Secure key storage with PUFs

DOI:
10.1007/978-1-84628-984-2_16

Document status and date:
Published: 01/01/2007

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

16

Secure Key Storage with PUFs

Pim Tuyls, Geert-Jan Schrijen, Frans Willems, Tanya Ignatenko, and Boris Škorić

16.1 Introduction

Nowadays, people carry around devices (cell phones, PDAs, bank passes, etc.) that have a high value. That value is often contained in the data stored in it or lies in the services the device can grant access to (by using secret identification information stored in it). These devices often operate in hostile environments and their protection level is not adequate to deal with that situation. Bank passes and credit cards contain a magnetic stripe where identification information is stored. In the case of bank passes, a PIN is additionally required to withdraw money from an ATM (Automated Teller Machine). At various occasions, it has been shown that by placing a small coil in the reader, the magnetic information stored in the stripe can easily be copied and used to produce a *cloned* card. Together with eavesdropping the PIN (by listening to the keypad or recording it with a camera), an attacker can easily impersonate the legitimate owner of the bank pass by using the cloned card in combination with the eavesdropped PIN.

A higher level of protection is obtained when the magnetic stripe is replaced by an integrated circuit (IC) to store secret or valuable information and to carry out the critical security operations. Using state-of-the-art cryptographic techniques, a very strong solution is guaranteed as long as the IC acts as a black box.

It has, however, been shown that the black-box assumption does not fit real life very well, especially for devices (smart cards, PDAs, mobile phones) that operate in hostile environments, even for ICs. Although the use of an IC makes a device harder to attack, it turns out that there are many successful physical attacks on ICs. These attacks can be divided into three classes: non-invasive physical attacks (e.g., side channel attacks: Simple Power Analysis, Differential Power Analysis, ElectroMagnetic Analysis attacks [9,172]), invasive physical attacks (attacks that modify the physical structure: e.g., focused ion beam, etching, etc.), and fault attacks [29] (attacks that cause faults in the cryptographic operations). The fact that some of those attacks are carried out in a relatively simple way (some without requiring knowledge about implementation details) makes them very dangerous. Often, the attacker can retrieve the whole secret key from the physical attack alone. Sometimes, the attack is combined with a traditional cryptanalytic attack to reveal the whole key. In order to bridge the gap between the black-box model and real life, new models and technological components have to be developed that take into account attackers that have some access to the devices carrying out cryptographic operations. This extended set of cryptographic techniques is what we call *grey-box cryptography.*

In a first and practical approach to deal with this critical gap, the smartcard industry is working on protective measures against invasive attacks, namely protective layers and coatings that are difficult to remove. Removing the layer implies removing part of the IC, which renders the IC unusable. Furthermore, sensors are sometimes built into the IC to check for the presence of the protective layer. If removal is detected, the IC will stop functioning and hence prevent an attacker from learning its secrets. Although such coatings further increase the threshold, it turns out that, in practice, an attacker can often still successfully remove a coating (and possibly fool the sensors) and get access to the IC's interior (e.g., by using a focused ion beam (FIB)). The FIB is used to influence the (yes/no) signal that indicates the presence of the protective coating. A more secure form of protective coatings, which has the potential to protect even against these sophisticated attacks, is the *active coating* that was first introduced in [226] and further investigated in [163].

Memory encryption [304] is an important algorithmic component used to protect sensitive information. The encryption protects information from being exposed to an attacker who gets access to the memory. However, it is important to observe that a key is still needed to encrypt and decrypt that information. The problem is then reduced to the secure storage of the secret key of the memory encryption scheme.

Recently, a fundamental theoretical approach [130, 200] was developed to tackle this problem and deal with the current unsatisfactory situation. In [200], a theoretical model dubbed *physically observable cryptography* was proposed and investigated. The concept of *algorithmic tamper-proof security* was introduced in [130] to provide very strong security against physical attacks. Within this last concept, three basic components were identified to achieve this: (1) *read-proof hardware*, (2) *tamper-proof hardware*, and (3) *a self-destruction capability.* Unfortunately, no practical implementations were presented in [130].

Contributions of This Chapter

In this chapter, we focus on the development of *practical* read-proof hardware against invasive physical attacks. The word "security" has to be understood in this context.

We explain how physical unclonable functions (PUFs) can be used for this goal. In particular, we describe how secure keys are extracted from PUFs in practice. We investigate two important kinds of PUFs: coating PUFs [271] and optical PUFs [221]. Note that we do not provide a solution against side-channel attacks.

We first give some intuition for the use of (optical) PUFs for building read-proof hardware in a very simple situation. Optical PUFs consist of a 3-D physical structure that produces a speckle pattern (response) when irradiated with a laser beam. A slight change in the conditions under which they are challenged produces a completely different response. It was shown in [272] that they support a very large number of challenge response pairs (CRPs). By embedding a PUF into a credit card, the card becomes unclonable and can hence be identified with an extremely high degree of confidence.

Upon reading, the reader checks the authenticity of the card by challenging the structure with a randomly chosen challenge and verifies the obtained response versus a reference response stored in a database. Even if the attacker captures such a card and gains access to it, he cannot make a clone in a reasonable amount of time. This implies that optical PUFs represent a gray-box equivalent of other identifiers such as holograms, which fit more in the black-box model.

As a first example of secure key storage, we present coating PUFs. Coating PUFs are based on on-chip capacitive measurements of random dielectric properties of a protective layer on top of an IC [271]. Their main application is to serve as the secure storage medium on the IC from which the secret keys are extracted at the point in time when they are needed. When the coating is attacked, it is damaged to such an extent that the key can be retrieved only with great effort.

As a more sophisticated and more powerful example, we consider optical PUFs integrated with an IC (introduced in Chapter 15). From a security perspective, the main difference with coating PUFs is that optical PUFs have a much higher number of CRPs and more entropy per response (see Chapters 12 and 13). Experiments show that slight changes in the PUF material (e.g., caused by an attack) create substantially different speckle patterns and hence destroy the key. Additionally, when the response of one challenge is compromised, many other challenges with unpredictable responses remain to extract a key. These properties make optical PUFs well suited to implement a strong version of read-proof hardware.

Read-proof hardware in general, and our construction in particular, can be applied for secure key storage in smartcards, SIM (Subscriber Identity Module) cards, TPMs (trusted platform modules), DRM (digital rights management) systems, and RFID (Radio-Frequency Identification) tags [269].

16.2 PUFs for Read-Proof Hardware

In order to protect a long-term key K against physical attacks, we propose the following principles:

- Do not store long-term keys in non-volatile digital memory.
- During cryptographic operations with a long-term key in volatile memory, do not at any time let significant portions of the key reside in the volatile memory.

The motivation for these principles is as follows. Digital storage is susceptible to powerful attacks whose effectiveness is based on the strong distinguishability of physical states representing a "0" or "1" value. Instead, we propose to extract keys from a *tamper-evident* physical structure (possibly integrated with the device) such that the key is only temporarily present in the device (only at the point in time when used). PUFs are a natural candidate for these physical sources. Furthermore, there are powerful attacks that "freeze" volatile (RAM) memory. We assume that such an attack critically damages a PUF, but it still allows the attacker to get information about the key that is present in the RAM at the time of attack. In order to reduce this information, the RAM should not contain a significant part of K at any time. The attack cannot be repeated (since the PUF is effectively destroyed), and, hence, most of the key then remains unknown to the attacker.

16.2.1 Attack Model for Read-Proof Hardware

To put the discussion on a more formal footing, we describe in detail the attacks against which the read-proof hardware must be resistant. An attack is considered to be successful if the attacker obtains sufficient information to reconstruct the key K. We consider a device containing a long-term storage medium (the physical structure), means for extracting responses from this medium (sensors), a processing unit, separate long-term memory for storing the instructions that run on the processor, and volatile memory in which cryptographic operations are performed with K. The attacker is allowed to attack any of these. It is assumed that the method of challenging the physical structure is known to him, as well as the precise challenges used to extract K. We consider the following attacks:

1. **Non-invasive inspection of the volatile memory, processor, or sensors.** These methods do not damage any part of the device. Examples are optical imaging and side-channel attacks.
2. **Invasive inspection of the volatile memory, processor, or sensors.** These methods physically modify/damage at least one part of the device. Examples are etching and FIBs.
3. **Detachment of the long-term storage medium.** The attacker detaches the physical structure from the rest of the device without

damaging it. Then he challenges it (using his own measurement device) to obtain K from the measured responses.

4. **Non-invasive physical attack on the long-term storage medium.** The attacker first subjects the physical structure to non-destructive scrutiny (e.g., by microscopy or X-ray imaging). Then he uses the obtained information to get K in one of the following ways:
   (a) He makes a physical clone of the structure and challenges it (using his own measurement device) to obtain K from the measured responses.
   (b) He mathematically models the physical structure. He computes the responses to the appropriate challenges to obtain K.
5. **Invasive physical attack on the long-term storage medium.** The attacker first subjects the physical structure to a destructive analysis (e.g., by etching, drilling, or sawing). Then he uses the obtained information to get K in one of the following ways:
   (a) He makes a physical clone of the structure and challenges it (using his own measurement device) to obtain K from the measured responses.
   (b) He mathematically models the physical structure. He computes the responses to the appropriate challenges to obtain K.
6. **Code modification.** The attacker modifies the algorithms stored in the device such that K is revealed.

16.2.2 Requirements

In order to be resistant against all of the above-mentioned attacks, the hardware has to meet the following requirements:

1. The physical structure must be bound inseparably to the rest of the hardware. Any attempt to separate it should result in significant damage to the structure. (This gives resistance against attack 3.)
2. "Inscrutability": Measurements (both destructive and non-destructive) must not reveal accurate information about the composition of the physical structure. (This protects against the data acquisition phase of attacks 4 and 5.)
3. The physical structure has to be opaque and it has to cover the sensors, the processor, and the volatile memory. (This gives resistance against attack 1.)
4. The structure has to be *unclonable*. This is defined as follows:
   (a) *Physical unclonability.* It should be hard to make a physical copy that has a similar challenge-response behavior as the original structure, even given accurate knowledge of the structure's composition.
   (b) *Mathematical unclonability.* It should be hard to construct a mathematical model that has a non-negligible probability of correctly predicting responses, even given accurate knowledge of the structure's composition.

   (This property protects against the cloning phase of attacks 4 and 5.)

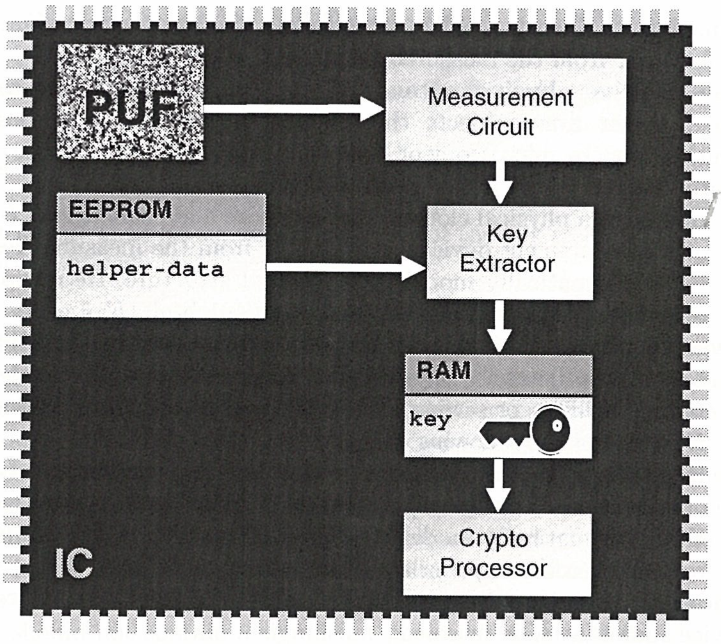

Fig. 16.1. Schematic layout of an IC with a PUF for secure key storage.

5. The physical structure has to be tamper-evident. Physical damage should significantly[1] change the challenge-response behavior. (This protects against attack 2 and adds resistance against the data acquisition phase of attack 5.)
6. The hardware must contain tamper-proof memory that can be read but not modified. In this memory, public data are stored (e.g., algorithms and public keys) that are not secret but whose integrity must be guaranteed. (This prevents attack 6.)

Finally, in order to be practically feasible, it must be easy to challenge the structure and measure its response. Preferably, the structure is inexpensive and easy to integrate in a device.

Given the requirements stated above, PUFs form a natural candidate technology as a basis for a secure key storage device. There are two main ways to use them for this purpose.

1. The long-term key K is extracted from one or more PUF responses by means of a helper-data algorithm (fuzzy extractor).
2. The long-term key K is stored in encrypted form, $E_{K_i}(K)$, in some non-secure memory. The short-term keys K_i used for encrypting the long-term key K are extracted from the PUF.

[1] This statement is made more precise in Section 16.6.3.

The second option has several advantages: (1) When the PUF has many challenges, the short-term keys are different every time they are generated. This provides protection against side-channel attacks on the PUF readout. (2) Additional protection against side-channel attacks is achieved when the key K is considered as a long-term secret on which several secret keys in later stages are based. These keys are derived by using so-called exposure-resilient functions [52].

A schematic overview of an IC with PUF-based secure key storage is shown in Fig. 16.1.

16.3 Cryptographic Preliminaries

We propose to extract a key from physical sources (PUFs) present on an IC. Since measurements on a physical structure are inherently noisy, the responses of such a structure cannot be directly used as a key. An additional problem is that these responses are not uniformly random. In order to guarantee security, the extracted key should have entropy equal to its length.

This implies that we need a helper-data algorithm/fuzzy extractor [91,182] for reconstruction of the keys. A helper-data scheme consists of a pair of algorithms (G, W) and two phases: an *enrollment* and a *reconstruction* phase. We will use the following notation: x denotes the measurement value of a response during the enrollment phase and y denotes the corresponding value during the reconstruction phase. During enrollment, the key K is created for the first time. The helper-data algorithm $W(\cdot, \cdot)$ is used during the enrollment phase and creates the helper data w based on the measurement value x during enrollment and the randomly chosen key K. The algorithm $G(\cdot, \cdot)$ is used during the key reconstruction phase for reconstruction of the key K as follows: $K = G(w, y)$. It was proven in [278] that this approach is equivalent to a fuzzy extractor.

As a second primitive, we need a standard digital signature scheme (SS): is $(\mathrm{SK}_g, \mathrm{Sign}, V)$, where SK_g is the secret-key generation algorithm, Sign is the signing algorithm, and V is the verification algorithm. The enroller runs SK_g and obtains a secret-public key pair (sk, pk). This is a one-time action. The public key pk is hard-wired in each IC (in tamper-proof memory). With the private key sk, the enroller signs the helper data w and $P(K)$ (where P is a one-way function). The signatures $\sigma(w)$ and $\sigma(P(K))$ are then stored in the EEPROM (Electrically Erasable Programmable ROM) of the IC together with the helper data w.[2]

[2] Instead of storing $\sigma(P(K))$, it is more secure to store $\sigma(P(K), \tilde{x})$, where $\tilde{x}$ is additional unpredictable key material that is obtained from the PUF (if necessary, derived from the response of a second challenge). We have chosen not to include this in the notation for the sake of transparency.

16.3.1 Procedure for Generation and Reconstruction

Creation and reconstruction of the key K is done as follows. First, the global statistical properties (noise level, etc.) of the behavior of the physical structure are determined. In particular, the entropy of the output of the physical structure is estimated and the secrecy capacity $\mathbf{I}(X;Y)$ (mutual information) is estimated of the channel describing the noisy observation.[3] This can be done using the methods described in [150]. These parameters determine the choice of the key length k and of an appropriate helper-data algorithm (G, W).

Enrollment

This phase consists of two steps:

1. Generation of a key $K \in \{0,1\}^k$ and helper data w by $w \leftarrow W(x, K)$.
2. The IC interprets K as a private key and generates the corresponding public key $P(K)$. Then the IC outputs $(w, P(K))$. The enroller signs these data and stores the signatures $\sigma(w)$ and $\sigma(P(K))$ in the IC's EEPROM.[4]

Reconstruction

The IC performs the following steps:

1. It retrieves $w, \sigma(w)$ from EEPROM and checks the signature $\sigma(w)$ by running V on w and $\sigma(w)$ using the public key pk. If the signature is not OK, the IC shuts down permanently. Otherwise, it continues.
2. The IC challenges its physical structure and obtains the measurement value y (note that, typically, $y \neq x$ due to noise).
3. The data w and y are processed by the helper-data algorithm G. This yields the key $K' \leftarrow G(w, y)$.
4. The IC computes $P(K')$. Then it runs V on $P(K')$ and $\sigma(P(K))$ using the public key pk. If the signature is OK, the IC proceeds and K can be used as a private key. Otherwise, the IC shuts down permanently.

16.3.2 The Continuous Case

Helper data and fuzzy extractors have been extensively studied in the discrete case, but only few papers have studied the continuous situation [182]. Since, in reality, PUF responses are often continuous data rather than discrete data, it is worthwhile to spend a few words on techniques for the continuous case.

[3] This is a one-time event that is performed during a pre-processing step.

[4] Alternatively, K is used as a symmetric key. The IC outputs K and the enroller stores $\sigma(P(K))$ in the EEPROM. The circuit that outputs K is destroyed after this procedure.

For discrete PUF responses $\mathbf{X} \in \{0,1\}^n$, a fuzzy extractor performs two steps:

- Information reconciliation, which is basically an error-correction step;
- Privacy amplification, which guarantees that the extracted string is uniformly random.

When the PUF response is continuous (i.e., the PUF response $\mathbf{X}$ is a random variable on $\mathbb{R}^n$), the situation is slightly different. Since keys are binary strings, a discretization (quantization) step has to be performed. We identify the following two steps:

- *Fuzzy discretization*: During this step, the continuous variable $\mathbf{X}$ is turned into a binary string $\mathbf{S}$ such that (1) when a noisy version $\mathbf{Y}$ of $\mathbf{X}$ is measured, a string $\mathbf{S}'$ is obtained that lies close to $\mathbf{S}$ according to some distance measure and (2) the string $\mathbf{S}$ is uniformly random distributed.
- *A discrete fuzzy extractor/helper data algorithm* is applied to the (noisy) string $\mathbf{S}$ in order to remove the noise and to guarantee the randomness of the extracted key K.

The main difference between the continuous case and the discrete case is the fact that in order to perform the *fuzzy discretization* step, the probability distribution has to be known. It turns out that by performing a sufficient amount of measurements on the PUF, this distribution can be determined in practice. Below, we illustrate how a fuzzy discretization is performed in some concrete cases such as optical and coating PUFs.

16.4 Secure Key Storage with Optical PUFs

In this section we describe some techniques for using optical PUFs as a source of large amounts of secret-key material. Details on the physical structure of optical PUFs can be found in Chapter 15.

An optical PUF consists of a physical structure containing many random, uncontrollable components. Typically, one can think of a piece of glass containing randomly distributed light-scattering particles. When irradiated with a laser beam, they produce a speckle pattern (see Fig. 16.2 for an example). This speckle pattern can be considered as the unique fingerprint of the structure. We distinguish between *bare* optical PUFs and *integrated* ones. Bare optical PUFs are simply referred to as optical PUFs and consist of the physical structure only. Integrated optical PUFs consist of a physical structure integrated with the laser and reading device, and all are optionally integrated into an IC. Note that an integrated PUF is not necessarily a controlled PUF (see Chapter 14). As usual with physical systems, the responses are not exactly equal when the system is measured several times, even when challenged under seemingly identical circumstances. This is what we call the robustness or noise problem.

Robustness

We briefly describe some noise sources for optical PUFs.

1. **Interdevice variation:** For bare PUFs, the external reader that challenges the PUF and detects the response during the verification phase is typically a different device than the one that was used during the enrollment phase. Alignment and sensitivity differences between readers give rise to noise, unless great pains are taken to enforce very small mechanical and/or electrical tolerances. However, the potential number of readers is enormous, making such a standardization impractical and expensive.
2. **Physical environment:** Even repeated measurements with the same challenging and detection device do not give identical results. Time-dependent external influences like temperature, moisture, vibrations, stray light, stray fields, and so forth can have an impact on the measurements.
3. **Interaction with environment:** The PUF itself is not immutable. It can accidentally get damaged. Another problem is spontaneous degradation. Most materials slowly change over time due to chemical reactions, friction, and repeated thermal deformations. The rate of drifting determines the lifetime of the key material in the PUF.

In order to get a sufficient level of robustness one can use several methods (which are best combined): (a) **physical noise reduction:** reducing the noise at the source; (b) **algorithmic noise correction**: given a certain level of noise, extracting as much robust key material as possible by properly choosing a fuzzy discretizer and fuzzy extractor. In the remainder of this chapter we discuss algorithmic countermeasures. Hardware countermeasures have been described in [287].

16.5 Key Extraction from Speckle Patterns

In this section, we present a concrete implementation of a fuzzy discretizer and a fuzzy extractor for speckle patterns.

16.5.1 Pre-processing

In order to turn the speckle pattern into a compact binary representation, it is first filtered using a 2-D Gabor transform, which is defined as follows [287]. The basis function $\Gamma(s, \mathbf{k}, \mathbf{x}_0, \mathbf{x})$ is the product of a plane wave with wave vector $\mathbf{k}$ and a Gaussian with width s centered on $\mathbf{x}_0$. Here, $\mathbf{x}$ denotes a location in the speckle image. The Gabor basis functions Γ and the Gabor coefficients G are given by

$$G(s, \mathbf{k}, \mathbf{x}_0) = \int d^2x \;\; \Gamma(s, \mathbf{k}, \mathbf{x}_0, \mathbf{x}) I(\mathbf{x}) \tag{16.1}$$

$$\Gamma(s, \mathbf{k}, \mathbf{x}_0, \mathbf{x}) = \frac{1}{s\sqrt{2\pi}} \sin[\mathbf{k} \cdot (\mathbf{x} - \mathbf{x}_0)] \; \exp\left[-\frac{(\mathbf{x} - \mathbf{x}_0)^2}{4s^2}\right] . \tag{16.2}$$

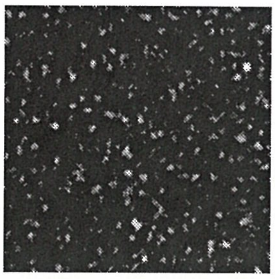
Fig. 16.2. Speckle pattern, 512x512 pixels.

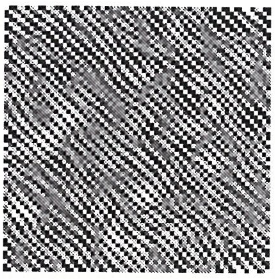
Fig. 16.3. 45° Gabor Coefficients of Fig. 16.2 after subsampling, 64x64 pixels.

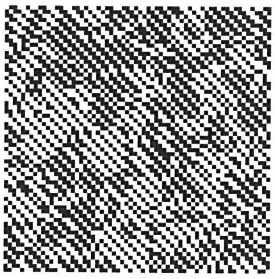
Fig. 16.4. Binarization of Fig. 16.3.

$I(\mathbf{x})$ denotes the light intensity at location $\mathbf{x}$. We have selected the imaginary part of the transform since it is invariant under spatially constant shifts of I. In our experiments, the following parameters have been used for the Gabor transform: A single Gaussian width $s = 18$ pixels; a single length $|\mathbf{k}| = \pi/(8 \text{ pixels})$; the direction of $\mathbf{k}$ is 45° with respect to the horizontal axis; $\mathbf{x}_0$ are positions in a square grid with a spacing of 8 pixels. This yields a Gabor image as depicted in Fig. 16.3, consisting of 4096 Gabor coefficients.

In order to turn this analog representation into a binary representation, the Gabor coefficients are binarized by quantizing the values into two quantization intervals: Positive values of G are mapped to 1 and negative values to are mapped to 0. The resulting image is depicted in Fig. 16.4.

16.5.2 Fuzzy Discretiser: The Concept of Robust Components

Here, we briefly describe the general idea of how to get a noise-robust vector from a speckle pattern. First we introduce some notation. For a vector $\mathbf{V} \in \mathbb{R}^n$ and a set $A \subset \{1, \ldots, n\}$, we denote by $\mathbf{V}_A$ the restriction of $\mathbf{V}$ to its components indexed by the set A. Based on the nature of a speckle pattern, we model the Gabor coefficients on the $\mathbf{x}_0$ grid as a real-valued vector $\mathbf{g} \in \mathbb{R}^n$. The binarized coefficients are denoted as a vector $\mathbf{X} \in \{0,1\}^n$. The enrollment measurement is denoted by $\mathbf{X}$ and the later measurements are denoted by $\mathbf{Y}$, which are noisy versions of $\mathbf{X}$.

We introduce the notion of *robust components.* Loosely speaking, a component i of $\mathbf{X}$ is called *robust* if the probability of a bit flip in that component is low. More precisely, we define this as follows:

Definition 16.1. *Let $m, \epsilon > 0$ and let $\mathbf{g}^1, \ldots, \mathbf{g}^m \in \mathbb{R}^n$ be a sequence of vectors. Then, for $i \in \{1, \ldots, n\}$, we say that the ith component is (ϵ, m)-robust if and only if*

$$\left| \frac{1}{m} \sum_{j=1}^{m} \operatorname{sgn}(g_i^j) \right| \geq 1 - \epsilon.$$

The sequence $\mathbf{g}^1, \ldots, \mathbf{g}^m$ has to be interpreted as a sequence of measurement results (analog Gabor coefficients) performed during enrollment.

The fuzzy discretizer performs two steps. First, robust components are selected from $\mathbf{X}$ and their positions are stored in a first set of helper data w_1. Since the components of $\mathbf{X}_{w_1}$ are not uniformly random distributed, a further (random) subselection of components is made in such a way that this subselection *is* uniformly distributed. The positions of the remaining bits are stored in helper data $w_2 \subset w_1$.

Several methods have been developed to determine the robust components. Below, we present two practical schemes. The first scheme (Section 16.5.3) selects the analog Gabor coefficients with the highest absolute value. In the second scheme (Section 16.5.4), the selection is based on an estimate of the probability distribution of the binarized Gabor coefficients.

16.5.3 Robust Components: Scheme 1

Robust components are selected using soft-decision information available before binarization. They are selected as the m components i with the largest corresponding absolute Gabor value $|g_i|$ (in our experiments, m was typically 511). It is intuitively clear that this is consistent with Definition 16.1. In Fig. 16.5 the selected robust bits are depicted as black (0) and white (1) pixels, whereas the non-robust bits are gray. The locations of these black and white pixels are stored in helper data w_1. By restricting $\mathbf{X}$ to its robust components, we obtain a binary string $\mathbf{S} = \mathbf{X}_{w_1} \in \{0,1\}^m$.

16.5.4 Robust Components: Scheme 2

In this scheme, the robust components are selected by predicting the value of a second enrollment image $\mathbf{X}^2$ given the values of neighboring pixels in a first enrollment image $\mathbf{X}^1$. The pixels whose values can be well predicted are defined as the robust ones. A well-known algorithm to construct a probabilistic model that allows to make such predictions is the context-tree weighting (CTW) compression algorithm [293].

Let $\mathbf{X}^1, \mathbf{X}^2 \in \{0,1\}^n$ be two observations of the same (ergodic) source over a noisy channel. Let A be a finite subset of $\{0, \ldots, n-1\}$. We use the CTW

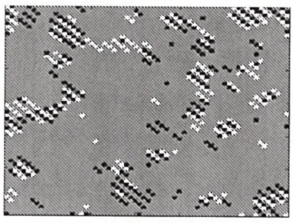

Fig. 16.5. Robust bits from large Gabor coefficients. Black = 0, white = 1, gray = non-robust.

algorithm [293] to estimate the model statistics $\mathbb{P}[X_t^2 = 1|\{X_{t-a}^1, a \in A\}]$ for $t \in \{0, \ldots, n-1\}$. Typically, the set A is defined as a number of positions in the neighborhood of the position t.

We define ϵ-robust components as those positions t for which, according to the estimated model, $\mathbb{P}[X_t^2 = 1|\{X_{t-a}^1, a \in A\}] \leq \epsilon$ or $\mathbb{P}[X_t^2 = 1|\{X_{t-a}^1, a \in A\}] \geq 1 - \epsilon$, where $\epsilon < 0.5$ is a positive constant. The positions t (in the enrollment image) that satisfy this criterion are stored in helper data $w_1 \subset \{0, \ldots, n-1\}$ and the string $\mathbf{S}$ is defined as $\mathbf{S} = \mathbf{X}_{w_1}^1$.

We apply this method to speckle patterns. During enrollment, we start from four (as an example) pre-processed speckle patterns $\mathbf{X}^1, \ldots, \mathbf{X}^4$. Then the CTW algorithm is used to estimate the model statistics $\mathbb{P}[X_t^2 = 1|\{X_{t-a}^1, a \in A\}]$. In order to get better estimates, based on more data, the CTW statistics are updated two more times. First, it is updated with probabilities $\mathbb{P}[X_t^3 = 1|\{X_{t-a}^1, a \in A\}]$ and then with probabilities $\mathbb{P}[X_t^4 = 1|\{X_{t-a}^1, a \in A\}]$. The set A consists of the local pixel (t) itself and the eight surrounding pixels.

This statistical model obtained with this procedure is employed to identify which bits are highly predictable (i.e., robust). An example is shown in Figs. 16.6 and 16.7 for $\epsilon = 0.1$. The locations of the robust pixels are stored as helper data $w_1 \subset \{0, \ldots, n-1\}$, and the string $\mathbf{S}$ is defined as $\mathbf{S} = X_{w_1}^1$.

16.5.5 Fuzzy Discretizer: Randomness Extractor

The robust bits $\mathbf{S}$, specified in the set W_1, are not uniformly random distributed. In other words, these bits do not have full entropy. Intuitively, this can be seen from the fact that the bits appear in diagonal stripes of equal value in the Gabor transformed images.[5] This is furthermore confirmed by compressing $\mathbf{S}$

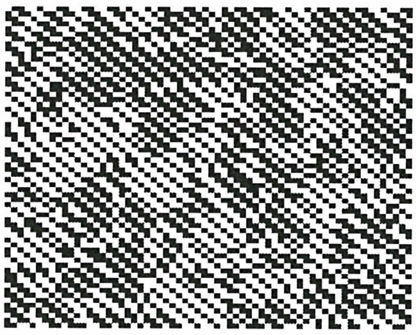
Fig. 16.6. Binarized Gabor image.

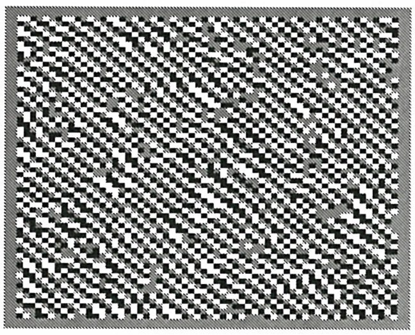
Fig. 16.7. Robust pixel locations (black = 0, white = 1). Non-robust pixels are depicted in gray.

[5] Note that the direction of the sequences is related to the 45° direction in which the Gabor transform was applied [287].

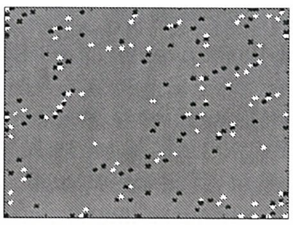

Fig. 16.8. Decimated robust bits.

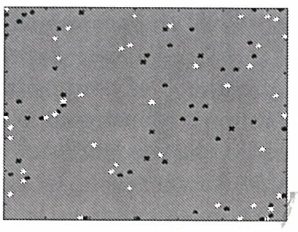

Fig. 16.9. Random sub-selection of decimated robust bits.

with the CTW universal source-coding algorithm [150,293], using neighboring pixels from Fig. 16.5 as context. With this method, the $m = 511$ bits can be compressed into 138 bits, a compression ration of 27%. Although this is merely an upper bound for the compression ratio, it clearly shows that the 511 bits certainly do not have full entropy. This is due to the fact that there are still strong correlations between the pixels. In order to remove the correlations in $\mathbf{S}$, the following steps are performed during enrollment:

- $\mathbf{S}$ is *decimated*: For every sequence of correlated bits (i.e., a sequence that forms a stripe), a single bit is randomly chosen. This process is called *decimating*. An example is shown in Fig. 16.8. From the original $m = 511$ robust bits, only 190 are left. (Using the CTW method, it turns out that these 190 bits can be compressed into 179 bits (94%), which shows that the decimated bits still do not have full entropy.)
- From this selection of bits, a further random subselection is made. Figure 16.9 shows an example: Fifty percent of the bits of Fig. 16.8 are randomly selected; only 95 bits are left.
- The resulting string should have entropy equal to its length, given all side information that is available (e.g., helper data and the knowledge that one has about the original measurement data). We verify this randomness property by running the CTW compression algorithm and checking whether the string is properly incompressible. When that is the case, the positions of those remaining bits are stored in the helper dataset $w_2 \subset w_1$. In the example case given above, the string turns out to be incompressible and, hence, the remaining 95 bit positions are stored in dataset w_2.

We note that the rate of the above scheme is equal to 2.5%, which is rather low when compared with the secrecy capacity of such a channel that has been reported to be equal to 25% in [150].

16.5.6 Discrete Fuzzy Extractor

In order to guarantee that no errors remain in the final key K, a discrete fuzzy extractor is applied in the final step to the string $\mathbf{X}_{w_2}$. Since the string $\mathbf{X}_{w_2}$ is uniformly random distributed by construction, this can be done by the so-called *code offset* approach. This technique is described in Chapter 4. The fuzzy extractor yields the final key K. It is guaranteed to be close to random and no longer contains noise.

16.5.7 Two-Way Use of Helper Data

In all schemes discussed so far, helper data are generated during enrollment and applied at the time of key reconstruction. However, the measuring device is capable of producing helper data also in the verification/key reconstruction phase. Instead of discarding this extra information, one can use it to improve the robustness of the extracted keys. We present an interactive protocol between a reader and a verifier. The robust components obtained from enrolment and verification are combined using an "AND" operation.

- **Enrollment:** The Verifier subjects the PUF to a challenge C and converts the analog response R to a bitstring $\mathbf{X}$. He determines robust components and constructs the helper dataset w of pointers to the robust parts of $\mathbf{X}$. He stores $(\mathrm{ID_{PUF}}, C, w, \mathbf{X})$.
- **Key reconstruction:** The PUF is inserted into the reader and the reader sends $\mathrm{ID_{PUF}}$ to the Verifier. The Verifier sends C and w. The reader challenges the PUF with C and measures a response R', which it converts into a bitstring $\mathbf{X}'$. It determines the robust components of X' and constructs new helper data w'. It sends w' to the Verifier. Both the reader and the Verifier now compute the *combined helper data* $W = w \cap w'$. The Verifier computes $\mathbf{S} = \mathbf{X}_W$, and the reader computes $\mathbf{S}' = \mathbf{X}'_W$. Finally, $\mathbf{S}$ and $\mathbf{S}'$ are used for the construction of a secret key (e.g., using a fuzzy extractor).

An analysis of error probabilities and key lengths was presented in [287]. It was shown there that the bit error probability in $\mathbf{S}$ is drastically improved, compared to the "one-way" case, where only the enrolled helper data are used ($\mathbf{S}_{1\mathrm{way}} = \mathbf{X}_w$; $\mathbf{S}'_{1\mathrm{way}} = \mathbf{X}'_w$). As a consequence, the amount of computational effort spent on the error correction is greatly reduced. Furthermore, it turns out that the extracted keys are longer because fewer redundancy bits are needed. For a reasonable choice of parameters, the improvement in bit error probability in S' can be as small as a factor 5 and as large as 50. The simultaneous improvement in key length varies between 20% and 70%. The difference between the two methods is most pronounced when the measurements are very noisy.

16.6 Secure Key Storage with Coating PUFs

In this section we discuss secure storage of keys in ICs using coating PUFs. First, we briefly list the properties of the coating. (See Chapter 15 for more details on the hardware.) Then we give a fuzzy discretization algorithm for obtaining uniformly distributed bitstrings from the capacitance measurements. Finally, we present experimental results, including an analysis of the resistance against FIB attacks.

16.6.1 Coating PUF Properties

A coating PUF consists of a coating with random dielectric particles that is deposited on top of the IC. The top metal layer of the IC contains sensors that locally measure the capacitance values of the coating. The chemical composition of the coating gives it a number of favorable properties. (1) The coating is opaque, absorbing light from the infrared part of the spectrum to the ultraviolet. (2) The conductive particles in the coating shield off electromagnetic emissions of the IC. (3) The coating is mechanically tough and very strongly bound to the IC. (4) It is resistant against chemical substances.

Furthermore, inspection of the coating PUF from the outside is difficult. Measuring from the outside gives different capacitance results than from the inside, since the on-chip measurements are very sensitive to the precise locations of the dielectric particles.

The information content of a coating is much lower than that of an optical PUF. The theoretical value is 6.6 bits of entropy per ($120\,\mu\text{m}^2$) sensor (see Chapter 12). In practice (Section 16.6.3), we extract less than 4 bits per sensor. However, this is sufficient entropy for key storage in a controlled PUF.

16.6.2 Fuzzy Discretization

In this section we describe the algorithmic part of our architecture for coating PUFs. Since a coating PUF consists of n sensors that measure coating properties independently, we model the responses of a coating PUF as a continuous vector $\mathbf{X} \in \mathbb{R}^n$ of independent identically distributed (i.i.d.) random variables with probability distribution ρ. Experiments have shown that the i.i.d. property is well satisfied and that ρ is well approximated by a Gaussian distribution.

We describe the discretisation algorithm for a single component $X \in \mathbb{R}$. The generalization to all components is straightforward. In order to extract a binary string $S \in \{0,1\}^l$ from X, the following steps are performed during enrollment:

- Choose an equiprobable partition of $\mathbb{R}$, $\mathcal{A} = \{A_0, \ldots, A_{2^l-1}\}$, such that $\mathbb{P}[X \in A_0] = \mathbb{P}[X \in A_1] = \cdots = \mathbb{P}[X \in A_{2^l-1}] = 1/2^l$. Subdivide this partition into equiprobable "left" and "right" parts: $A_i = A_i^{\text{left}} \cup A_i^{\text{right}}$, with $\mathbb{P}[X \in A_i^{\text{left}}] = \mathbb{P}[X \in A_i^{\text{right}}] = 2^{-l-1}$. (The resulting partition is denoted as $\mathcal{A}_{\text{sub}}$.)
- Give S a value according to the interval $i(x)$ in $\mathcal{A}$ to which x belongs; for example, if $l = 3$ and $x \in A_i$ then S is a 3-bit representation of the integer i.

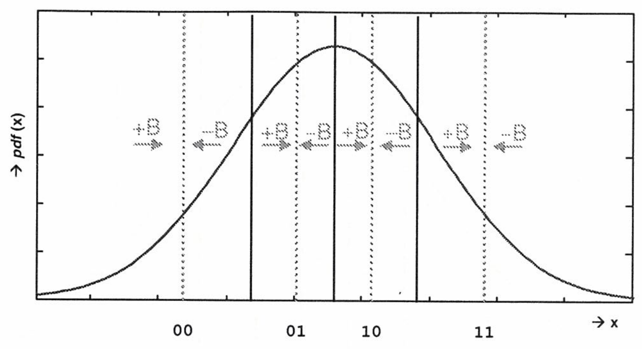

Fig. 16.10. The equiprobable partition $\mathcal{A}_{\text{sub}}$ for $l = 2$ and a Gaussian distribution. To each interval A_i (separated by solid lines), a 2-bit code is assigned. The helper data are shown as well.

- Construct helper data w as follows. If $x \in A_i^{\text{left}}$ for some i, then set $w = +1$. Otherwise set $w = -1$.
- Create a lookup table (LUT) B. The purpose of this LUT is to provide noise resistance during key reconstruction. The LUT contains "step sizes" B for pushing x toward the middle of its interval A_i. In the most general case, B is a function of x. However, since this could require a large LUT, B can also be defined as a function of the interval index i, or, even simpler, B can be a constant.
- Store the partition $\mathcal{A}_{\text{sub}}$, the helper data w and the LUT B in tamper-proof memory.

An example of the partition $\mathcal{A}_{\text{sub}}$ is shown in Fig. 16.10. During key reconstruction, a noisy version X' of X is measured. Then the following steps are performed to extract a bit string S':

- Read $\mathcal{A}_{\text{sub}}$, w, and B from tamper-proof memory.
- Compute $q = x' + w \cdot B(x')$.
- Determine the interval index $i' \in \{0, \dots, 2^l - 1\}$ such that $q \in A_{i'}$.
- Set S' to the l-bit encoding of i'.

Note that the bitstring S is uniformly distributed by construction. Also note that the partition $\mathcal{A}$, the LUT B, and the helper data w are public. However, these reveal no information on S, since the probability $\mathbb{P}[X \in A_i|w]$ is equal for all i.

Due to the measurement noise, it can happen that $S' \neq S$.[6] The number of bit errors is minimized by using a Gray code. In the step of determining S, we set the l-bit representation of the interval index i equal to the ith codeword of the Gray code (see Fig. 16.11). In this way, a reconstruction error $i' = i \pm 1$ only leads to a single bit flip in S'.

[6] The error probability $\mathbb{P}[X' + wB(X') \notin A_{i(X)}]$ depends on the choice of l and B.

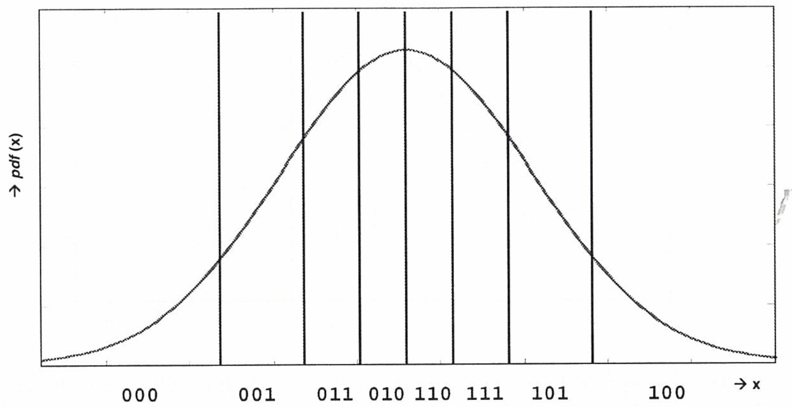

Fig. 16.11. Equiprobable discretization into $l = 3$ bits. Robustness is improved by assigning Gray codewords to the intervals.

16.6.3 Experimental Results

We have produced a batch of coated ICs as described in Chapter 15. The top metal layer of the IC contains 31 sensor structures. We have measured the capacitances from 90 different ICs.

Capacitance Measurements

As explained in Chapter 15, a capacitance measurement at location i yields an integer counter value C_i. In order to suppress noise, each measurement is immediately followed by a "blank" measurement in which no capacitor is addressed. This results in a counter value C_0. The difference $D_i := C_i - C_0$ is proportional to the RC time of the ith capacitor. In order to obtain stochastic variables with zero average, we subtract the average over all sensors ($D_{\text{av}} = \sum_{i=1}^{m} D_i$, where m is the number of sensors) from each measurement:

$$f_i = D_i - D_{\text{av}}. \tag{16.3}$$

On each IC, one of the 31 sensors is used as a reference sensor (with value f_{ref}) for compensating temperature variations. We assume that all enrollment measurements f_i and f_{ref} are done at a controlled temperature T_0. The enrolled value f_{ref} is stored along with the helper data.

However, in the key reconstruction phase, the temperature is not under control. Measurements under these conditions are denoted with a prime (i.e., f_i' and f_{ref}'). Only the quotient of two such measurements yields a reproducible value. Hence, we define temperature-compensated values B_i as follows:

$$b_i = \frac{f_i'}{f_{\text{ref}}'} f_{\text{ref}}. \tag{16.4}$$

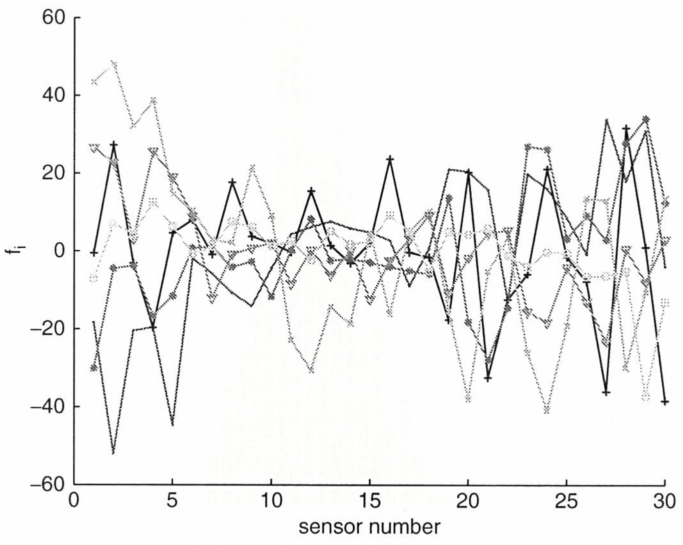

Fig. 16.12. Capacitance values f_i at 30 sensors of 6 different ICs.

Figure 16.12 shows the f_i values of 30 sensors, measured at 6 different ICs. Measurements of similar coating and sensor structures with a Hewlett Packard 4192 impedance analyzer show that the average capacitance value is around 0.18 pF (i.e., corresponding to 0 in Fig. 16.12). The capacitance measurements show an average within-class standard deviation of $\sigma_N = 0.95$ and an average between-class standard deviation of $\sigma_f = 18.8$. In our practical setup, we derive 3 bits per sensor, which gives the best robustness results.

Fingerprints

By way of example, we show key extraction from our experimental data according to the method of Section 16.6.2. First, the distribution $\rho(f_i)$ was estimated empirically by measuring all 30 sensors on 90 ICs. The range of measured f_i values was divided into $L = 2^3 = 8$ intervals and assigned a 3-bit Gray code to each interval. In this way, we derived fingerprints of 90 bits. Histograms of the fractional Hamming distances between the extracted fingerprints for the between-class distribution are shown in Fig. 16.13. The between-class distribution is centered around a fractional Hamming distance of 0.5, which means that the fingerprints derived from two different ICs will, on average, differ in 50% of the bits.

It turns out that bitstrings derived from the same IC (within-class distribution) have at most four bit errors, with an average of approximately one error. Hence, an error-correcting code that corrects 4/90 of all bits is suitable in this case. For example, a $(127, 99, 9)$ BCH code that corrects 4 errors from 127-bit codewords (with 99 information bits) can be used. By setting 37 of

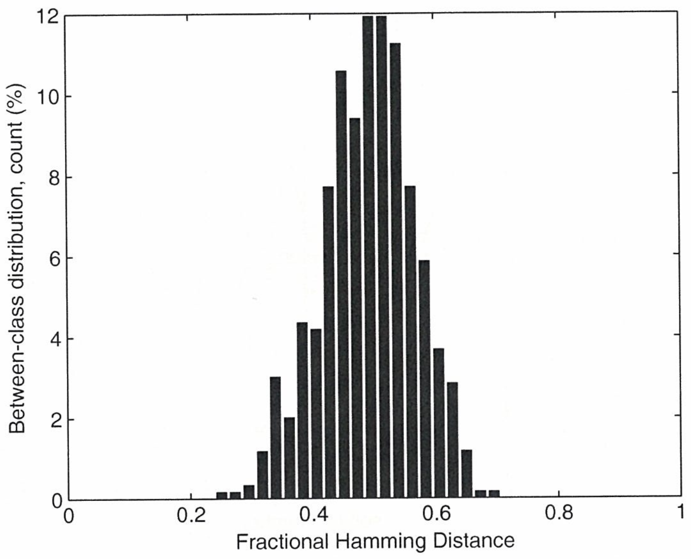

Fig. 16.13. Histogram of fractional Hamming distances between fingerprints derived from different ICs (between-class).

the information bits to 0 (code shortening) we effectively achieve a code that corrects 4 out of 90 bits and the remaining key size is $99 - 37 = 62$ bits. This corresponds to an entropy density of the order of 100 bits per square millimeter of the coating.

Robustness against Temperature Variations

The measured capacitance values f'_i and f'_{ref} increase with increasing temperature. This is shown in Fig. 16.14. In the key reconstruction phase, dividing the enrolled reference value f_{ref} by the measured f'_{ref} gives a temperature compensation factor. The resulting B_i values (16.4) are depicted in Fig. 16.15. Note that the temperature effects are almost completely compensated.

Attack Detection

Physical attacks in which the coating is damaged are detected from the capacitance measurements. A well-known method for getting access to internal circuit lines of an IC is making a hole through the IC with a FIB. Afterward the hole is filled with metal such that a surface contact is created. This can be used by the attacker for easy access to an internal line.

A FIB was used to create a hole in the coating of an IC by shooting gallium particles on an area of size $10\,\mu\mathrm{m} \times 10\,\mu\mathrm{m}$ above sensor 18 (see Fig. 16.16). The depth of the hole was around $5\,\mu\mathrm{m}$, whereas the coating thickness was around $4.5\,\mu\mathrm{m}$.

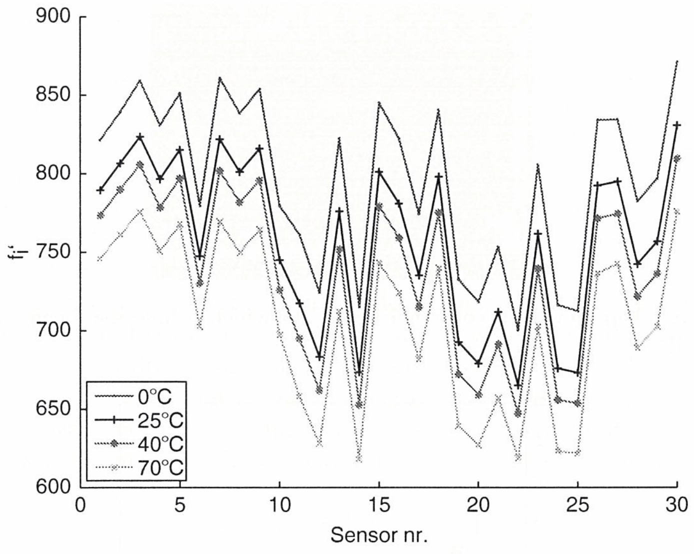

Fig. 16.14. f_i' values for IC 2, measured at 0°C, 25°C, 40°C, and 70°C.

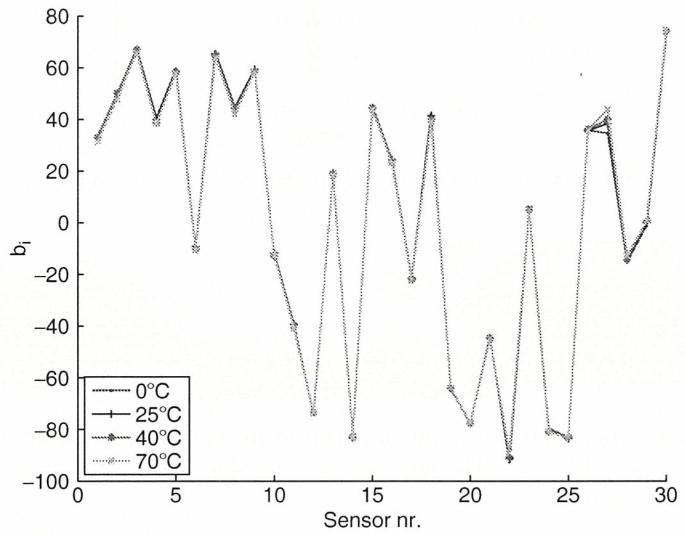

Fig. 16.15. B_i values for IC 2, measured at 0°C, 25°C, 40°C, and 70°C.

Figure 16.17 shows the effect of the FIB attack on the measured capacitances f_i'. A significant decrease in capacitance is measured at sensor 18, right under the area of impact of the FIB beam. The measurement values at other sensors are also slightly influenced. A cross-sectional scanning electron micrography image of the created hole is depicted in Fig.16.18. Table 16.1 summarizes the direct effect of gallium FIB and argon beam attacks with different hit areas.

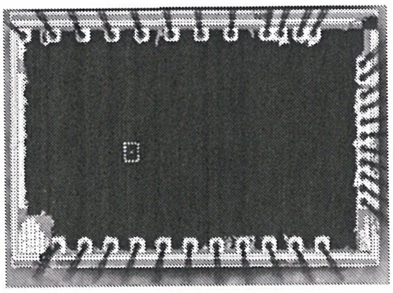

Fig. 16.16. Top view of a coating PUF IC in which a hole has been shot with a gallium FIB.

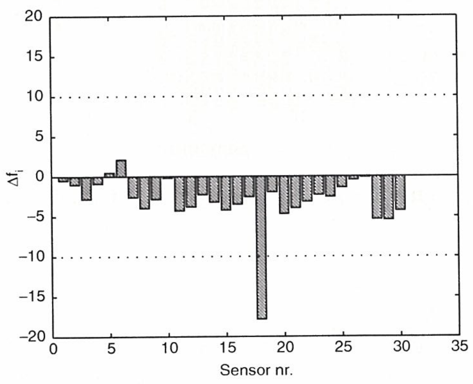

Fig. 16.17. Differences in capacitance f_i' before and after the gallium FIB attack.

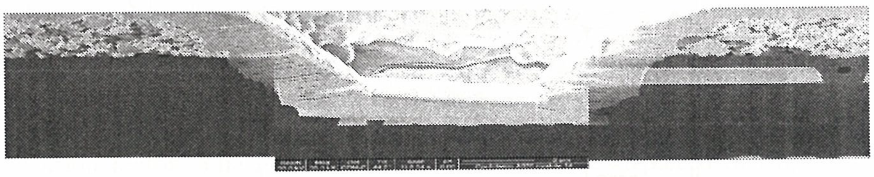

Fig. 16.18. Cross-sectional scanning electron micrography image of a FIB hole above sensor 18 of the IC. The hole has an area of $10\,\mu$m $\times$ $10\,\mu$m and a depth of approximately $4\,\mu$m.

Table 16.1. Average change of capacitance measured by the sensor lying under the area of impact of the beam

Beam type	Hit Area	Depth	Δf
Gallium	$100\,\mu$m $\times$ $100\,\mu$m	$1.5\,\mu$m	-40
Gallium	$15\,\mu$m $\times$ $15\,\mu$m	$4\,\mu$m	-34
Gallium	$10\,\mu$m $\times$ $10\,\mu$m	$5\,\mu$m	-15
Argon	$100\,\mu$m $\times$ $100\,\mu$m	$1.5\,\mu$m	-28

16.6.4 Security of the Coating

Since the coating is opaque, optically looking into the digital memory is very hard without damaging the coating. Furthermore, since the coating is tough and chemically inert, it is very hard to remove mechanically or chemically. In a more advanced attack, the attacker first uses a FIB to make a hole in the coating and then makes the IC start the key reconstruction phase. During this phase, the attacker uses microprobes to retrieve key bits. Obviously, since he has damaged the coating, the original key will not be reconstructed. Yet, the attacker might still be able to retrieve information about the original key. In Section 16.6.5, the FIB attack is modeled as an additional bit error rate ϵ on top of the errors α due to measurement noise (where $\epsilon > \alpha$). This effectively leads to a noisy channel with combined error rate $\chi = \alpha(1-\epsilon) + \epsilon(1-\alpha)$, as seen by the attacker. The amount of uncertainty he has about the key K can be expressed as a number N_c of "candidate" keys, which turns out to be of the order

$$N_c = \mathcal{O}\left(2^{n[\mathsf{h}(\chi) - \mathsf{h}(R\alpha)]}\right), \tag{16.5}$$

where $R > 1$ is a constant such that the code corrects error rates up to $R\alpha$, and the function h is defined as $\mathsf{h}(p) = -p \log p - (1-p)\log(1-p)$. This formula is derived in Section 16.6.5. Based on experimentally measured error rates, we estimate the parameters α and $R\alpha$ by $\alpha = 1/30$ and $R\alpha = 4/90$. The values for ϵ range from $\epsilon = 8/90$ to $\epsilon = 14/90$. Therefore, we take an average value $\epsilon = 11/90$. In practice, one would like to have a key of length 128 bits for encryption purposes. Given these error rates that would require $n = 174$ (in order to have a mutual information of 128 bits between two noisy strings). Substituting this value of n into (16.5), we obtain $N_c = 2^{51}$.

16.6.5 FIB Attack Model

In this subsection we estimate the impact of a FIB attack on the secrecy of the key $K \in \{0,1\}^k$. We model the output of the enrollment phase as a binary string $X \in \{0,1\}^n$. The string X is modeled as a string of n i.i.d. random variables. The measurement results during the key reconstruction phase of an undamaged coating are modeled as random variables $Y \in \{0,1\}^n$. The "noise" between X and Y is modeled as a binary symmetric channel $X \to Y$ with bit flip probability α. Hence, this situation allows one to use the technique based on error-correcting codes described at the end of Section 16.3.1. The Error-Correcting Code (ECC) $\mathcal{C}$ is capable of correcting $nR\alpha$ errors, where $R > 1$ is a constant slightly larger than 1. Hence, if the Hamming distance between Y and the codeword c_K is smaller than $nR\alpha$, then Y is successfully decoded to K. As we saw in Section 16.6.3, an invasive attack with a FIB damages the coating (which changes its challenge response behavior). We denote the outcome of the measurement after FIB attack by a random variable $Z \in \{0,1\}^n$. The process of damaging the coating is modeled as an adversarial channel $Y \to Z$

with crossover probability $\epsilon > \alpha$. The total error rate χ (for the attacker) is the result of two independent error-inducing processes, namely noise and damage:

$$\chi = \alpha(1-\epsilon) + \epsilon(1-\alpha) = \alpha + \epsilon - 2\alpha\epsilon. \tag{16.6}$$

We assume that the parameters α and ϵ are known to the attacker. The attack is successful if Z still provides enough information to efficiently compute K.

We present an estimate (in the case of large n) of the amount of effort needed to obtain K from the knowledge of Z, α, and ϵ. The attacker knows that any n-bit string within a "sphere" of radius $nR\alpha$ around c_K will decode to K. He also knows the probability distribution of the Hamming distance D between c_K and Z. It is the binomial distribution

$$\mathbb{P}[D = \beta] = \binom{n}{\beta} \chi^\beta (1-\chi)^{n-\beta}. \tag{16.7}$$

The attacker knows the average $\langle D \rangle = n\chi$ and the variance $\sigma_D = \sqrt{n\chi(1-\chi)}$. There is an overwhelming probability (due to the law of large numbers) that c_K lies in a shell of thickness σ_D at a distance $\langle D \rangle$ from Z. The attacker makes a list of all codewords lying in that shell. With overwhelming probability, one of those codewords will decode to K. The number N_c of "candidate" codewords in the shell is given by the volume of the shell times the density of codewords in the space $\{0,1\}^n$. This number represents the amount of computational effort that the attacker has to expend in an exhaustive attack to find the secret K or, equivalently, the uncertainty that he still has about the key K.

The shell volume is of the order $\sigma_D \binom{n}{\langle D \rangle}$. Assuming that the codewords are uniformly distributed, the density of codewords is 2^{k-n}, where k is the number of information bits of the codewords. Thus, the required attack effort is given by

$$N_\mathrm{c} \approx 2^{k-n} \sigma_D \binom{n}{\langle D \rangle} = 2^{k-n} \sqrt{n\chi(1-\chi)} \binom{n}{n\chi} \approx (2\pi)^{-1/2} 2^{n\mathsf{h}(\chi)+k-n}, \tag{16.8}$$

where h is the binary entropy function. Here, we have used Stirling's formula to approximate the binomial. An "optimal" code for correcting $nR\alpha$ errors has message length $k = n - nh(R\alpha)$. For such a code, the complexity of the attack effort is of the order

$$N_\mathrm{c} = \mathcal{O}\left(2^{n[\mathsf{h}(\chi) - \mathsf{h}(R\alpha)]}\right). \tag{16.9}$$

17

Anti-Counterfeiting

Pim Tuyls, Jorge Guajardo, Lejla Batina, and Tim Kerins

17.1 Introduction

Counterfeiting of goods is becoming a very huge problem for our society. It not only has a global economic impact, but it also poses a serious threat to our global safety and health. Currently, global economic damage across all industries due to the counterfeiting of goods is estimated at over $600 billion annually [2]. In the United States, seizure of counterfeit goods has tripled in the last 5 years, and in Europe, over 100 million pirated and counterfeit goods were seized in 2004. Fake products cost businesses in the United Kingdom approximately $17 billion [2]. In India, 15% of fast-moving consumer goods and 38% of auto parts are counterfeit. Other industries in which many goods are being counterfeit are the toy industry, content and software, cosmetics, publishing, food and beverages, tobacco, apparel, sports goods, cards, and so forth.

The above-mentioned examples show that counterfeiting may lead to highly reduced income for companies and to a damaged brand. Perhaps more unexpectedly, counterfeiting has some impact on our safety too. In some cases, this is even tragic. Counterfeited spare parts of planes have caused planes to crash [148]. Counterfeiting of medicines poses a real growing threat to our health. Thousands of people die because of taking medicines containing ingredients that are very dangerous or taking medicines that do not contain any active ingredient at all. This kind of problem is very large in Southeast Asia where many fake anti-malaria drugs containing no active ingredient are being distributed. The World Health Organisation (WHO) estimates that counterfeit drugs account for 10% of the world pharmaceutical market, representing currently a value of $32 billion. In developing countries, these numbers are often much higher: In China and Colombia, 40% of drugs are counterfeit; in Vietnam, 33% of the anti-malaria drugs are fake; and in Nigeria, 50% of the drugs are counterfeit. According to the Food and Drug Administration, in the United States there has been a rise of 600% in counterfeit drugs since 1997. In Europe, the rise increased by 45% since 2003.