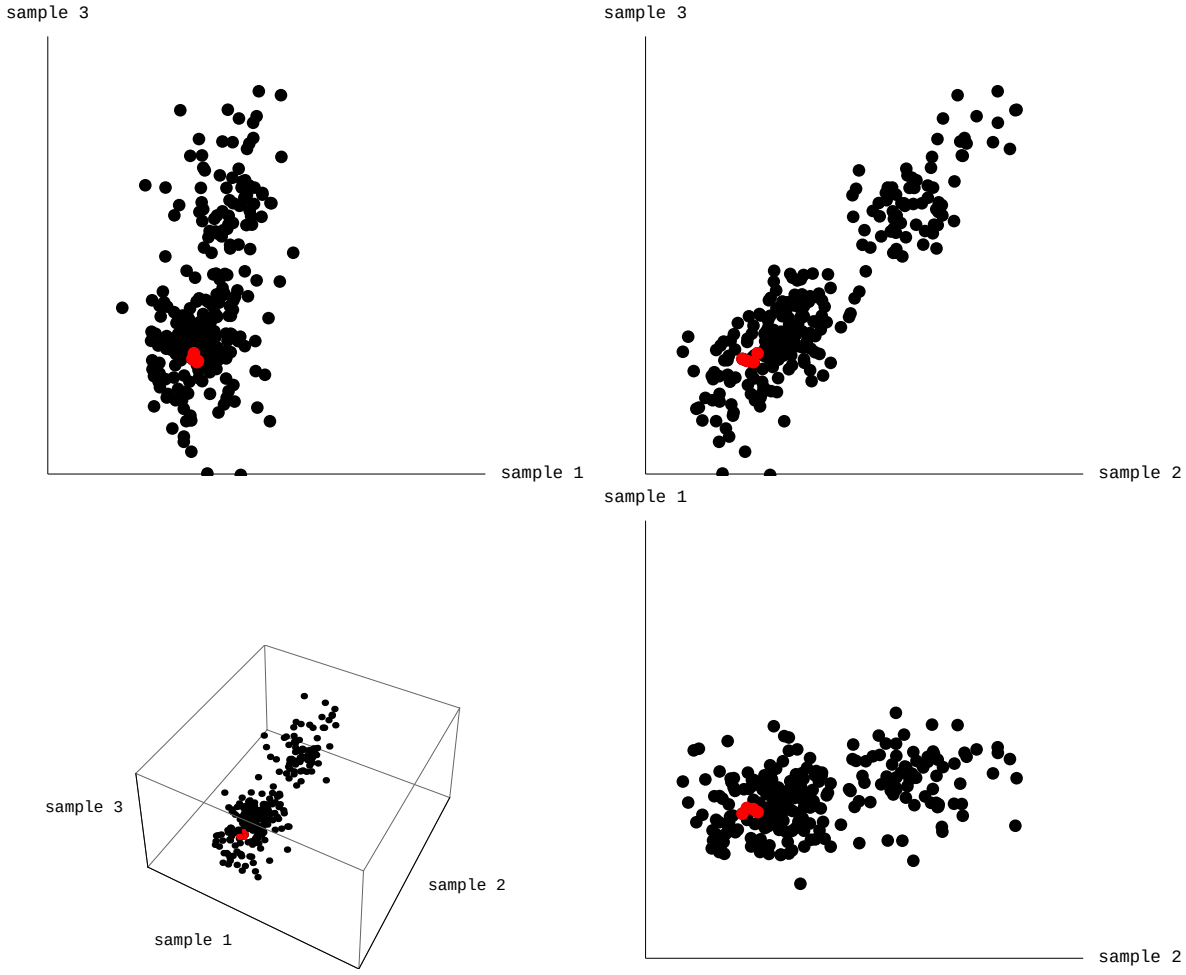


Figure 7.12 – Experimental results for $n = 8$. Position of the optimal solutions.

$$\frac{2^n}{\sqrt{2n}} < b_n = \binom{n}{\frac{n}{2}} < \frac{2^n}{\sqrt{\pi n/2}}.$$

Assigning $c_n = \lceil \log_2 \sqrt{2n} \rceil$ implies that $2^{n-c_n} \leq 2^n/\sqrt{2n} < b_n$, *i.e.*, $2^s < b_n$ for $s = n - c_n$. We can thus choose a distinct configuration of weight $n/2$ to encode each secret key k. It follows that $c_n = (3 + \log_2 n)/2$ bits are sufficient to perfectly hide the emanations from $s = n - c_n$ keys over the n bits of an isotropic bus.

If the noise level is high enough then the implementer may use the fact that

$$\log\left(\left(\binom{n}{\frac{n}{2}}\right)\right) \simeq \log\left(\left(\binom{n}{\frac{n}{2} \pm \gamma}\right)\right) \quad \text{for moderate } \gamma \text{ values}$$

and increase bandwidth at the cost of a carefully controlled security risk.

CHAPTER 8

BUYING AES DESIGN RESISTANCE WITH SPEED AND ENERGY

Summary

Despite the fact that AES is mathematically safer than the DES, straightforward AES implementations are not necessarily secure and several authors [Koc96, KJJ99, MOP07] have exhibited ways of exploring information that leaks from AES implementations. Such leakage is typically power consumption, electromagnetic emanations or the time required to process data. Additional constraints such as fault-resistance, chip technology, performance, area, power consumption, and even patent compliance further complicate the design of real-life AES coprocessors.

This chapter addresses resistance against two physical threats: power and fault attacks. The proposed AES architecture leverages the algorithm's structure to create low-cost protections against these attacks. The proposed design allows very flexible runtime configurability without significantly affecting performance.

This chapter is organized as follows: Section 8.1 proposes an architecture for implementing AES. Section 8.2 explains how to add power scrambling and fault detection to the proposed implementation. The result is a chip design allowing 29 different software-controlled runtime configurations. Section 8.3 compares simulation and synthesis results between an unprotected AES and our protected implementations.

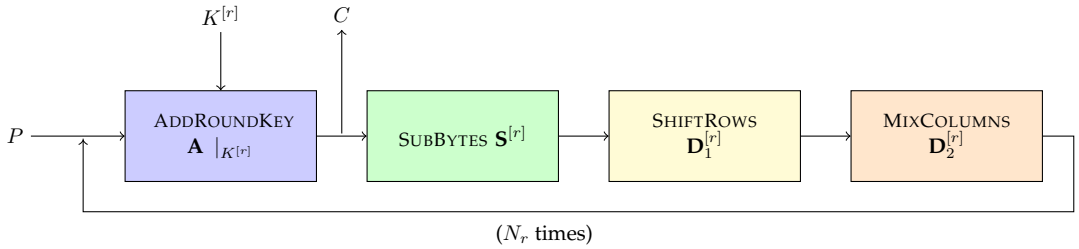


Figure 8.1 – AES encryption flowchart.

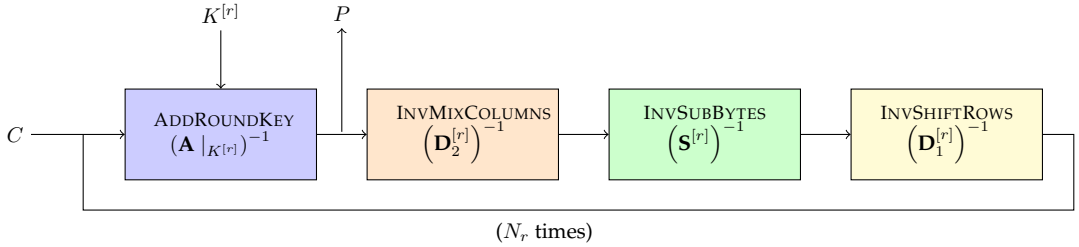


Figure 8.2 – AES decryption flowchart.

8.1 The Proposed AES Design

AES algorithm structure is explained in Section 1.3.1. The proposed design places a register barrier after each transformation (and their inverse), namely, AddRoundKey, SubBytes, ShiftRows, and MixColumns. The register barrier is used to save intermediate results. Therefore the intermediate information that eventually yields $S^{[r]}$ is saved four times during each AES round. It takes $4N_r + 1$ clock cycles to encrypt (or decrypt) a data block using this design.

Fig. 8.1 and Fig. 8.2 show that, during each clock cycle, only one block of the chain actually computes the state, while the other three blocks are processing useless data. This is potentially risky, as the three concerned blocks "chew" computationally useless data related to P (or C) and $K^{[r]}$ and thereby expose the design to unnecessary side-channel attacks.[1] This computation is shown in Fig. 8.3 where red arrows represent the path of usefully active combinatorial logic.

8.2 Energy and Security

8.2.1 Power Analysis

SCA and FA are explained in Section 2 and Section 3 respectively. To benchmark our design the AES was implemented on FPGA. Power was measured at 1GS/s sampling rate with 250MHz bandwidth using PicoScope 3407A oscilloscope. To guarantee the identical conditions every new plaintext was given to the FPGA at the same clock after the reset.

We performed a Correlation Power Attack (CPA) on the first AES S-box output since S-box operation is generally considered as the most power gluttonous. Our power model was based on the number of flipped register's bits in the S-box module when the initial register's barrier R_0 is rewritten with the S-box output as follows:

$$\mathrm{HD}\left(\mathbf{S}\left(P \oplus K_0\right), R_0\right) = \mathrm{HW}\left(\mathbf{S}\left(P \oplus K_0\right) \oplus R_0\right) \tag{8.1}$$

where R_0 is the previous register's state; P is a given plaintext; K_0 is the AES master key.

The value R_0 was assumed to be constant since all the encryptions were performed at the same clock after the reset. When R_0 could not be computed then all possible 256 values were tried. Pearson correlation

[1] In that respect see our open question in Section 8.5.

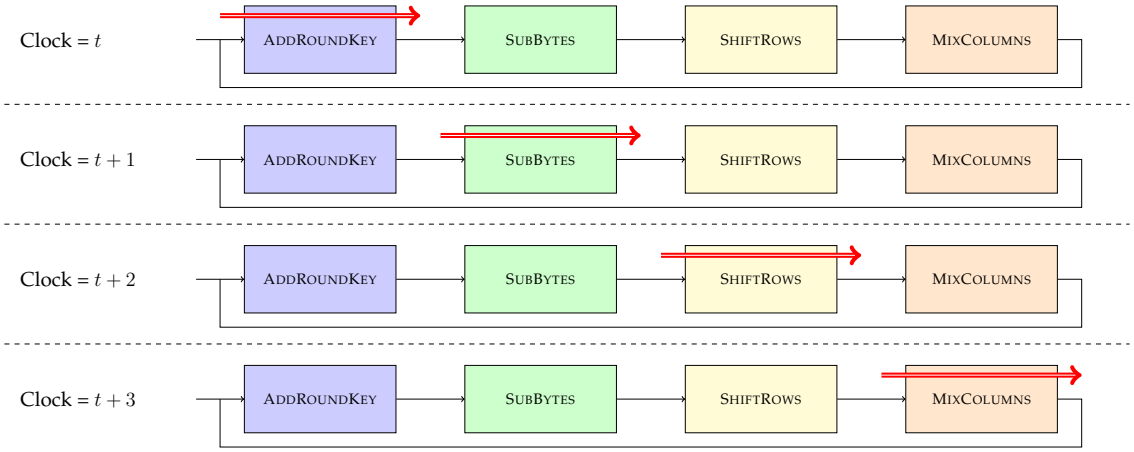


Figure 8.3 – Flow of computation in time.

coefficient was used to link the model and the genuine consumed power.

The following section presents a reference evaluation of the unprotected AES implementation showing its vulnerability compared to two (LFSR and tri-state buffers) side-channel countermeasures introduced later.

8.2.2 Power Scrambling

It is a natural idea to shut down unnecessarily active blocks. To do so, each block receives a new 1-bit input named *ready* activating the block when $ready = 1$. If $ready = 0$, the block's pull-up resistors are disconnected using a tri-state buffer connected to the power source. This saves power and also prevents the circuit from leaking "unnecessary" side-channel information.

Logically the pipeline architecture that we have just described has to be less vulnerable against First Order DPA attacks. Its four register barriers introduce additional noise, so we expect that the correlation shall be at least smaller that for the AES design with one round per clock computation.

To asses the security of each proposed design, we will compare an incorrect key byte correlation to a correct key byte correlation. Fig. 8.4 shows these two coefficients. As expected, the correct key is correlated to the power traces, however even for 500,000 traces Pearson correlation coefficient is smaller than 0.015. Anyway, this implementation is vulnerable.

To exploit the unused blocks to hide the device's power signature even better we propose two modifications. The first consists in injecting (pseudo) random data into the unused blocks, making them process that random data. Subsequently, three of the four blocks will consume power in an unpredictable manner. Note that because we use the exact same gates to compute and to generate noise, the expected spectral and amplitude characteristics of the generated noise should mask leakage quite well. Although any random generator may be used as a noise source, we performed our experiments using a 128-bit LFSR. An LFSR is purely coded in digital HDL, making tests easier to implement.

Fig. 8.5 shows that a multiplexer controlled by the *ready* signal selects either the useful intermediate state information or the pseudo-random LFSR output. For the AddRoundKey block, LFSR data replaces the key. Therefore when AddRoundKey's $ready = 0$, pseudo-random data (unrelated to the key) are xored with the state coming from the previous block (MixColumns if encrypting, InvShiftRows if decrypting). For the other blocks, the pseudo-random data replaces the state when $ready = 0$.

Attacks performed on this implementation revealed that this countermeasure increases key lifetime. Fig. 8.6 is the equivalent of Fig. 8.4 for the protected implementation using an LFSR. The correct key correlation can not be distinguished from the incorrect key correlation even with 1,200,000 traces. However, we assume that this implementation still might be vulnerable if more traces are acquired or if Second Order DPA is applied.

Real-life implementations must use true random generators. Indeed, if a deterministic PRNG seed

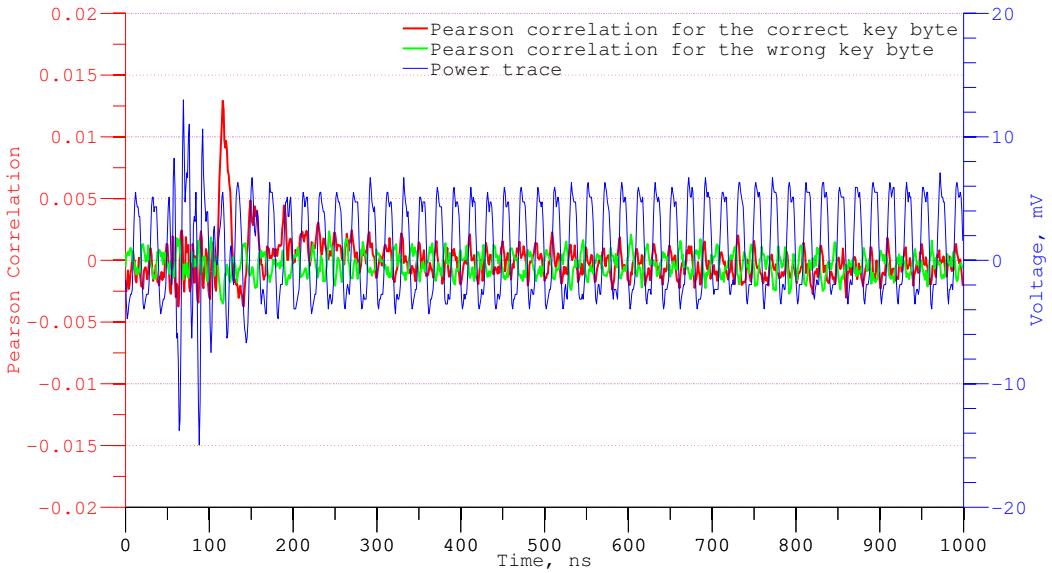


Figure 8.4 – Unprotected implementation: Pearson correlation value of a correct (red) and an incorrect (green) key byte guess. 500,000 power traces.

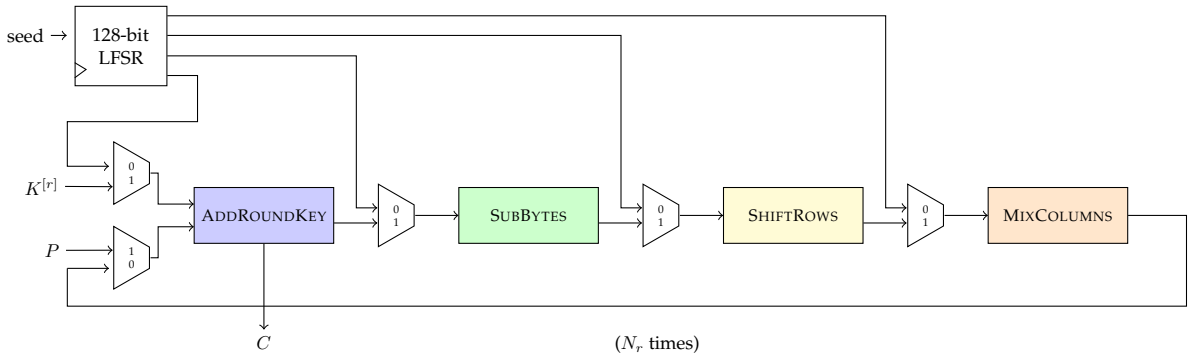


Figure 8.5 – Power scrambling with a PRNG.

is used the noise component in all encryptions becomes constant and cancels-out when computing differential power curves.

A second design option interleaves tri-state buffers between blocks to hide power consumption. By shutting down the three useless blocks, we create a scrambled power trace where one block computes meaningful data while the other three "process" high impedance inputs, which means that these blocks "compute" leakage current coming from their inputs.

As illustrated in Fig. 8.7, the input signal $ready_i$ determines which blocks are tri-stated and which block is computing the AES state. In other words, the $ready_i$ signal "jumps" from one block to the next, so that only one block is computing while the other three are scrambling the power consumption. Although this solution has a smaller overhead in terms of area (as it does not require random number generation) tri-state buffers tend to be slow. Furthermore, the target environment (FPGA or IC digital library) must offer tri-state cells.

The experimental results we obtained on FPGA were surprising, we couldn't attack the design with 800,000 power traces. The correlations shown in Fig. 8.8 do not allow to visually distinguish the correct key from a wrong guess. As before we assume that this implementation can be still attackable if more power traces are acquired or if Second Order DPA is applied.

A full study of this solution would require an ASIC implementation with real tri-state buffers, as an FPGA emulates these buffers and may turn out to be resistant because of an undesired CLB mapping side effects.

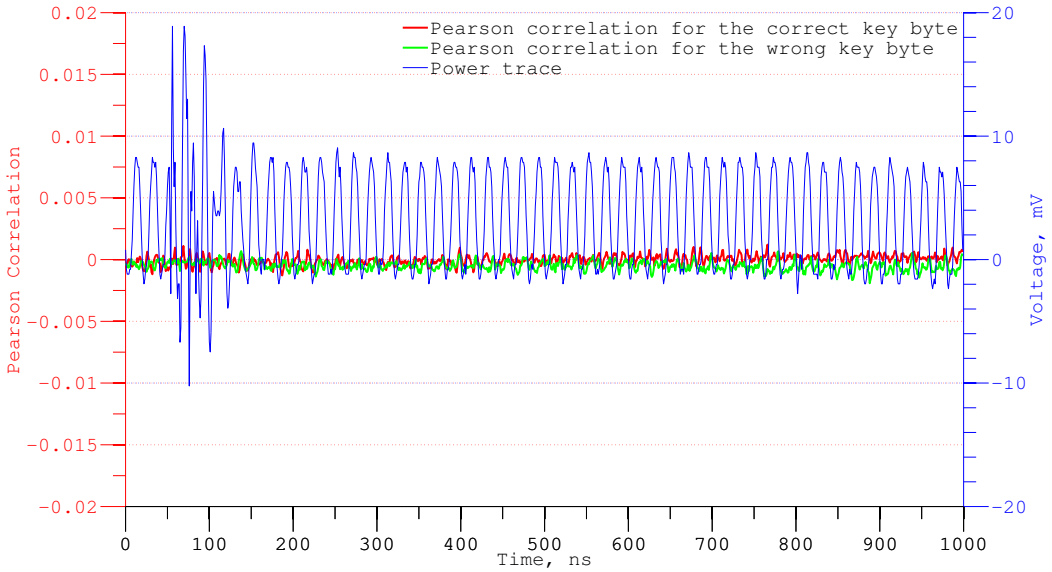


Figure 8.6 – LFSR implementation: Pearson correlation value of a correct (red) and an incorrect (green) key byte guess. 1,200,000 power traces.

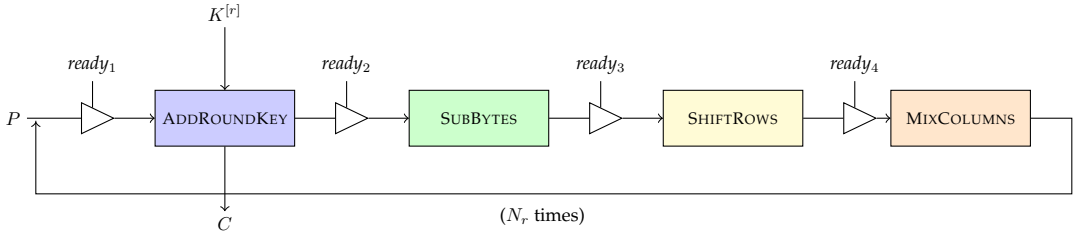


Figure 8.7 – Power scrambling with tri-state buffers.

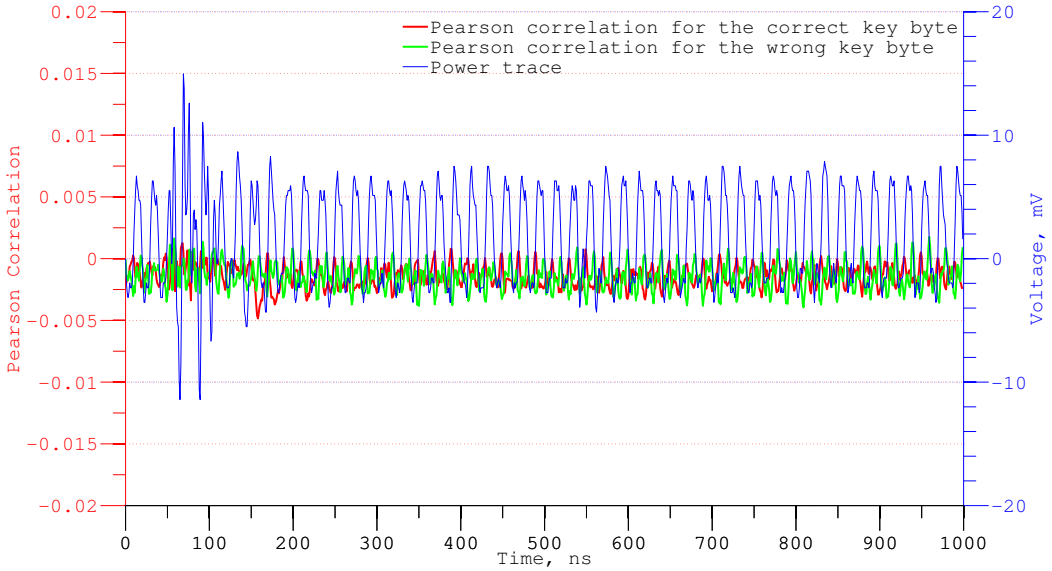


Figure 8.8 – Tri-state buffers implementation: Pearson correlation value of the correct key byte (green) and a wrong key byte guess (red). 800,000 power traces.

8.2.3 Transient Fault Detection

We will now use idle blocks to check for transient faults. Each block in the chain can "stutter" during two consecutive clock cycles to recompute and check its own calculation. For instance, as shown in Fig. 8.9, at clock t, a given block B_i receives a $ready_i$ signal, computes the state and saves it in the register barrier R_i. At clock $t+1$, the result enters the next block $B_{i+1 \bmod 4}$ which is now working, while B_i reverts to checking, *i.e.*, B_i recomputes the same output as at clock t and compares it to the saved B_i value. This process is repeated for the other blocks in the chain. If any transient fault happens to cause a wrong result at the output of any block, the error will be detected within one clock cycle.

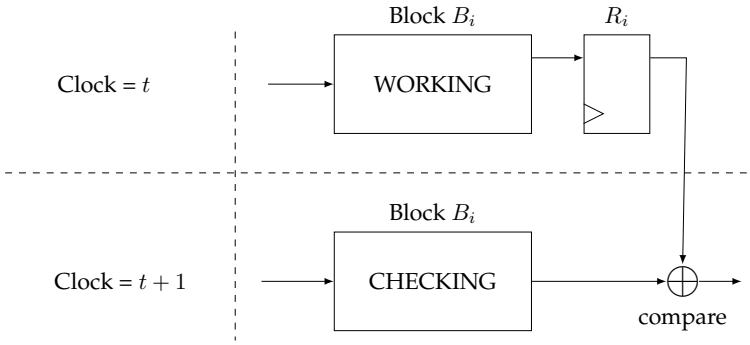


Figure 8.9 – Transient fault detection scheme for AES.

8.2.4 Permanent Fault Detection

The AES structure of Section 1.3.1 also allows us to use one block of the chain to compute a pre-determined plaintext or ciphertext. The encryption (or decryption) of a chosen input (*e.g.*, the all-zero input Z) is pre-computed once for all and hardwired (let $W = \text{AES}(Z)$ denote this value). While the system processes the actual input through one block (out of four) during any given clock cycle, another block is dedicated to recompute W. One clock after the actual C emerges, AES(Z) can be compared to the hardwired reference value W. If $W \neq \text{AES}(Z)$, a transient or a permanent fault occurred.

In this scenario, the system starts by computing AES(Z) in the first clock cycle, followed by the actual computation of C. This allows the implementation to check up all the blocks during the execution and make sure that no permanent fault occurred. In the last clock cycle, while C is being processed in the last block, the correctness of AES(Z) is compared with the hardwired value before outputting C.

In Fig. 8.10, the red arrows represent data flow through the transformation blocks. After the initial clock cycle, the first block starts computing C. The WORKING blocks represent the calculation of C. The CHECKING blocks represent the calculation of AES(Z).

While AES(Z) will be calculated in $4N_r + 1$ clock cycles, C will be calculated in $4N_r + 2$ cycles. If the fault needs to be caught earlier, the solution described in [BBKP02] can be adapted. Yet another option consists in comparing intermediate Z encryption results (*i.e.* intermediate state values) to hardwired ones. Note that our design differs from [BBKP02] where a the decryption block is used for checking the encryption's correctness [BBK+03].

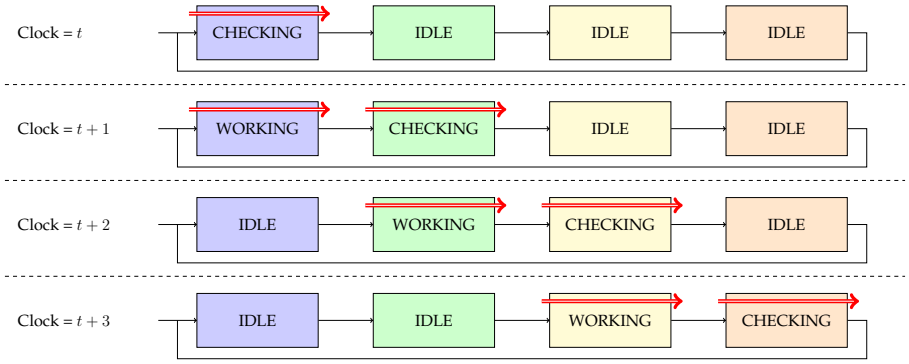


Figure 8.10 – Permanent fault detection scheme for AES.

8.2.5 Runtime Configurability

The proposed AES architecture is a 4-stage pipeline where each stage can be used independently of the others. As already noted, blocks can perform five different tasks:

- Compute a meaningful state;
- Be in idle state to save energy;
- Scramble power consumption;
- Check for transient faults by recomputing previous calculation;
- Check for permanent faults by computing a known input.

To explore all possible combinations, we proceed as follows: first, we generate all $5^4 = 625$ combinations (5 operations for 4 transformation blocks). We can consider a subset of these combinations if we work with 4 operations only, and remember that each E entry represents two actual options (tri-state or idle). This reduces the number of combinations to $4^4 = 256$. We eliminate all configurations that are circular permutations of others, *i.e.*, already counted configurations shifted in time. We also eliminate the meaningless configurations in which there isn't at least one block computing. All configurations having more than one permanent fault protection block at a time are removed as they don't add any extra protection. Finally, we eliminate the cases where a transient fault checking is not preceded by a computing block or by a permanent fault verification.

Table 8.1 shows that the design can perform 29 different task combinations, where *C* stands for computing, *E* stands for energy (power scrambling, idleness or any combination of these two if there are more than two *Es* in the considered configuration), *T* stands for transient fault checking and *P* stands for permanent fault checking. These options can be activated *during runtime* according to the system's constraints such as power consumption or speed. If there are no specific requirements, we recommend any of the four best configurations protecting against all attacks at once. These are singled-out in Table 8.1 by a $\star$.

Table 8.2 shows the number of configurations per protection goal. Note that for a given protection goal, different configurations can be alternated between executions without any performance loss.

8.3 Implementation Results

A 128-bit datapath AES encryption core was coded and tested in Verilog and compiled using Cadence *irun* tool. Cadence *RTL Compiler* was used to map the design into a 45nm *FreePDK* open cell digital library. Fig. 8.11 represents the inputs and outputs of the AES core. The module contains a general clock signal called `CLOCK_IN`, an asynchronous low-edge reset called `RESET_IN` and a `READY_IN` signal that flags the beginning of a new encryption. Plaintext is fed into the device *via* the 128-bit bus `TEXT_IN`, while the 128-bit key is fed to the system through the input called `KEY_IN`. The module outputs two signals: `TEXT_OUT`, which contains the resulting plaintext and `READY_OUT`, that represents a valid output.

Table 8.3 compares an unprotected AES core to the countermeasures described in this paper. The increase in terms of area is $\sim 6\%$ for the LFSR implementation and $\sim 4\%$ for the tri-state design. The LFSR

Table 8.1 – 29 possible AES-block configurations.

	Block 1	Block 2	Block 3	Block 4
	C	C	C	C
	C	C	C	E
	C	C	C	T
	C	C	C	P
	C	C	E	E
	C	C	E	T
	C	C	E	P
	C	C	T	T
	C	C	T	P
	C	C	P	E
	C	C	P	T
	C	E	C	E
	C	E	C	T
	C	E	C	P
	C	E	E	E
	C	E	E	T
	C	E	E	P
	C	E	T	T
⋆	C	E	T	P
	C	E	P	E
⋆	C	E	P	T
	C	T	C	P
	C	T	T	T
	C	T	T	P
⋆	C	T	P	E
	C	T	P	T
	C	P	E	E
⋆	C	P	E	T
	C	P	T	T

Table 8.2 – Number of configurations.

C	E	P	T	Configurations
4				1
3	1			1
1	3			1
3		1		1
3			1	1
1			3	1
2			2	1
1	1		2	1
1		2	1	1
2	2			2
1		1	2	2
2	1	1		3
1	2	1		3
1		1	2	3
2		1	1	3
1	1	1	1	4

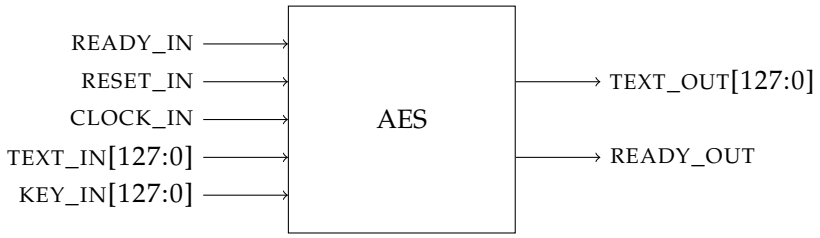


Figure 8.11 – AES design's inputs and outputs.

Table 8.3 – Unprotected AES, LFSR and tri-state buffer designs synthesized to the 45nm *FreePDK* Open Cell Library.

	Unprotected	**LFSR**	**Tri-state**
Area (μm^2)	61,581	65,194	64,243
Number of cells	10,643	11,035	11,162
sequential	783	911	787
inverters	1,483	1,614	1,493
logic	8,375	8,506	8,368
buffers	2	4	2
tri-state buffers	0	0	512
Total power (*mW*)	2.10	2.16	1.68
leakage power	1.20	1.28	1.26
dynamic power	0.89	0.87	0.41
Timing (*ps*)	645	645	806
Frequency (*GHz*)	1.55	1.55	1.24
Throughput (*Gbit/s*)	4.84	4.84	3.87

implementation showed almost no increase in terms of power consumption. Since tri-state buffers shut down three out of four blocks per clock, we expect a reduction in the power consumption. The tri-state design saves roughly 20% of power compared to the unprotected AES. As tri-state buffers tend to be slower, this design lost 20% in terms of clock frequency and throughput, while the LFSR version showed no speed loss, as expected.

Table 8.4 shows the three designs benchmarks in FPGA. They were coded in Verilog and synthesized to the Spartan3E-500 board using the Xilinx ISE 14.7 tool. LFSR and tri-state designs showed an area overhead of $\sim 15\%$ compared to the unprotected AES implementation. In terms of performance, LFSR design showed no loss, while the tri-state core lost $\sim 7\%$.

8.4 Conclusion

We described an unprotected AES implementation sliced in four clock cycles per round. Making use of this approach, we built on top of the unprotected core two power scrambling ideas to thwart side-channel attacks, such as CPA. We also demonstrated how the design can also prevent fault injection by recomputing its internal state values or by compromising one out of four blocks at each clock to compute the encryption of a known plaintext. We then exhibited simulation results and showed the comparison of the unprotected against the protected cores. The results confirm that the overhead in terms of area, power and performance are small, making this countermeasure attractive.

Moreover, the proposed AES architecture provides different options to tune the design into the user's need. Among 29 different configurations, examples include: to make the proposed AES a 4-stage pipeline

Table 8.4 – Spartan3E-500 utilization summary report.

	Unprotected	**LFSR**	**Tri-state**
Number of Occupied Slices	1,994	2,290	2,296
Number of Flip Flops	1,142	1,270	1,146
Number of LUTs	3,521	4,106	4,031
Timing (*ns*)	10.789	10.714	11.580
Frequency (*MHz*)	92.68	93.33	86.35
Throughput (*Mbit/s*)	289.3	291.3	269.6

(*i.e.*, compute four different plaintexts per execution), or to use three blocks to generate noise against power attacks, or to use one inactive block in the chain to recompute for encryption correctness.

8.5 Further Research: Ghost Data Attacks?

The footnote in Section 8.1 raises an interesting question: is it possible to exploit leakage from uselessly active circuit blocks to infer information about P, C or K? In this model the attacker is not allowed to access the side-channel information resulting from the actual computation of the active block (that we can assume to be ideally protected or not leaking) but only the side-channel information leaked by the three uselessly active blocks. To the best of our knowledge such attacks, that we call *ghost data attacks*, were never considered in the literature.

CHAPTER 9

CONCLUSION

This thesis explores vulnerabilities of cryptographic algorithms implemented with countermeasures. The thesis explored a signal derivative applicable against hiding side-channel countermeasures, key-dependent distributions imposing cryptosystems where plaintexts and ciphertexts are inaccessible, and security breaches introduced by CRT-RSA countermeasures. As an extension, the thesis also designed an algorithm calculating hardware-dependent constant to mask power consumption, and an AES implementation taking advantage of the unused blocks to thwart side-channel and fault attacks.

Our main empirical finding is that protected cryptographic algorithms remain susceptible to novel offensive techniques and to attacks with additional requirements.

One of the main thesis findings are subkey dependent Hamming weight distributions present in block ciphers. Coupled together with side-channel and fault information those distributions are used to mount blind attacks, *i.e.*, attacks that don't require the knowledge of plaintexts and ciphertexts. These blind attacks can be potentially applied against protocol-level countermeasures, such as key-derivation ladders.

Another significant result is an application of instantaneous frequency in side-channel attacks. This signal derivative supplements power trace parameters, namely, power amplitude and spectrum that are usually applied in power analysis attacks. Instantaneous frequency is a local characteristic assigned to all power trace samples which is tolerant to amplitude shifts and time shuffling countermeasures. Therefore, instantaneous frequency can be used as a side-channel vector or as a grouping factor to combine points with the same characteristic. This thesis shows that instantaneous frequency analysis presents specific benefits when applied against protected hardware implementations.

The thesis also presents practical attacks against the modern 32-bit ARM Cortex M3 general purpose microcontroller. Particularly, the work shows that implementing secure software running on vulnerable hardware is not an easy task. The chosen device could be tampered by laser from both the front and the back sides. Once the fault injection point was localized and the resulting error characterized an attack against protected CRT-RSA could be performed within seconds. We performed single and two fault attacks, which were successful against a conditional check and an infective countermeasure.

As an extension this thesis also present two collaborative defensive results. The first result is used to securely transfer digital data over leaky and noisy communication channels. The defensive strategy is to combine the transferred data with a certain camouflage value selected specifically for this channel. One of the important results is an algorithm finding this value. This algorithm can be used in other applications, for example, finding the least leaking S-box permutation.

The second result presents a hardware AES implementation unrolled in a set of four operations: AddRoundKey, SubBytes, ShiftRows, and MixColumns. While one block performs intermediate computations the other three blocks can be used to generate noise, and/or to recompute previous values. These blocks can also perform meaningful computations to increase algorithm's throughput or they can be switched off to reduce power consumption.

INDEX

BIBLIOGRAPHY

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula Rao, and Pankaj Rohatgi. The EM Side - Channel(s). In Burton Kaliski, Cetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer Berlin Heidelberg, 2003. 34, 37, 66

[ABCS06] Ross Anderson, Mike Bond, Jolyon Clulow, and Sergei Skorobogatov. Cryptographic Processors - A Survey. *Proceedings of the IEEE*, 94(2):357–369, Feb 2006. 56

[ABDM00] Mehdi-Laurent Akkar, Régis Bevan, Paul Dischamp, and Didier Moyart. Power Analysis, What Is Now Possible... In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 489–502. Springer Berlin Heidelberg, 2000. 37

[ABF+03] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and J-P Seifert. Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In Burton S. Kaliski, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 260–275. Springer Berlin Heidelberg, 2003. 50, 95, 96

[Ada97] Carlisle Adams. The CAST-128 Encryption Algorithm. `https://www.ipa.go.jp/security/rfc/RFC2144EN.html`, 1997. [Online; accessed 28-September-2015]. 24

[ADM+10] Michel Agoyan, Jean-Max Dutertre, Amir-Pasha Mirbaha, David Naccache, Anne-Lise Ribotta, and Assia Tria. Single-bit DFA Using Multiple-Byte Laser Fault Injection. In *IEEE International Conference on Technologies for Homeland Security - HST 2010*, pages 113–119, Nov 2010. 50

[ADN+10] Michel Agoyan, Jean-Max Dutertre, David Naccache, Bruno Robisson, and Assia Tria. When Clocks Fail: On Critical Paths and Clock Faults. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application - CARDIS 2010*, volume 6035 of *Lecture Notes in Computer Science*, pages 182–193. Springer Berlin Heidelberg, 2010. 48, 49, 91

[AES01] NIST AES. Advanced Encryption Standard. *Federal Information Processing Standard, FIPS-197*, 12, 2001. 16, 21, 80

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer Berlin Heidelberg, 2001. 46

[AOP+09] Ali Alaeldine, Thomas Ordas, Richard Perdriau, Philippe Maurine, Mohamed Ramdani, Lionel Torres, and M'hammed Drissi. Assessment of the Immunity of Unshielded Multi-Core Integrated Circuits to Near-Field Injection. In *20th International Zurich Symposium on Electromagnetic Compatibility*, pages 361–364, Jan 2009. 51

[APRV07] Massimo Alioto, Massimo Poli, Santina Rocchi, and Valerio Vignoli. Mixed Techniques to Protect Precharged Busses against Differential Power Analysis Attacks. In *IEEE International Symposium on Circuits and Systems - ISCAS 2007*, pages 861–864, May 2007. 46

[APSQ06] Cédric Archambeau, Eric Peeters, Francois-Xavier Standaert, and Jean-Jacques Quisquater. Template Attacks in Principal Subspaces. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin Heidelberg, 2006. 44

[ARM] ARM Cortex M3 Description. http://www.arm.com/products/processors/cortex-m/cortex-m3.php. [Online; accessed 28-September-2015]. 91

[BBA+12] Pierre Bayon, Lilian Bossuet, Alain Aubert, Viktor Fischer, François Poucheret, Bruno Robisson, and Philippe Maurine. Contactless Electromagnetic Active Attack on Ring Oscillator Based True Random Number Generator. In Werner Schindler and Sorin A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - COSADE 2012*, pages 151–166, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 51

[BBK+03] Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, and Vincenzo Piuri. Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard. *IEEE Transactions on Computers*, 52(4):492–505, April 2003. 124

[BBKP02] Guido Bertoni, Luca Breveglieri, Israel Koren, and Vincenzo Piuri. Fault Detection in the Advanced Encryption Standard. In *Proceedings of the Conference on Massively Parallel Computing Systems*, pages 92–97, 2002. 124

[BBPP09] Alessandro Barenghi, Guido Bertoni, Emanuele Parrinello, and Gerardo Pelosi. Low Voltage Fault Attacks on the RSA Cryptosystem. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2009*, pages 23–31, Sept 2009. 91

[BCF03] Eric Brier, Christophe Clavier, and Olivier Francis. Optimal Statistical Power Analysis. https://eprint.iacr.org/2003/152.pdf, 2003. [Online; accessed 31-July-2016]. 104

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer Berlin Heidelberg, 2004. 37, 38, 40, 66, 68, 78

[BDGP14] Luk Bettale, Emmanuelle Dottax, Laurie Genelle, and Gilles Piret. Collision-Correlation Attack Against a First-Order Masking Scheme for MAC Based on SHA-3. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - COSADE 2014*, volume 8622 of *Lecture Notes in Computer Science*, pages 129–143. Springer International Publishing, 2014. 38

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer Berlin Heidelberg, 1997. 16, 48, 49, 90, 97, 99

[BECN+06] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. *Proceedings of the IEEE*, 94(2):370–382, Feb 2006. 50

[Ben73] Charles Bennett. Logical Reversibility of Computation. *IBM J. Res. Dev.*, 17(6):525–532, November 1973. 30

[BF93] John Bunge and Michael Fitzpatrick. Estimating the Number of Species: a Review. *Journal of the American Statistical Association*, 88(421):364–373, 1993. 84

[BGV11] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2011*, pages 105–114, Sept 2011. 50

[BHT09] Arnaud Boscher, Helena Handschuh, and Elena Trichina. Blinded Fault Resistant Exponentiation Revisited. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2009*, pages 3–9, Sept 2009. 57

[BHT10] Arnaud Boscher, Helena Handschuh, and Elena Trichina. Fault Resistant RSA Signatures: Chinese Remaindering in Both Directions. http://eprint.iacr.org/2010/038.pdf, 2010. [Online; accessed 28-September-2015]. 90, 95, 96

[BK06] Johannes Blömer and Volker Krummel. Fault Based Collision Attacks on AES. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2006*, volume 4236 of *Lecture Notes in Computer Science*, pages 106–120. Springer Berlin Heidelberg, 2006. 52, 54

[BK07] Alex Biryukov and Dmitry Khovratovich. Two New Techniques of Side-Channel Cryptanalysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 195–208. Springer Berlin Heidelberg, 2007. 38

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J.B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer Berlin Heidelberg, 2007. 80

[BKMG07] Bradley Battista, Camelia Knapp, Tom McGee, and Vaughn Goebel. Application of the Empirical Mode Decomposition and Hilbert-Huang Transform to Seismic Reflection Data. *Geophysics*, 72(2):H29–H37, 2007. 73

[BKMG12] Bradley Battista, Camelia Knapp, Tom McGee, and Vaughn Goebel. Matlab Program Demonstrating Performing the Empirical Mode Decomposition and Hilbert-Huang Transform on Seismic Reflection Data. http://software.seg.org/2007/0003/mat_emd.zip, August 2012. [Online; accessed 28-September-2015]. 73

[BL96] Dan Boneh and Richard Lipton. New Threat Model Breaks Crypto Codes. https://cryptome.org/jya/belcor.txt, 1996. [Online; accessed 31-July-2016]. 55, 95

[BLOW10] Kean Hong Boey, Yingxi Lu, Maire O'Neill, and Roger Woods. Random Clock Against Differential Power Analysis. In *IEEE Asia Pacific Conference on Circuits and Systems - APCCAS 2010*, pages 756–759, Dec 2010. 46

[BM06] Joseph Bonneau and Ilya Mironov. Cache-Collision Timing Attacks Against AES. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 201–215. Springer Berlin Heidelberg, 2006. 37

[BMS86] Eric Bach, Gary Miller, and Jeffrey Shallit. Sums of Divisors, Perfect Numbers and Factoring. *SIAM Journal on Computing*, 15(4):1143–1154, 1986. 25

[Boa92] Boualem Boashash. Estimating and Interpreting the Instantaneous Frequency of a Signal. I. Fundamentals. *Proceedings of the IEEE*, 80(4):520 –538, April 1992. 61

[Boi04] Christian Boit. Fundamentals of Photon Emission (PEM) in Silicon–Electroluminescence for Analysis of Electronic Circuit and Device Functionality. *Microelectronics Failure Analysis: Desk Reference*, pages 357–369, 2004. 35

[BQK+13] Eric Brier, Fortier Quentin, Roman Korkikian, K. W. Magld, David Naccache, Guilherme Ozari de Almeida, Adrien Pommellet, A. H. Ragab, and Jean Vuillemin. Defensive Leakage Camouflage. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications - CARDIS 2012*, pages 277–295, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 159

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott Vanstone, editors, *Advances in Cryptology - CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer Berlin Heidelberg, 1991. 78

[BS97] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In Burton S. Kaliski, editor, *Advances in Cryptology - CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer Berlin Heidelberg, 1997. 15

[BS03] Johannes Blömer and Jean-Pierre Seifert. Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In Rebecca Wright, editor, *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer Berlin Heidelberg, 2003. 52, 85

[BSH75] Daniel Binder, Edward C. Smith, and A.B. Holman. Satellite Anomalies from Galactic Cosmic Rays. *IEEE Transactions on Nuclear Science*, 22(6):2675–2680, Dec 1975. 49

[BSR06] Fraidy Bouesse, Gilles Sicard, and Marc Renaudin. Path Swapping Method to Improve DPA Resistance of Quasi Delay Insensitive Asynchronous Circuits. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 384–398. Springer Berlin Heidelberg, 2006. 46

[BTL13] Guillaume Bouffard, Bhagyalekshmy N. Thampi, and Jean-Louis Lanet. Detecting Laser Fault Injection for Smart Cards Using Security Automata. In Sabu M. Thampi, Pradeep K. Atrey, Chun-I Fan, and Gregorio Martinez Perez, editors, *Security in Computing and Communications - SSCC 2013*, pages 18–29, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 56

[BvdPSY14] Naomi Benger, Joop van de Pol, Nigel Smart, and Yuval Yarom. "Ooh Aah... Just a Little Bit" : A Small Amount of Side Channel Can Go a Long Way. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 75–92. Springer Berlin Heidelberg, 2014. 37

[BZ07] Karthik Baddam and Mark Zwolinski. Evaluation of Dynamic Voltage and Frequency Scaling as a Differential Power Analysis Countermeasure. In *20th International Conference on VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems*, pages 854–862, Jan 2007. 60, 73

[CDG+14] Jean-Michel Cioranesco, Jean-Luc Danger, Tarik Graba, Sylvain Guilley, Yves Mathieu, David Naccache, and Xuan Thuy Ngo. Cryptographically Secure Shields. In *IEEE International Symposium on Hardware-Oriented Security and Trust - HOST 2014*, pages 25–31, May 2014. 56

[Cha83] David Chaum. Blind Signatures for Untraceable Payments. In *Advances in cryptology*, pages 199–203. Springer, 1983. 46

[Cha13] Amit Chaudhry. Nanoscale Effects: Gate Oxide Leakage Currents. In *Fundamentals of Nanoscaled Field Effect Transistors*, pages 25–36. Springer New York, 2013. 33

[CJ05] Mathieu Ciet and Marc Joye. Practical Fault Countermeasures for Chinese Remaindering Based RSA. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2005*, volume 5, pages 124–132. Springer Berlin Heidelberg, 2005. 90

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An Efficient Method for Random Delay Generation in Embedded Software. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, pages 156–170, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. 57

[CK14a] Omar Choudary and Markus Kuhn. Efficient Template Attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - CARDIS 2013*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer International Publishing, 2014. 44

[CK14b] Omar Choudary and Markus G. Kuhn. Template Attacks on Different Devices. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - COSADE 2014*, volume 8622 of *Lecture Notes in Computer Science*, pages 179–198. Springer International Publishing, 2014. 43

[CKN01] Jean-Sébasticn Coron, Paul Kocher, and David Naccache. Statistics and Secret Leakage. In Yair Frankel, editor, *Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 157–173. Springer Berlin Heidelberg, 2001. 78

[CP02] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer Berlin Heidelberg, 2002. 15, 45

[CPB+14] Rafael Boix Carpi, Stjepan Picek, Lejla Batina, Federico Menarini, Domagoj Jakobovic, and Marin Golub. Glitch It If You Can: Parameter Search Strategies for Successful Fault Injection. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - CARDIS 2013*, pages 236–252, Cham, 2014. Springer International Publishing. 50

[CRR03] Suresh Chari, Josyula Rao, and Pankaj Rohatgi. Template Attacks. In Burton Kaliski, Cetin Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer Berlin Heidelberg, 2003. 43, 44

[Dar04] John Coleman Darnell. The Enigmatic Netherworld Books of the Solar-Osirian Unity. *Cryptographic Compositions in the Tombs of Tutankhamun, Ramesses VI and Ramesses IX*, 2004. 14

[DBCR+10] Jerome Di-Battista, Jean-Christophe Courrege, Bruno Rouzeyre, Lionel Torres, and Philippe Perdu. When Failure Analysis Meets Side-Channel Attacks. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems - CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 188–202. Springer Berlin Heidelberg, 2010. 35

[DBVKOS00] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational Geometry*. Springer, 2000. 108

[DGRS09] Emmanuelle Dottax, Christophe Giraud, Matthieu Rivain, and Yannick Sierra. On Second-Order Fault Analysis Resistance for CRT-RSA Implementations. In Olivier Markowitch, Angelos Bilas, Jaap-Henk Hoepman, ChrisJ. Mitchell, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practice. Smart Devices, Pervasive Systems, and Ubiquitous Networks*, volume 5746 of *Lecture Notes in Computer Science*, pages 68–83. Springer Berlin Heidelberg, 2009. 51, 90, 102

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 24

[DLV03] Pierre Dusart, Gilles Letourneux, and Olivier Vivolo. Differential Fault Analysis on A.E.S. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *Applied Cryptography and Network Security*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer Berlin Heidelberg, 2003. 52, 54

[DM03] Paul E. Dodd and Lloyd W. Massengill. Basic Mechanisms and Modeling of Single-Event Upset in Digital Microelectronics. *IEEE Transactions on Nuclear Science*, 50(3):583–602, June 2003. 49, 50

[DN07] Boray S. Deepaksubramanyan and Adrian Nunez. Analysis of Subthreshold Leakage Reduction in CMOS Digital Circuits. In *50th Midwest Symposium on Circuits and Systems, 2007. MWSCAS 2007*, pages 1400–1404, Aug 2007. 33

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate Side Channel Attacks and Leakage Modeling. *Journal of Cryptographic Engineering*, 1(2):123–144, 2011. 37

[EABK14] Edward C. Epp, Sachin Agrawal, Michael Bergeron, and Hormuzd Khosravi. Content Protection Key Management, 2014. 57

[Ell97] James H Ellis. The Story of Non-Secret Encryption, 1987. *Released by CSEG in 1997*, 1997. 24

[Fei73] Horst Feistel. Cryptography and Computer Privacy. *Scientific american*, 228:15–23, 1973. 19

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, Inc., 3rd edition, 1968. 83

[FIP95] PUB FIPS. 180-1. Secure Hash Standard. *National Institute of Standards and Technology*, 17, 1995. 16

[FJLT13] Thomas Fuhr, Eliane Jaulmes, Victor Lomne, and Adrian Thillard. Fault Attacks on AES with Faulty Ciphertexts Only. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2013*, pages 108–118, Aug 2013. 52, 53, 55, 85

[FML+03] Jacques J. A. Fournier, Simon Moore, Huiyun Li, Robert Mullins, and George Taylor. Security Evaluation of Asynchronous Circuits. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, pages 137–151. Springer Berlin Heidelberg, 2003. 46

[FS04] Daniel M. Fleetwood and Ron D. Schrimpf. *Radiation Effects and Soft Errors in Integrated Circuits and Electronic Devices*. Selected topics in electronics and systems. World Scientific Pub., 2004. 49

[FT09] Toshinori Fukunaga and Junko Takahashi. Practical Fault Attack on a Cryptographic LSI with ISO/IEC 18033-3 Block Ciphers. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2009*, pages 84–92, Sept 2009. 49

[Gal97] Évariste Galois. *Œuvres Mathématiques d'Évariste Galois*. Gauthier-Villars et fils, 1897. 18

[Gan] Amlan Ganguly. EE466: VLSI Design: Power Dissipation. `www.eecs.wsu.edu/~ee466/fall09/lecture_10.ppt`. [Online; accessed 28-September-2015]. 30

[Gar99] Bernd Gartner. Fast and Robust Smallest Enclosing Balls. In *Algorithms - ESA'99*, volume 1643 of *Lecture Notes in Computer Science*, pages 325–338. Springer Berlin Heidelberg, 1999. 109

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual Information Analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer Berlin Heidelberg, 2008. 38, 66

[GDMPV09] Benedikt Gierlichs, Elke De Mulder, Bart Preneel, and Ingrid Verbauwhede. Empirical Comparison of Side Channel Analysis Distinguishers on DES in Hardware. In *European Conference on Circuit Theory and Design - ECCTD 2009*, pages 391–394, Aug 2009. 39

[GHJ92] Cynthia A. Gossett, Barrie W. Hughlock, and Allan H. Johnston. Laser Simulation of Single-Particle Effects. *IEEE Transactions on Nuclear Science*, 39(6):1647–1653, Dec 1992. 50

[GHT05] Catherine Gebotys, Simon Ho, and Chin Chi Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 250–264. Springer Berlin Heidelberg, 2005. 34

[Gir05a] Christophe Giraud. DFA on AES. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer Berlin Heidelberg, 2005. 52, 53, 54

[Gir05b] Christophe Giraud. Fault Resistant RSA Implementation. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2005*, pages 142–151, 2005. 95

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin K. Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer Berlin Heidelberg, 2001. 34

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on computing*, 18(1):186–208, 1989. 78

[GNL12] Zheng Gong, Svetla Nikova, and YeeWei Law. KLEIN: A New Family of Lightweight Block Ciphers. In Ari Juels and Christof Paar, editors, *RFID. Security and Privacy*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2012. 80

[GOK+05] Frank Gurkaynak, Stephan Oetiker, Hubert Kaeslin, Norbert Felber, and Wolfgang Fichtner. Improving DPA Security by Using Globally-Asynchronous Locally-Synchronous Systems.

In *Proceedings of the 31st European Solid-State Circuits Conference - ESSCIRC 2005*, pages 407–410, Sept 2005. 46

[Gol98] Oded Goldreich. Secure Multi-Party Computation. *Manuscript. Preliminary version*, 1998. 78

[Goo] Google Patents. https://patents.google.com/. [Online; accessed 28-September-2015]. 29

[GP08] Martin Goldack and Ing Christof Paar. Side-Channel Based Reverse Engineering for Microcontrollers. *Master's thesis, Ruhr-Universität Bochum, Germany*, 2008. 87

[GPLL09] Catherine Godlewski, Vincent Pouget, Dean Lewis, and Mathieu Lisart. Electrical Modeling of the Effect of Beam Profile for Pulsed Laser Fault Injection. *Microelectronics Reliability*, 49(9):1143–1147, 2009. 94

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matt Robshaw. The LED Block Cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer Berlin Heidelberg, 2011. 77, 80

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461. Springer Berlin Heidelberg, 2014. 35, 37

[GT04] Christophe Giraud and Hugues Thiebeauld. A Survey on Fault Attacks. In Jean-Jacques Quisquater, Pierre Paradinas, Yves Deswarte, and AnasAbou El Kalam, editors, *Smart Card Research and Advanced Applications - CARDIS 2004*, volume 153 of *IFIP International Federation for Information Processing*, pages 159–176. Springer US, 2004. 51

[GTC05] Catherine Gebotys, Chin Chi Tiu, and Xi Xi Chen. A Countermeasure for EM Attack of a Wireless PDA. In *International Conference on Information Technology: Coding and Computing - ITCC 2005*, volume 1, pages 544 – 549, April 2005. 34

[Har68] Bernard Harris. Statistical Inference in the Classical Occupancy Problem Unbiased Estimation of the Number of Classes. *Journal of the American Statistical Association*, pages 837–847, 1968. 83

[Haz01] Michiel Hazewinkel. Kolmogorov-Smirnov Test. *Encyclopedia of Mathematics, Springer, ISBN*, pages 978–1, 2001. 78

[Hem04] Ludger Hemme. A Differential Fault Attack Against Early Rounds of (Triple-)DES. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 254–267. Springer Berlin Heidelberg, 2004. 52

[HHM+14] Naofumi Homma, Yu-ichi Hayashi, Noriyuki Miura, Daisuke Fujimoto, Daichi Tanaka, Makoto Nagata, and Takafumi Aoki. EM Attack Is Non-invasive? - Design Methodology and Validity Verification of EM Attack Sensor. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2014. 46

[HNT+13] Clemens Helfmeier, Dmitry Nedospasov, Christopher Tarnovsky, Jan Starbug Krissler, Christian Boit, and Jean-Pierre Seifert. Breaking and Entering Through the Silicon. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS'13*, CCS'13, pages 733–744, New York, NY, USA, 2013. ACM. 48

[HPSS08] Jeffrey Hoffstein, Jill Catherine Pipher, Joseph H Silverman, and Joseph H Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2008. 17, 24

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good Is Not Good Enough. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 55–74. Springer Berlin Heidelberg, 2014. 38

[HS14] Michael Hutter and Jörn-Marc Schmidt. The Temperature Side Channel and Heating Fault Attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - CARDIS 2014*, volume 8419 of *Lecture Notes in Computer Science*, pages 219–235. Springer International Publishing, 2014. 37

[HSL+98] Norden Huang, Zheng Shen, Steven Long, Manli Wu, Hsing Shih, Nai-Chyuan Yen, Quanan Zheng, Chi Chao Tung, and Henry Liu. The Empirical Mode Decomposition and the Hilbert Spectrum for Nonlinear and Non-Stationary Time Series Analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, 1998. 63

[HSS12] Annelie Heuser, Werner Schindler, and Marc Stöttinger. Revealing Side-Channel Issues of Complex Circuits by Enhanced Leakage Models. In *Design, Automation Test in Europe Conference Exhibition - DATE 2012*, pages 1179–1184, March 2012. 37

[HSTV06] David D. Hwang, Patrick Schaumont, Kris Tiri, and Ingrid Verbauwhede. Securing embedded systems. *Security Privacy, IEEE*, 4(2):40–49, March 2006. 15

[IEE] IEEE Xplore Digital Library. http://ieeexplore.ieee.org/Xplore/home.jsp. [Online; accessed 28-September-2015]. 29

[ISYT13] Hiroaki Igarashi, Youhua Shi, Masao Yanagisawa, and Nozomu Togawa. Concurrent Faulty Clock Detection for Crypto Circuits Against Clock Glitch Based DFA. In *IEEE International Symposium on Circuits and Systems - ISCAS 2013*, pages 1432–1435, May 2013. 56

[ITT02] Kouichi Itoh, Masahiko Takenaka, and Naoya Torii. DPA Countermeasure Based on the "Masking Method". In Kwangjo Kim, editor, *Information Security and Cryptology - ICISC 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 440–456. Springer Berlin Heidelberg, 2002. 46

[JJKB16] D. Jennifer Judy and V. S. Kanchana Bhaaskaran. Subthreshold Operation of Energy Recovery Circuits. In Chandra Suresh Satapathy, Bheema N. Rao, Srinivas S. Kumar, Dharma C. Raj, Malleswara V. Rao, and K. G. V. Sarma, editors, *Proceedings of Microelectronics, Electromagnetics and Telecommunications - ICMEET 2015*, volume 372 of *Lecture Notes in Electrical Engineering*, pages 1–12. Springer India, 2016. 45

[Joy09] Mark Joye. Protecting RSA against Fault Attacks: The Embedding Method. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2009*, pages 41–45, Sept 2009. 96

[JPY01] Marc Joye, Pascal Paillier, and Sung-Ming Yen. Secure Evaluation of Modular Functions. In *International workshop on cryptology and network security*, volume 2001, pages 227–229. Citeseer, 2001. 90

[JRE12] Darshana Jayasinghe, Roshan Ragel, and Dhammika Elkaduwe. Constant Time Encryption as a Countermeasure Against Remote Cache Timing Attacks. In *6th International Conference on Information and Automation for Sustainability - ICIAFS 2012*, pages 129–134, Sept 2012. 45

[JT12] Marc Joye and Michael Tunstall, editors. *Fault Analysis in Cryptography*. Information Security and Cryptography. Springer, 2012. 49, 52, 55, 87

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K Lenstra, Emmanuel Thomé, Joppe W Bos, Pierrick Gaudry, Alexander Kruppa, Peter L Montgomery, Dag Arne Osvik, et al. Factorization of a 768-bit RSA modulus. In *Annual Cryptology Conference*, pages 333–350. Springer, 2010. 26

[Kas63] Friedrich Wilhelm Kasiski. Die Geheimschriften und die Dechiffrir-kunst. *ES Mittler und Sohn*, 1863. 14

[KD09] Boris Kopf and Markus Durmuth. A Provably Secure and Efficient Countermeasure against Timing Attacks. In *IEEE Computer Security Foundations Symposium - CSF'09*, pages 324–335, July 2009. 45

[KDK+14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping Bits in Memory Without Accessing Them:

An Experimental Study of DRAM Disturbance Errors. In *ACM/IEEE 41st International Symposium on Computer Architecture - ISCA 2014*, pages 361–372, June 2014. 15

[Key75] Robert W. Keyes. Physical Limits in Digital Electronics. In *IEEE Proceedings*, volume 63, pages 740–767, May 1975. 30

[KFA+95] Johan Karlsson, Peter Folkesson, Jean Arlat, Yves Crouzet, and Günther Leber. Integration and Comparison of Three Physical Fault Injection Techniques. In Brian Randell, Jean-Claude Laprie, Hermann Kopetz, and Bev Littlewood, editors, *Predictably Dependable Computing Systems*, pages 309–327, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. 50

[KGS+11] Armin Krieg, Johannes Grinschgl, Christian Steger, Reinhold Weiss, and Josef Haid. A Side Channel Attack Countermeasure Using System-On-Chip Power Profile Scrambling. In *IEEE 17th International On-Line Testing Symposium - IOLTS 2011*, pages 222–227, July 2011. 46, 60, 73

[KHL11] HeeSeok Kim, Seokhie Hong, and Jongin Lim. A Fast and Provably Secure Higher-Order Masking of AES S-Box. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 95–107. Springer Berlin Heidelberg, 2011. 87

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer Berlin Heidelberg, 1999. 16, 37, 38, 40, 45, 66, 78, 119

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *USENIX workshop on Smartcard Technology*, volume 12, pages 9–20, 1999. 50

[KK06] Ranjith Kumar and Volkan Kursun. Reversed Temperature-Dependent Propagation Delay Characteristics in Nanometer CMOS Circuits. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 53(10):1078–1082, Oct 2006. 48, 50

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography: Principles and Protocols*. Chapman & Hall/CRC Cryptography and Network Security Series. Taylor & Francis, 2007. 22

[KM08] Mehrdad Khatir and Amir Moradi. Secure Adiabatic Logic: a Low-Energy DPA-Resistant Logic Style. https://eprint.iacr.org/2008/123.pdf, 2008. [Online; accessed 01-August-2016]. 45

[KM10] Daniel Kaslovsky and Francois Meyer. Noise Corruption of Empirical Mode Decomposition and Its Effect on Instantaneous Frequency. http://arxiv.org/pdf/1008.4176v1, August 2010. [Online; accessed 01-August-2016]. 61

[KNdAdC14] Roman Korkikian, David Naccache, Guilherme Ozari de Almeida, and Rodrigo Portella do Canto. Practical Instantaneous Frequency Analysis Experiments. In Mohammad S. Obaidat and Joaquim Filipe, editors, *E-Business and Telecommunications*, volume 456 of *Communications in Computer and Information Science*, pages 17–34. Springer Berlin Heidelberg, 2014. 37, 160

[KNSS13] Juliane Krämer, Dmitry Nedospasov, Alexander Schlösser, and Jean-Pierre Seifert. Differential Photonic Emission Analysis. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - COSADE 2013*, volume 7864 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2013. 35, 37

[Knu98] Donald Knuth. *The Art of Computer Programming, vol. 3, Sorting and Searching*. Addison Wesley, 1998. 106

[Koc96] Paul Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer Berlin Heidelberg, 1996. 16, 37, 60, 119

[Koc05] Paul Kocher. Design and Validation Strategies for Obtaining Assurance in Countermeasures to Power Analysis and Related attacks. In *Proceedings of the NIST Physical Security Workshop*, 2005. 46

[KOP09] Timo Kasper, David Oswald, and Christof Paar. EM Side-Channel Attacks on Commercial Contactless Smartcards Using Low-Cost Equipment. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications*, volume 5932 of *Lecture Notes in Computer Science*, pages 79–93. Springer Berlin Heidelberg, 2009. 37

[KPN14] Roman Korkikian, Sylvain Pelissier, and David Naccache. Blind Fault Attack Against SPN Ciphers. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2014*, FDTC '14, pages 94–103. IEEE Computer Society, 2014. 39, 160

[KQ07a] Chong Hee Kim and Jean-Jacques Quisquater. Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures. In Damien Sauveron, Konstantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, volume 4462 of *Lecture Notes in Computer Science*, pages 215–228. Springer Berlin Heidelberg, 2007. 90, 95, 96, 102

[KQ07b] Chong Hee Kim and Jean-Jaques Quisquater. How Can We Overcome Both Side Channel Analysis and Fault Attacks on RSA-CRT? In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2007*, pages 21–29, Sept 2007. 90, 102

[KWMK02] Ramesh Karri, Kaijie Wu, Piyush Mishra, and Yongkook Kim. Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(12):1509–1517, Dec 2002. 57

[Lab12] RSA Laboratories. PKCS # 1 v2.2: RSA Cryptography Standard. http://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf, October 2012. [Online; accessed 01-August-2016]. 25

[LAM+07] Régis Leveugle, Abdelaziz Ammari, V. Maingot, E. Teyssou, Pascal Moitrel, Christophe Mourtel, Nathalie Feyt, J-B Rigaud, and Assia Tria. Experimental Evaluation of Protections Against Laser-induced Faults and Consequences on Fault Modeling. In *Design, Automation Test in Europe Conference Exhibition - DATE 2007*, pages 1–6, April 2007. 94

[LDLL14] Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. Using the Joint Distributions of a Cryptographic Function in Side Channel Analysis. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - COSADE 2014*, volume 8622 of *Lecture Notes in Computer Science*, pages 199–213. Springer International Publishing, 2014. 39

[Len96] Arjen K. Lenstra. Memo on RSA Signature Generation in the Presence of Faults. https://infoscience.epfl.ch/record/164524/files/nscan20.PDF, 1996. [Online; accessed 01-August-2016]. 55, 95

[LLJMP93] Arjen K. Lenstra, Hendrik W. Lenstra J., Mark S. Manasse, and John M. Pollard. The Number Field Sieve. In *The development of the number field sieve*, pages 11–42. Springer, 1993. 26

[LPM+06] Damien Leroy, Stanislaw J. Piestrak, Fabrice Monteiro, Abbas Dandache, Stéphane Rossignol, and Pascal Moitrel. Characterizing Laser-Induced Pulses in ICs: Methodology and Results. In *12th IEEE International On-Line Testing Symposium - IOLTS 2006*, page 6, 2006. 94

[LRD+12] Ronan Lashermes, Guillaume Reymond, Jean-Max Dutertre, Jacques Fournier, Bruno Robisson, and Assia Tria. A DFA on AES Based on the Entropy of Error Distributions. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2012*, pages 34–43, Sept 2012. 53, 55

[LRT12] Victor Lomne, Thomas Roche, and Adrian Thillard. On the Need of Randomness in Fault Attack Countermeasures - Application to AES. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2012*, pages 85–94, Sept 2012. 87

[LSG+10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault Sensitivity Analysis. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems - CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 320–334. Springer Berlin Heidelberg, 2010. 48

[Luo10] Qiasi Luo. Enhance Multi-bit Spectral Analysis on Hiding in Temporal Dimension. In Dieter Gollmann, Jean-Louis Lanet, and Julien Iguchi-Cartigny, editors, *Smart Card Research and Advanced Application - CARDIS 2010*, volume 6035 of *Lecture Notes in Computer Science*, pages 13–23. Springer Berlin Heidelberg, 2010. 34

[Man03] Stefan Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002*, volume 2587 of *Lecture Notes in Computer Science*, pages 343–358. Springer Berlin Heidelberg, 2003. 37, 38

[Mat94] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Berlin Heidelberg, 1994. 78

[Mau05] Ueli Maurer. Abstract Models of Computation in Cryptography. In Nigel P. Smart, editor, *Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2005. 77

[MC80] Carver Mead and Lynn Conway. *Introduction to VLSI Systems*. Addison-Wesley, 1980. 30

[MDH+13] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2013*, pages 77–88, Aug 2013. 85

[Mes00] Thomas Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In Çtin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer Berlin Heidelberg, 2000. 78

[MG10] Edgar Mateos and Catherine Gebotys. A New Correlation Frequency Analysis of the Side Channel. In *Proceedings of the 5th Workshop on Embedded Systems Security*, WESS '10, pages 4:1–4:8, New York, NY, USA, 2010. ACM. 37

[MG11] Edgar Mateos and Catherine Gebotys. Side Channel Analysis using Giant Magneto-Resistive (GMR) Sensors. In *Constructive Side-Channel Analysis and Secure Design - COSADE 2011*, pages 42–49, February 2011. 34

[MKK00] James Massey, Gurgen Khachatrian, and Melsik Kuregian. Nomination of SAFER++ as Candidate Algorithm for the New European Schemes for Signatures, Integrity, and Encryption (NESSIE), 2000. 77, 80

[MM09] A. Theodore Markettos and Simon W. Moore. The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, CHES '09, pages 317–331, Berlin, Heidelberg, 2009. Springer-Verlag. 51

[MM14] Matthew Mayhew and Radu Muresan. On-Chip Nanoscale Capacitor Decoupling Architectures for Hardware Security. *IEEE Transactions on Emerging Topics in Computing*, 2(1):4–15, March 2014. 45

[MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems - CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 125–139. Springer Berlin Heidelberg, 2010. 38

[Mog08] Manuel Mogollon. *Cryptography and Security Services: Mechanisms and Applications: Mechanisms and Applications*. IGI Global, 2008. 14

[Mol06] Richard A. Mollin. *An Introduction to Cryptography*. CRC Press, 2006. 18

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer-Verlag, 2007. 119

[Mor14a] Amir Moradi. Side-Channel Leakage through Static Power. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 562–579. Springer Berlin Heidelberg, 2014. 31, 37

[Mor14b] Timothy Prickett Morgan. Oracle Cranks Up The Cores To 32 With Sparc M7 Chip. http://www.enterprisetech.com/2014/08/13/oracle-cranks-cores-32-sparc-m7-chip/, August 2014. [Online; accessed 01-August-2016]. 34

[MP13] Gary L. Mullen and Daniel Panario. *Handbook of Finite Fields*. CRC Press, 2013. 17

[MPM+96] Don Myerscough, Don Ploger, Lynn McCarthy, Hallie Hopper, and Vicki G Fegers. Cryptography: Cracking Codes. *The Mathematics Teacher*, 89(9):743, 1996. 14

[MPR+11] Marcel Medwed, Christoph Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh Re-keying II: Securing Multiple Parties against Side-Channel and Fault Attacks. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - CARDIS 2011*, pages 115–132. Springer Berlin Heidelberg, 2011. 46

[MRL+06] Yannick Monnet, Marc Renaudin, Régis Leveugle, Nathalie Feyt, Pascal Moitrel, and M'Buwa F. Nzenguet. Practical Evaluation of Fault Countermeasures on Asynchronous DES Crypto Processor. In *12th IEEE International On-Line Testing Symposium - IOLTS 2006*, pages 6 pp.–, 2006. 56, 94

[MRM00] Joao Baptista Martins, Ricardo Reis, and Jose Monteiro. Capacitance and Power Modeling at Logic-Level. *Power*, 1(2):2, 2000. 31

[MS08] Marcel Medwed and Jorn-Marc Schmidt. A Generic Fault Countermeasure Providing Data and Program Flow Integrity. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2008*, pages 68–73, Aug 2008. 57

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology: Third International Conference on Cryptology in Africa - AFRICACRYPT 2010*, pages 279–296, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 57

[MSQL05] François Macé, François-Xavier Standaert, Jean-Jacques Quisquater, and Jean-Didier Legat. A Design Methodology for Secured ICs Using Dynamic Current Mode Logic. In Vassilis Paliouras, Johan Vounckx, and Diederik Verkest, editors, *15th International Workshop on Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation - PATMOS 2005*, pages 550–560. Springer Berlin Heidelberg, 2005. 45

[MSS06] Amir Moradi, Mohammad T. Manzuri Shalmani, and Mahmoud Salmasizadeh. A Generalized Method of Differential Fault Attack Against AES Cryptosystem. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 91–100. Springer Berlin Heidelberg, 2006. 54

[MSY06] Tal G. Malkin, François-Xavier Standaert, and Moti Yung. A Comparative Cost/Security Analysis of Fault Attack Countermeasures. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2006*, volume 4236 of *Lecture Notes in Computer Science*, pages 159–172. Springer Berlin Heidelberg, 2006. 56

[Muk09] Debdeep Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009*, volume

5580 of *Lecture Notes in Computer Science*, pages 421–434. Springer Berlin Heidelberg, 2009. 54

[MVL07] Paolo Maistri, Pierre Vanhauwaert, and Régis Leveugle. A Novel Double-Data-Rate AES Architecture Resistant against Fault Injection. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2007*, pages 54–61, Sept 2007. 57

[MVOV96] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC press, 1996. 17

[MW79] Timothy May and Murray Woods. Alpha-Particle-Induced Soft Errors in Dynamic Memories. *IEEE Transactions on Electron Devices*, 26(1):2–9, Jan 1979. 49

[Nik01] Mikhail S. Nikulin. Hellinger Distance. *Encyclopedia of Mathematics*, 2001. 78

[NM14] Mădălin Neagu and Liviu Miclea. Protecting Cache Memories Through Data Scrambling Technique. In *Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*, pages 297–303. IEEE, 2014. 57

[NN06] Valtteri Niemi and Kaisa Nyberg. *UMTS Security*. Wiley, 2006. 52

[Nov02] Roman Novak. SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 252–262. Springer Berlin Heidelberg, 2002. 38

[Nov03] Roman Novak. Side-Channel Based Reverse Engineering of Secret Algorithms. In *Proceedings of the Twelfth International Electrotechnical and Computer Science Conference - ERK 2003*, pages 25–26. Citeseer, 2003. 37, 87

[NSA15] Fact Sheet Suite B Cryptography. https://www.keylength.com/en/6/, August 2015. [Online; accessed 01-August-2016]. 27

[OGSM16] Sébastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. Electromagnetic Fault Injection: the Curse of Flip-Flops. *Journal of Cryptographic Engineering*, pages 1–15, 2016. 51

[OGST+15] Sébastien Ordas, Ludovic Guillaume-Sage, Karim Tobich, Jean-Max Dutertre, and Philippe Maurine. Evidence of a Larger EM-Induced Fault Model. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - CARDIS 2014*, pages 245–259, Cham, 2015. Springer International Publishing. 51

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer Berlin Heidelberg, 2006. 78

[oS77] National Bureau of Standards. *Data Encryption Standard*, January 1977. 15, 16, 22, 23

[oSN01] National Institute of Standards and Technology (NIST). *Announcing the Advanced Encryption Standard (AES)*, November 2001. 77

[Pai99] Pascal Paillier. Evaluating Differential Fault Analysis of Unknown Cryptosystems. In *Public Key Cryptography: Second International Workshop on Practice and Theory in Public Key Cryptography, PKC'99 Kamakura, Japan, March 1–3, 1999 Proceedings*, pages 235–244, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. 52

[PDCK14] Rodrigo Portella Do Canto and Roman Korkikian. Hardware Encryption and Decryption Apparatus Using a N Round AES Algorithm. http://www.google.com/patents/EP2720402A1?cl=en, April 2014. EP Patent App. EP20,120,306,245. 159

[PdCKN16] Rodrigo Portella do Canto, Roman Korkikian, and David Naccache. Buying AES Design Resistance with Speed and Energy. In A. Peter Y. Ryan, David Naccache, and Jean-Jacques Quisquater, editors, *The New Codebreakers: Essays Dedicated to David Kahn on the Occasionof His 85th Birthday*, pages 134–147, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. 159

[PGQK09] Peng Peng, Deng Gaoming, Zhao Qiang, and Chen Kaiyan. EM Frequency Domain Correlation Analysis on Cipher Chips. In *1st International Conference on Information Science and Engineering - ICISE 2009*, pages 1729–1732, Dec 2009. 34

[PP09] Christof Paar and Jan Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer Science & Business Media, 2009. 25

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer Berlin Heidelberg, 2003. 54

[PSQ07] Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. *Integration, the VLSI journal*, 40(1):52–60, 2007. 34

[PSSG10] Preeti Ranjan Panda, B.V.N. Silpa, Aviral Shrivastava, and Krishnaiah Gummidipudi. *Power-Efficient System Design*. Springer, 2010. 33

[PTL+11] François Poucheret, Karim Tobich, Mathieu Lisarty, Laurent Chusseau, Bruno Robisson, and Philippe Maurine. Local and Direct EM Injection of Power Into CMOS Integrated Circuits. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2011*, pages 100–104, Sept 2011. 51

[PUB99] FIPS PUB. *Security Requirements for Cryptographic Modules*. PhD thesis, National Institute of Standards and Technology, 1999. 56

[QC82] Jean-Jaques Quisquater and Chantal Couvreur. Fast Decipherment Algorithm for RSA Public-Key Cryptosystem. *Electronics letters*, 18(21):905–907, 1982. 27

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards. In Isabelle Attali and Thomas Jensen, editors, *Smart Card Programming and Security*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer Berlin Heidelberg, 2001. 34

[RCN02] Jan M. Rabaey, Anantha P. Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits*, volume 2. Prentice hall Englewood Cliffs, 2002. 46

[Rej82] Marian Rejewski. Mathematical Solution of the Enigma Cipher. *Cryptologia*, 6(1):1–18, 1982. 14

[RO05] Christian Rechberger and Elisabeth Oswald. Practical Template Attacks. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications*, volume 3325 of *Lecture Notes in Computer Science*, pages 440–456. Springer Berlin Heidelberg, 2005. 44

[RS97] Ronald L. Rivest and Robert D. Silvermany. Are Strong Primes Needed for RSA. In *In The 1997 RSA Laboratories Seminar Series, Seminars Proceedings*. Citeseer, 1997. 25

[RS10] Mathieu Renauld and François-Xavier Standaert. Algebraic Side-Channel Attacks. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology*, volume 6151 of *Lecture Notes in Computer Science*, pages 393–410. Springer Berlin Heidelberg, 2010. 45

[RSA78] Ronald L. Rivest, Adi Shamir, and Len Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 25

[RSDT13] Cyril Roscian, Alexandre Sarafianos, Jean-Max Dutertre, and Assia Tria. Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2013*, pages 89–98, Aug 2013. 85

[SA03] Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. In Burton S. Kaliski, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer Berlin Heidelberg, 2003. 50

[SC89] George W Snedecor and Witiiam G Cochran. Statistical Methods. *Iowa State Univ*, 1989. 40

[SDB+10] Oliver Schimmel, Paul Duplys, Eberhard Boehl, Jan Hayek, Robert Bosch, and Wolfgang Rosenstiel. Correlation Power Analysis in Frequency Domain. In *Constructive Side-Channel Analysis and Secure Design - COSADE 2010*, pages 1–3, 2010. 34

[Sen13] Eugene Seneta. A Tricentenary History of the Law of Large Numbers. *Bernoulli*, 19(4):1088–1121, 2013. 35

[SGA07] Michael Sutton, Adam Greene, and Pedram Amini. *Fuzzing: Brute Force Vulnerability Discovery*. Pearson Education, 2007. 48

[SGD08] Nidhal Selmane, Sylvain Guilley, and Jean-Luc Danger. Practical Setup Time Violation Attacks on AES. In *Dependable Computing Conference, 2008. EDCC 2008. Seventh European*, pages 91–96, May 2008. 49

[Sha49] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell system technical journal*, 28(4):656–715, 1949. 15, 19

[Sha97] Adi Shamir. How to Check Modular Exponentiation. *rump session of EUROCRYPT*, 97, 1997. 90

[Sha03] Adi Shamir. Protecting smart cards from power analysis with detachable power supplies, January 14 2003. US Patent 6,507,913. 45

[Sin11] Simon Singh. *The Code Book: the Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor, 2011. 14, 24

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A Collision-Attack on AES. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 163–175. Springer Berlin Heidelberg, 2004. 38

[SLIO12] Kazuo Sakiyama, Yang Li, Mitsugu Iwamoto, and Kazuo Ohta. Information-Theoretic Approach to Optimal Differential Fault Analysis. *IEEE Transactions on Information Forensics and Security*, 7(1):109–120, Feb 2012. 53

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer Berlin Heidelberg, 2005. 37

[SMKLM02] Yen Sung-Ming, Seungjoo Kim, Seongan Lim, and Sangjae Moon. RSA Speedup with Residue Number System Immune against Hardware Fault Cryptanalysis. In Kwangjo Kim, editor, *Information Security and Cryptology - ICISC 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 397–413. Springer Berlin Heidelberg, 2002. 95

[SMWO11] Yonghee Shin, Andrew Meneely, Laurie Williams, and Jason A. Osborne. Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. *IEEE Transactions on Software Engineering*, 37(6):772–787, Nov 2011. 15

[SNK+12] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple Photonic Emission Analysis of AES. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 41–57. Springer Berlin Heidelberg, 2012. 37, 38

[SPS01] Oleg Semenov, Andrzej Pradzynski, and Manoj Sachdev. Contribution of Gate Induced Drain Leakage to Overall Leakage and Yield Loss in Digital Submicron VLSI Circuits. In *IEEE International Integrated Reliability Workshop Final Report, 2001*, pages 49–53, 2001. 34

[SWP03] Kai Schramm, Thomas Wollinger, and Christof Paar. A New Class of Collision Attacks and Its Application to DES. In Thomas Johansson, editor, *Fast Software Encryption*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer Berlin Heidelberg, 2003. 38

[TAV02] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential

Power Analysis on Smart Cards. In *Proceedings of the 28th European Solid-State Circuits Conference - ESSCIRC 2002*, pages 403–406, Sept 2002. 45

[TDF+14] Shahin Tajik, Enrico Dietz, Sven Frohmann, Jean-Pierre Seifert, Dmitry Nedospasov, Clemens Helfmeier, Christian Boit, and Helmar Dittrich. Physical Characterization of Arbiter PUFs. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 493–509. Springer Berlin Heidelberg, 2014. 35

[TK10] Elena Trichina and Roman Korkikyan. Multi Fault Laser Attacks on Protected CRT-RSA. In *Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC'2010*, pages 75–86, Aug 2010. 94, 159

[TMA+02] George Taylor, Simon Moore, Ross Anderson, Robert Mullins, and Paul Cunningham. Improving Smart Card Security Using Self-Timed Circuits. *2014 20th IEEE International Symposium on Asynchronous Circuits and Systems*, 0:211, 2002. 56

[TN98] Yuan Taur and Tak H. Ning. *Fundamentals of Modern VLSI Devices*. Cambridge University Press, New York, NY, USA, 1998. 34

[TOT+14] Sébastien Tiran, Sébastien Ordas, Yannick Teglia, Michel Agoyan, and Philippe Maurine. A Model of the Leakage in the Frequency Domain and its Application to CPA and DPA. *Journal of Cryptographic Engineering*, 4(3):197–212, 2014. 37

[TQL+12] Ming Tang, Zhenlong Qiu, Weijie Li, Shubo Liu, and Huanguo Zhang. Power Analysis Based Reverse Engineering on the Secret Round Function of Block Ciphers. In Yang Xiang, Mukaddim Pathan, Xiaohui Tao, and Hua Wang, editors, *Data and Knowledge Engineering*, volume 7696 of *Lecture Notes in Computer Science*, pages 175–188. Springer Berlin Heidelberg, 2012. 37

[TS09] Peter Tummeltshammer and Andreas Steininger. On the Role of the Power Supply as an Entry for Common Cause Faults - An Experimental Analysis. In *12th International Symposium on Design and Diagnostics of Electronic Circuits Systems - DDECS '09*, pages 152–157, April 2009. 50

[TSG03] Elena Trichina, Domenico Seta, and Lucia Germani. Simplified Adaptive Multiplicative Masking for AES. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 187–197. Springer Berlin Heidelberg, 2003. 46

[TV04] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *Proceedings of the Conference on Design, Automation and Test in Europe - DATE 2004*, DATE'04, pages 246–251, Washington, DC, USA, 2004. IEEE Computer Society. 45

[Uye92] John P. Uyemura. *Circuit Design for CMOS VLSI*. Springer, 1992. 31

[Ver58] Gilbert S. Vernam. Automatic Telegraph Switching System Plan 55-A. *American Institute of Electrical Engineers, Part I: Communication and Electronics, Transactions of the*, 77(2):239–247, May 1958. 14

[VS94] Srinivasa R. Vemuru and Norman Scheinberg. Short-Circuit Power Dissipation Estimation for CMOS Logic Gates. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 41(11):762–765, Nov 1994. 31

[VSK13] Rajesh Velegalati, Karan Shah, and Jens-Peter Kaps. Glitch Detection in Hardware Implementations on FPGAs Using Delay Based Sampling Techniques. In *2013 Euromicro Conference on Digital System Design*, pages 947–954, Sept 2013. 56

[VZ04] Bart Van Zeghbroeck. Principles of Semiconductor Devices. *Colarado University*, 2004. 34

[Wel91] Emo Welzl. Smallest Enclosing Disks (Balls and Ellipsoids). In Hermann Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes in Computer Science*, pages 359–370. Springer Berlin Heidelberg, 1991. 108, 109

[WM62] J. Torkel Wallmark and S.M. Marcus. Minimum Size and Maximum Packing Density of Nonredundant Semiconductor Devices. In *Proceedings of Institute of Radio Engineers*, volume 50, pages 286–298, 1962. 49

[WO11] Carolyn Whitnall and Elisabeth Oswald. A Fair Evaluation Framework for Comparing Side-Channel Distinguishers. *Journal of Cryptographic Engineering*, 1(2):145–160, 2011. 38

[YF13] Yuval Yarom and Katrina Falkner. Flush+ Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack. *IACR Cryptology ePrint Archive*, 2013:448, 2013. 46

[Yiu] Joseph Yiu. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. Newnes, 2013. 91

[YKLM03] Sung-Ming Yen, Seungjoo Kim, Seongan Lim, and Sang-Jae Moon. RSA Speedup With Chinese Remainder Theorem Immune Against Hardware Fault Cryptanalysis. *IEEE Transactions on Computers*, 52(4):461–472, April 2003. 57

[YWV+05] Shengqi Yang, Wayne Wolf, Narayanan Vijaykrishnan, Dimitrios Serpanos, and Yuan Xie. Power Attack Resistant Cryptosystem Design: a Dynamic Voltage and Frequency Switching Approach. In *Proceedings of Design, Automation and Test in Europe - DATE 2005*, pages 64–69 Vol. 3, March 2005. 46

[ZDT+14] Loic Zussa, Amine Dehbaoui, Karim Tobich, Jean-Max Dutertre, Maurine Maurine, Ludovic Guillaume-Sage, Jessy Clediere, and Assia Tria. Efficiency of a Glitch Detector Against Electromagnetic Fault Injection. In *Design, Automation and Test in Europe Conference and Exhibition - DATE 2014*, pages 1–6, March 2014. 56

[Zus14] Loic Zussa. *Étude des Techniques d'Injection de Fautes par Violation de Contraintes Temporelles Permettant la Cryptanalyse Physique de Circuits Sécurisés*. PhD thesis, École Nationale Supérieure des Mines de Saint-Étienne, 2014. 50

List of Main Abbreviations

2D	2 Dimensions
3D	3 Dimensions
AC	Alternating Current
ACPU	Application Central Processing Unit
ADC	Analogue to Digital Converter
AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
ARM	Acorn RISC Machine
ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
BGA	Ball Grid Array
CFA	Collision Fault Analysis
CIFA	Correlation Instantaneous Frequency Analysis
CMOS	Complementary Metal-Oxide-Semiconductor
CPA	Correlation Power Analysis
CPU	Central Processing Unit
CRP	Challenge-Response Protocols
CRT	Chinese Reminder Theorem
CSBA	Correlation Spectral-Based Analysis
DC	Direct Current
DES	Data Encryption Standard
DFA	Differential Fault Analysis
DHP	Diffie-Hellman Problem
DHT	Discrete Hilbert Transform
DIP	Dual In-line Package
DLP	Discrete Logarithm Problem

DMA	Direct Memory Access
DNA	Deoxyribonucleic Acid
DPA	Differential Power Analysis
DPD	Dynamic Power Dissipation
DRAM	Dynamic Random Access Memory
DSBA	Differential Spectral Based Analysis
DVS	Dynamic Voltage Scrambling
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromagnetism
EMD	Empirical Mode Decomposition
ERP	e-th Root Problem
FA	Fault Attack
FIPS	Federal Information Processing Standard
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GCD	Greatest Common Divisor
GCHQ	Government Communications Headquarters
GIDL	Gate-Induced Drain Leakage
HDL	Hardware Language
HHT	Hilbert Huang Transform
HWPD	Hamming Weight Probability Distribution
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IF	Instantaneous Frequency
IKE	Internet Key Exchange
IMF	Intrinsic Mode Function
IP	Intellectual Property
LDA	Linear Discriminant Analysis
LED	Light Encryption Device
LFSR	Linear Feedback Shift Register
MIA	Mutual Information Analysis
MIPS	Microprocessor without Interlocked Pipeline Stages

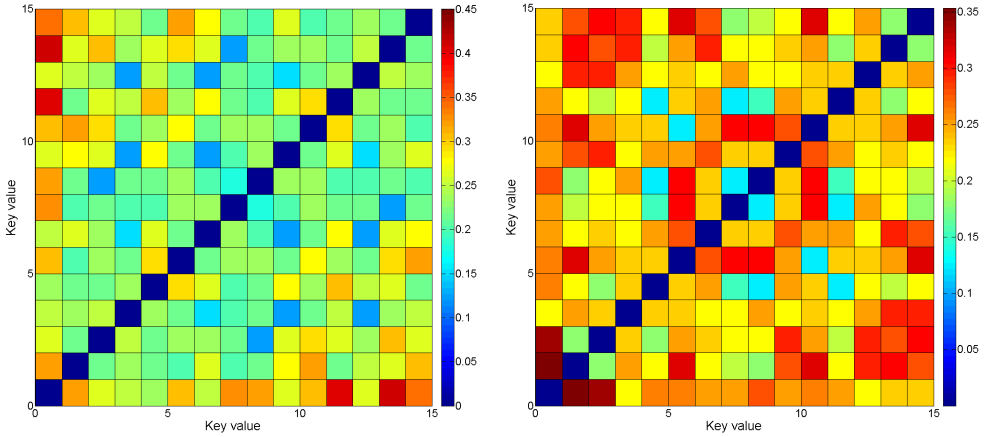
MOS	Metal-Oxide-Semiconductor
NASA	National Aeronautics and Space Administration
NSA	National Security Agency
OAEP	Optimal Asymmetric Encryption Padding
OS	Operating System
PC	Personal Computer
PCA	Principal Component Analysis
PEA	Photon Emission Analysis
PLL	Phase-Locked Loop
POI	Points Of Interests
PRNG	Pseudo Random Number Generator
PSD	Power Spectral Density
PUF	Physical Unclonable Function
RAM	Random Access Memory
RC	Resistor–Capacitor
RISC	Reduced Instruction Set Computing
RNG	Random Number Generator
RSA	Rivest Shamir Adleman
SCA	Side-Channel Analysis
SCA	Side-Channel Analysis
SDK	Starter Development Kit
SHA	Secure Hash Algorithm
SNR	Signal to Noise Ratio
SoC	System on Chip
SPD	Static Power Dissipation
SPN	Substitution Permutation Networks
SRAM	Static Random-Access Memory
TLS	Transport Layer Security
TSA	Temperature Side Channel
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
YAG	Yttrium Aluminum Garnet
ZKP	Zero-Knowledge Proofs

APPENDIX A

STATISTICAL DISTANCES FOR VARIOUS S-BOXES

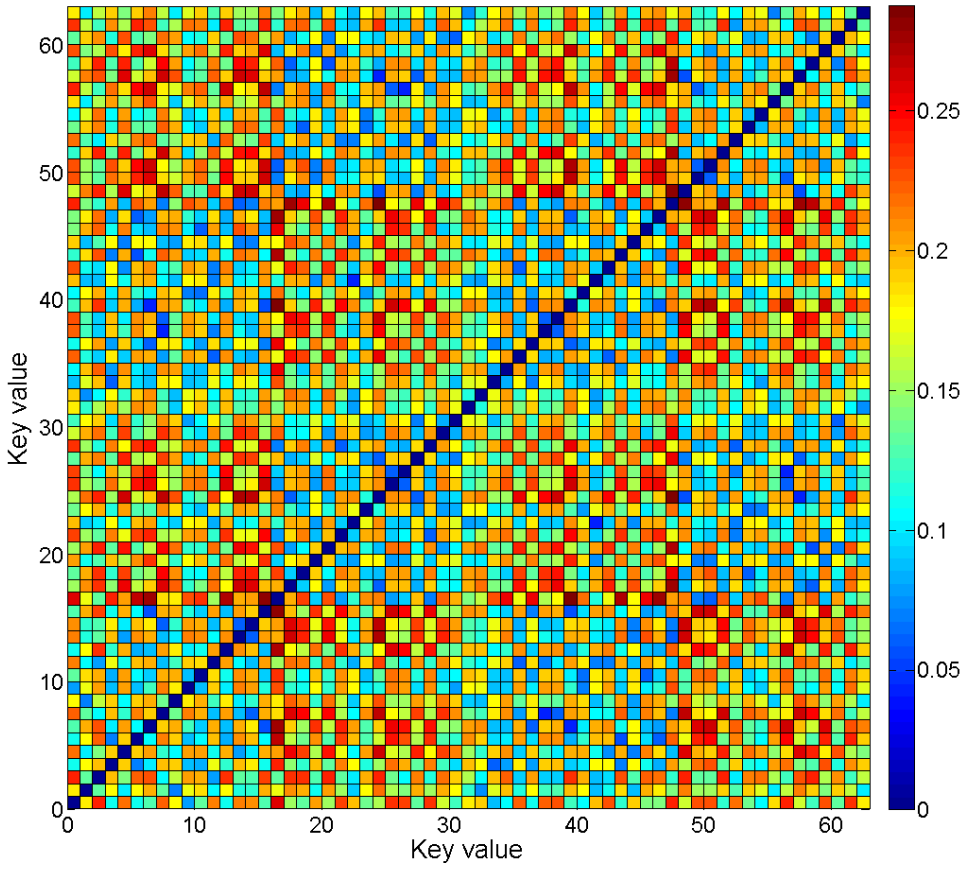
This appendix presents statistical distances computed using S-boxes of various cryptographic algorithms according to the formula:

$$\Delta(k_i, k_j) = \Delta\left(\Pr_{k_i}[\mathrm{HW}(x_{\mathrm{in}}), \mathrm{HW}(\mathbf{S}(k_i \oplus x_{\mathrm{in}}))], \Pr_{k_j}[\mathrm{HW}(x_{\mathrm{in}}), \mathrm{HW}(\mathbf{S}(k_j \oplus x_{\mathrm{in}}))]\right)$$

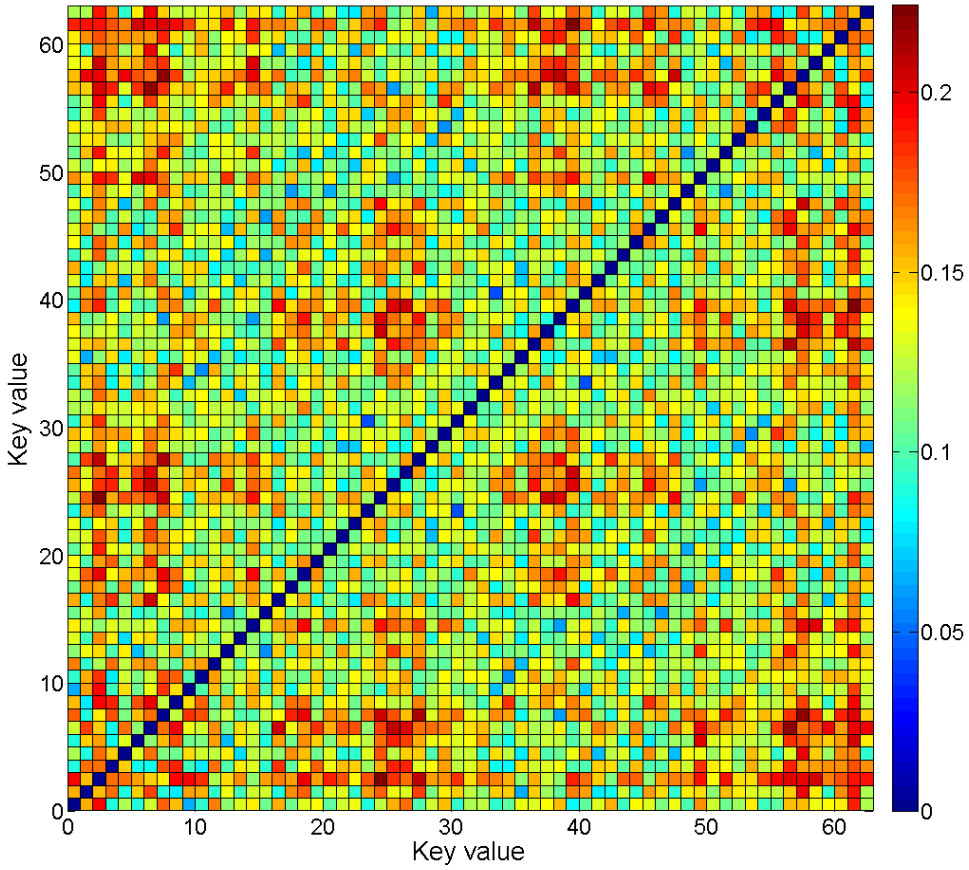


(a) LED S-box. (b) TWINE S-box.

Figure A.1 – Statistical distance for 4-to-4 LED and TWINE S-boxes.

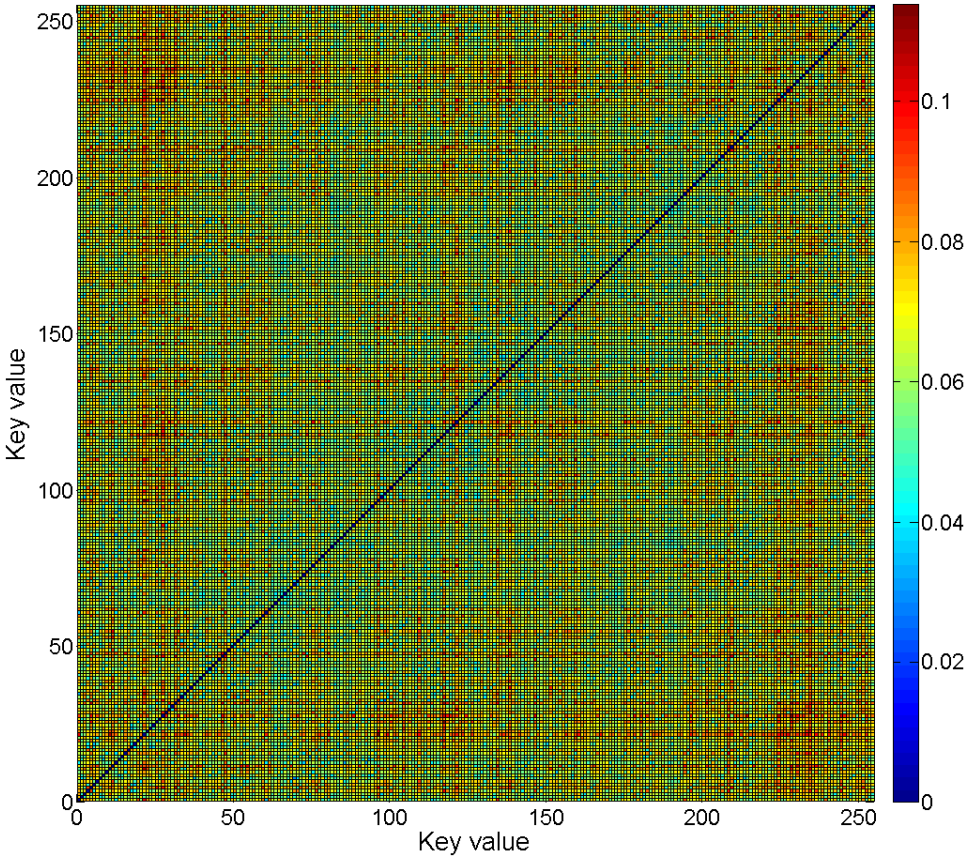


(a) First DES S-box.

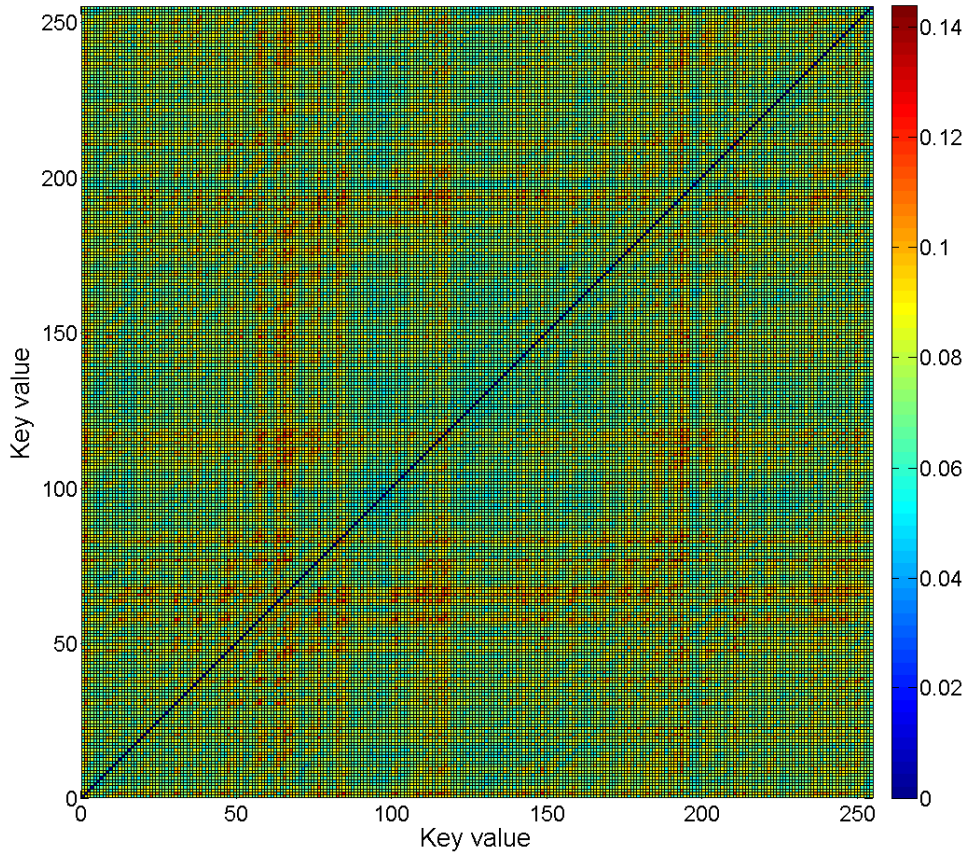


(b) Second DES S-box.

Figure A.2 – Statistical distance for 6-to-4 DES S-boxes.

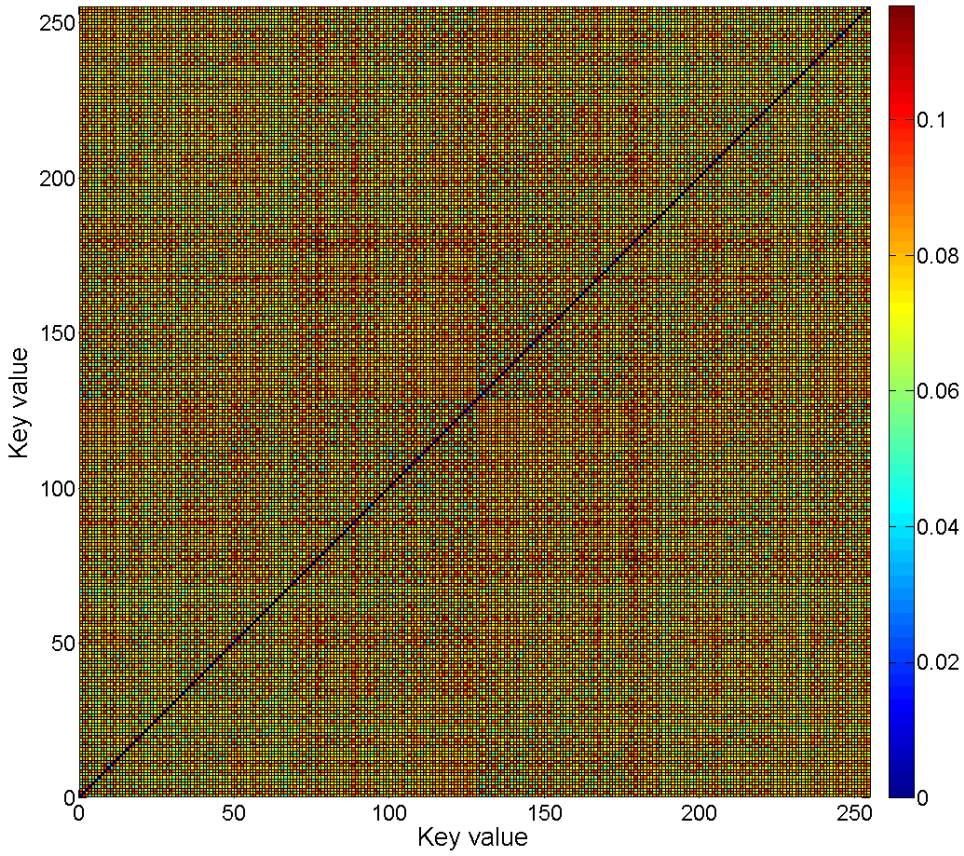


(a) AES S-box.

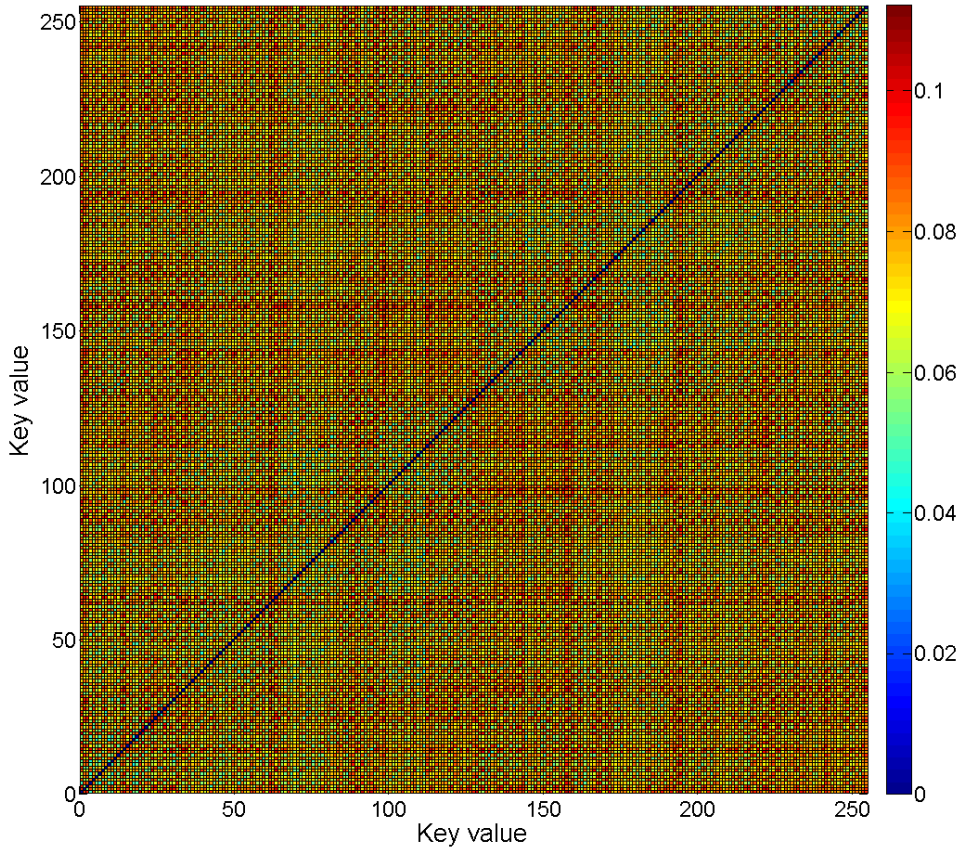


(b) Safer++ S-box.

Figure A.3 – Statistical distance for 8-to-8 AES and Safer++ S-boxes.



(a) First CAST-128 S-box.



(b) Second CAST-128 S-box.

Figure A.4 – Statistical distance for 8-to-32 CAST S-boxes.

APPENDIX B

LIST OF PUBLICATIONS

This thesis is primarily based on the peer-reviewed publications described in this section. Among the developed work are published papers, e-prints, journal and blog articles, a book article, a patent, three workshops and various program.

Multi Fault Laser Attacks on Protected CRT-RSA [TK10]

With Elena Trichina

Abstract. Since the first publication of a successful practical two-fault attack on protected CRT-RSA surprisingly little attention was given by the research community to an ensuing new challenge. The reason for it seems to be two-fold. One is that generic higher order fault attacks are very difficult to model and thus finding robust countermeasures is also difficult. Another reason may be that the published experiment was carried out on an outdated 8 bit microcontroller and thus was not perceived as a serious threat to create a sense of urgency in addressing this new menace. In this paper we describe two-fault attacks on protected CRT-RSA implementations running on an advanced 32 bit ARM Cortex M3 core. To our knowledge, this is the first practical result of two fault laser attacks on a protected cryptographic application. Considering that laser attacks are much more accurate in targeting a particular variable, the significance of our result cannot be overlooked.

Note. This work is presented in detail in Chapter 6.

Buying AES Design Resistance with Speed and Energy [PDCK14, PdCKN16]

With Rodrigo Portella do Canto, and David Naccache

Abstract. Fault and power attacks are two common ways of extracting secrets from tamper-resistant chips. Although several protections have been proposed to thwart these attacks, resistant designs usually claim significant area or speed overheads. Furthermore, circuit-level countermeasures are usually not reconfigurable at runtime. This paper exploits the AES' algorithmic features to propose low-cost and low-latency protections. We provide Verilog and FPGA implementation details. Using our design, real-life applications can be configured during runtime to meet the user's needs and the system's constraints.

Note. This article is presented in detail in Chapter 8. This work has served as basis of a patent [PDCK14] and a book chapter, volume 9100 of the series Lecture Notes in Computer Science [PdCKN16]. This work was also presented during 10^{th} AES Anniversary.

Defensive Leakage Camouflage [BQK+13]

With Eric Brier, Fortier Quentin, K. W. Magld, David Naccache, Guilherme Ozari de Almeida, Adrien Pommellet, A. H. Ragab, and Jean Vuillemin

Abstract. This paper considers the transfer of digital data over leaky and noisy communication channels. We develop defensive strategies exploiting the fact that noise prevents the attacker from accurately measuring leakage. The defense strategy described in this paper pairs each useful data element k with a

camouflage value v and simultaneously transmits both k and v over the channel. This releases an emission $e(k, v)$. We wish to select the camouflage values $v(k)$ as a function of k in a way that makes the quantities $e(k, v(k))$ as indistinguishable as possible from each other. We model the problem and show that optimal camouflage values can be computed from side-channels under very weak physical assumptions. The proposed technique is hence applicable to a wide range of readily available technologies. We propose algorithms for computing optimal camouflage values when the number of samples per trace is moderate (typically ≤ 6) and justify our models by a statistical analysis. We also provide experimental results obtained using FPGAs.

Note. This article is presented in detail in Chapter 7.

Practical Instantaneous Frequency Analysis Experiments [KNdAdC14]

With David Naccache, Guilherme Ozari de Almeida, and Rodrigo Portella do Canto

Abstract. This paper investigated the use of instantaneous frequency (IF) instead of power amplitude and power spectrum in side-channel analysis. By opposition to the constant frequency used in Fourier Transform, instantaneous frequency reflects local phase differences and allows detecting frequency variations. These variations reflect the processed binary data and are hence cryptanalytically useful. IF exploits the fact that after higher power drops more time is required to restore power back to its nominal value. Whilst our experiments reveal IF does not bring specific benefits over usual power attacks when applied to unprotected designs, IF allows to obtain much better results in the presence of amplitude modification countermeasures.

Note. This article is presented in detail in Chapter 4.

Blind Fault Attack against SPN Ciphers [KPN14]

With Sylvain Pelissier, and David Naccache

Abstract. This paper presents a novel fault attack against Substitution Permutation Networks. The main advantage of the method is an absence of necessity to know the exact cipher's input and output values. The attack relies only on the number of faulty cipher texts originated from the same unknown plaintext. The underlying model is a multiple bit-set or bit-reset faults injected several times at the same intermediate round state. This method can be applied against any round thus any round key can be extracted. The attack was shown to be efficient by simulation against several SPN block ciphers.

Note. This article is presented in detail in Chapter 5.

Résumé

Dans cette thèse nous développons et améliorons des attaques de systèmes cryptographiques. Un nouvel algorithme de décomposition de signal appelé transformation de Hilbert-Huang a été adapté pour améliorer l'efficacité des attaques par canaux auxiliaires. Cette technique permet de contrecarrer certaines contre-mesures telles que la permutation d'opérations ou l'ajout de bruit à la consommation de courant.

La seconde contribution de ce travail est l'application de certaines distributions statistiques de poids de Hamming à l'attaque d'algorithmes de chiffrement par bloc tels que AES, DES ou LED. Ces distributions sont distinctes pour chaque valeur de sous-clef permettent donc de les utiliser comme modèles intrinsèques. Les poids de Hamming peuvent être découverts par des analyses de canaux auxiliaires sans que les clairs ni les chiffrés ne soient accessibles.

Cette thèse montre que certaines contre-mesures peuvent parfois faciliter des attaques. Les contre-mesures contagieuses proposées pour RSA protègent contre les attaques par faute mais ce faisant et moyennant des calculs additionnels facilitent la découverte de la clef.

Finalement, des contre-mesures à faible complexité calculatoire sont proposées. Elles sont basées sur le masquage antagoniste, c'est-à-dire, l'exécution d'une opération d'équilibrage sur des données sensibles pour masquer la consommation de courant.

Mots Clés

Attaques par canaux auxiliaires, attaques par fautes, cryptographie, systèmes embarqués, contremesures, transformation de Hilbert-Huang, loi de probabilité pour poids de Hamming, statistiques.

Abstract

The goal of the thesis is to develop and improve methods for defeating protected cryptosystems. A new signal decomposition algorithm, called Hilbert Huang Transform, was adapted to increase the efficiency of side-channel attacks. This technique attempts to overcome hiding countermeasures, such as operation shuffling or the adding of noise to the power consumption.

The second contribution of this work is the application of specific Hamming weight distributions of block cipher algorithms, including AES, DES, and LED. These distributions are distinct for each subkey value, thus they serve as intrinsic templates. Hamming weight data can be revealed by side-channel and fault attacks without plaintext and ciphertext. Therefore these distributions can be applied against implementations where plaintext and ciphertext are inaccessible.

This thesis shows that some countermeasures serve for attacks. Certain infective RSA countermeasures should protect against single fault injection. However, additional computations facilitate key discovery.

Finally, several lightweight countermeasures are proposed. The proposed countermeasures are based on the antagonist masking, which is an operation occurring when targeting data processing, to intelligently mask the overall power consumption.

Keywords

Side-channel attacks, fault attacks, cryptography, embedded systems, countermeasures, Hilbert-Huang transform, Hamming weight probability distribution, statistics.