# Studying Leakages on an Embedded Biometric System Using Side Channel Analysis

Maël Berthier[1], Yves Bocktaels[1], Julien Bringer[1(✉)], Hervé Chabanne[1,2], Taoufik Chouta[2], Jean-Luc Danger[2], Mélanie Favre[1], and Tarik Graba[2]

[1] Morpho, Issy-les-Moulineaux, France
julien.bringer@morpho.com
[2] Télécom ParisTech Identity and Security Alliance
(The Morpho and Télécom ParisTech Research Center), Paris, France

**Abstract.** This paper addresses the potential information leakages of a fingerprint comparison algorithm embedded as a hardware implementation. Such solution aims at comparing a reference fingerprint with a freshly acquired one completely inside an embedded system (e.g. ASIC, smart card, FPGA). The same way as for cryptographic operations within a cryptoprocessor, we consider the reference fingerprint template as a sensitive data that one may try to retrieve by attacking the chip. On one hand, we show that we can find relevant information by the means of Side Channel Analysis (SCA) that may help to retrieve the reference fingerprint. On the other hand, we illustrate that reconstructing the fingerprint remains not trivial and we give some simple countermeasures to protect further the comparison algorithm.

**Keywords:** Side channel analysis · Fingerprint · Hardware biometric coprocessor · Biometric comparison · Hill climbing

## 1 Introduction

Biometric authentication, particularly using fingerprints, is commonly used to uniquely identify individuals. Compared to the well know *What I know* (password) and *What I have* (token), the *Who I am* (biometrics) offers an inherent security. However, biometric data are personal data and their usage in authentication systems requires to take care of privacy issues. Compared to a database, the use of a personal device as a smart card to store the reference template is a way to protect it and thus be compliant to user privacy. An even better approach is the Match-On-Card (MOC) principle as it performs the comparison[1] inside the smart card [7,10,12,19]. The demand for such devices is growing. At *Fingerprint Verification Competition* (FVC) of 2004 [3], a new competition category was added to evaluate performances of authentications under resource constraints: a 1.41 GHz working frequency, a maximum of 4 MB RAM usage

[1] The comparison algorithm is often also called a matching algorithm.


E. Prouff (Ed.): COSADE 2014, LNCS 8622, pp. 281–298, 2014.
DOI: 10.1007/978-3-319-10175-0_19

and matching execution time limited to 0.3 s. Even with this restrictions, the available resources on these platforms are far better than what we can find in a common smart card used for authentication (around 30 MHz frequency and 5 KB RAM in [7]). Recently, to overcome the limited resources of a smart card when a comparison algorithm is implemented in software, [9] introduced the design of a hardware implementation of a fingerprint comparison algorithm in order to define a biometric coprocessor, similarly to what had been done years ago for cryptographic coprocessors to speed up cryptographic operations. Note that some other embedded implementations for small devices have been proposed earlier (see for instance [21]), but we focus on the work presented in [9] as it is based on a classical fingerprint comparison algorithm.

A parallel to embedded cryptographic implementations on electronic chips can thus be done by evaluating the information leakages of the biometric comparison algorithm. The so called Side Channel Analysis (SCA) consists in passively exploiting leaked information. Since Kocher presented the first *timing analysis* to extract the private key of RSA asymmetric ciphering algorithm [14], a lot of other vulnerabilities were studied mainly related to power consumption [15] and electromagnetic emanations [17]. In this work we want to take advantage of leakages on something else than cryptographic operations, namely biometric comparison. These leakages have not the same consequences than for cryptography: while the knowledge of a secret is targeted in the latter, in biometrics it's authentication that is sought, like, for instance, in PIN comparison.

Concerning the security of biometric matching systems, authors of [18] identified 8 points of vulnerability that an attacker may exploit. In fact, a generic biometric system can be divided into four main modules (see Fig. 1): the *sensor* taking a raw image of the fingerprint, the *extractor* that performs pre-processing and features extraction, the *matcher* that calculates the similarity between two biometric templates, leading to a similarity *score*, and the *database* that contains the reference template. The embedded comparison approach, or Match-On-Card, only considers the matcher and the reference fingerprint template.

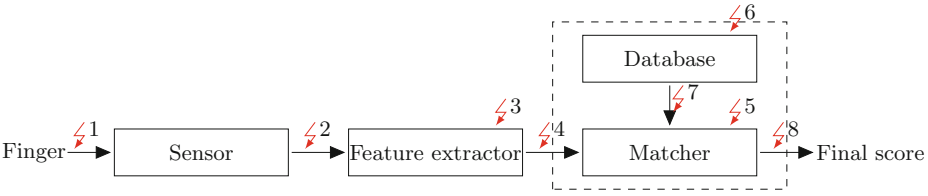



**Fig. 1.** Modules of a generic biometric system

Specific attention has also been paid to *Hill Climbing* attacks [16,22]. These algorithms produce synthetic templates iteratively adapted to the score they produce. We can as well cite a timing analysis on fingerprint matching [11] where authors show that there can be a correlation between execution time and score.

There is a mention of SCA on Match-On-Card in [6] but, to our knowledge, this has not been studied much further. The ThumbPod project [21] has designed an FPGA implementation (cf. for instance [20,23]) that resists to side channel leakage thanks to dual rail techniques but the biometric algorithm used [24] is not a standard one contrary to the one used in [9] and the study made was not specific to the biometric leakages. Note again that side-channel analysis on biometric comparison has been hardly studied in the literature and so have the countermeasures, that is why our analysis could rely on simpler ones compared to what is known today for attacking protected cryptographic implementations.

In this paper we present methods based on the simple analysis of power consumption during the matching process within an embedded system to recover some sensitive information. Then, we go further in our analysis of the leakage by presenting a template based attack that permits to retrieve, under some conditions, the hidden comparison score. All in one, this enables to launch an improved hill climbing algorithm to approach the reference fingerprint template. We illustrate our work on the hardware biometric comparison solution described in [9]. We present also some simple countermeasures to strengthen the embedded matcher against these information leakages. Our main goal is to highlight how hardware biometric solutions like [9], that rely on state-of-the-art minutiae-based fingerprint comparison techniques, could be improved to lead to a secure biometric coprocessor, thus avoiding sensitive leakages.

The article is structured as follows. In Sect. 2, we give some general information about fingerprint biometrics and the studied Match-On-Card algorithm and about its hardware implementation. Section 3 presents our observations coming from Side Channel Analysis while Sect. 4 presents a template attack on the matching score. Section 5 deals with the exploitation of the leakages mainly based on a hill climbing strategy. Finally, we give some countermeasures in Sect. 6.

## 2 Biometric Matching System

### 2.1 Fingerprint Biometrics

Fingerprints are one of the most used biometrics. The matching process is commonly based on the similarity analysis of some specific points called minutiae, extracted from a fingerprint image. Minutiae are discontinuity points on the ridge flows (ridge ending and ridge bifurcation). The INCITS 378 and the ISO 19794-2 [4] standards specify a compact template format based on minutiae for limited resource systems. Each fingerprint can be represented as a set of 3-dimensional minutiae points, where a single minutia point is described as an oriented 2D point $\{x(8\,\text{bits}),\ y(8\,\text{bits}),\ \theta(6\,\text{bits})\}$. The angle $\theta$ is the ridge ending or bifurcation orientation. Fingerprint comparison algorithms aim at best superimpose both minutiae sets and measure their similarity. In what follows we consider that the sensitive data that we are aiming to retrieve from the embedded system is a set of standard minutiae points $\mathbb{S} = (\{(x_0, y_0, \theta_0)\}, \ldots, \{x_n, y_n, \theta_n\})$.

### 2.2 The Studied Fingerprint Matching Module

In [9], the authors propose a hardware module to achieve an embedded biometric comparison (hardware MOC), with the goal to define a biometric coprocessor, the aim being to speed up operations as do cryptographic coprocessors. The corresponding algorithm has two main steps called registration and pairing. Registration phase aims at retrieving best rotation and translation that make overlap reference and input minutiae sets. After applying this affine transformation to the input set, pairing uses a Gaussian scoring method to evaluate more accurately the similarity between both sets.

The coprocessor is composed of three modules (*Transformation*, *Votes* and *Pairing*). It uses a Read Only Memory (ROM) to store the reference minutiae and has a private volatile memory for all the processing steps. For our study we have used a SASEBO GII board [1] that is specially designed for the study of side channels and that includes a Virtex-5 LX30 FPGA on which the coprocessor was embedded.

Compared to the main related works on biometric comparison with hardware implementations, two important properties of [9]'s implementation are that it relies on a biometric algorithm working simply with a standard compact fingerprint template [4] and that is very close to the best performing algorithms with respect to biometric error rates. For instance, with FVC2000 DB2 dataset (cf. [2]), it achieves 1.50 % of false reject rate at $10^{-3}$ of false acceptance rate. The speed of one comparison is also sufficiently good (less than 0.5 s) to enable efficient side channel captures.

**First Phase of a Fingerprint Comparison: Registration.** Registration (also called alignment) consists in the construction of a histogram of all possible affine transformations $(\Delta_x, \Delta_y, \Delta_\theta)$ by overlapping each input minutia with each reference minutia. The most voted parameter triplet is considered to be optimal. However, the number of possible transformations is too big to store the whole histogram in a smart card. Its construction is thus adapted by dividing the research space in many small subspaces and votes are only done with respect to the processed subspace. This allows to reduce the size of the embedded memory to the size of a subspace: the same memory is used for all subspace histograms. These sub-histograms are calculated in an increasing rotation angle $(\Delta_\theta)$ from $\Delta_{\theta_{min}}$ to $\Delta_{\theta_{max}}$ and their memory is completely reset between each subset. The most voted $(\Delta_x, \Delta_y, \Delta_\theta)$ triplet is updated on the fly in an internal register.

The drawback of this optimization is the need to test all possible affine transformations for each subspace even if the result is not within the processed subspace borders. To improve the processing time, the sub-histogram construction is not done on the whole reference minutiae set. For each minutia of the input set, only minutiae of the reference set, such that the difference in orientation angles $(\Delta_\theta)$ belongs to the subspace, are tested. To optimize the research of these particular reference minutiae, the reference set is sorted offline regarding the minutiae angle. A mapping array is added, called *set_access*, with the orientation angle as key, to point directly to the first and last minutiae (noted $F_{\theta i}$ and

$L_{\theta i}$) with this particular orientation angle. A special *NONE* value is used if no reference minutia has this orientation angle. Figure 2 and Algorithm 1 describe the iterative registration process. $m_{ref}$ denotes a minutia of the reference fingerprint and $m_{in}$ a minutia of the input fingerprint (the fingerprint that has been submitted to the embedded comparison module).

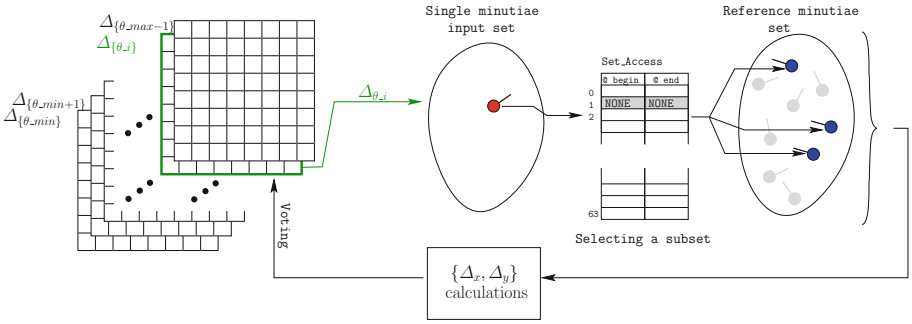



**Fig. 2.** Sub-histogram construction using a memory mapping array

**foreach** $\Delta_{\theta i} \in [\Delta_{\theta_{min}}, \Delta_{\theta_{max}}]$ **do**
  **foreach** *subspace* **do**
    **foreach** $m_{in} \in$ *input set* **do**
      Read $(F_{\theta_i}, L_{\theta_i}) = set\_access(\Delta_{\theta_i} + \theta_{m_{in}})$
      **if** $F_{\theta_i} \neq NONE$ **and** $L_{\theta i} \neq NONE$ **then**
        Calculate $\Delta_x$ and $\Delta_y$ parameters
        Fill subspace histogram memory with votes
        Update best $\{\Delta_\theta, \Delta_x, \Delta_y\}$ if greater triplet is voted
      **else**
        Continue `// No processing activity`
      **end**
    **end**
  **end**
  Erase subspace memory
**end**

**Algorithm 1.** Subspaces histogram built during registration phase

**Second Phase of the Fingerprint Comparison: Pairing.** In the pairing phase the affine transformation found during registration is applied on the input set. Then a similarity measure is used to associate pairs of input and reference minutiae: each input set point is iteratively compared to all the points of the reference set. If close enough, the reference minutia resulting in the highest pairing score is paired with the processed input minutia. Pairing phase is therefore data dependent, the number of input and reference minutiae is directly related to the duration of this step. Algorithm 2 illustrates the pairing process after the affine transformation has been applied.

```
foreach m_in ∈ input set do
    max_3D_score = 0
    pair[m_in] = {none,0}
    foreach m_ref ∈ reference set do
        pair_score = Gauss(Dist_θ, Dist_X, Dist_Y)
        if max_3D_score ⩽ pair_score then
            max_3D_score = pair_score
            pair[m_in] = {m_ref, pair_score }
        else
            Continue // No processing activity
        end
    end
end
Compute final score using local scores in pair
```

**Algorithm 2.** Pairing phase

**The Final Score Computation and Decision.** After the pairing, the final matching score is computed by summing all the individual pairing scores. The final decision is then taken by comparing a normalized value of the score with a predefined threshold $Score_{Th}$.

The normalization of the score is necessary because the two minutiae sets could have very different sizes leading to erroneous results. In the studied implementation, the computation of the final score is done as follows:

$$finalScore = \frac{\sum_{i=0}^{size_{in}} pair[i]}{\mathrm{Max}(size_{in}, size_{ref})} \quad (1)$$

Where $size_{in}$ and $size_{ref}$ are respectively the number of minutiae in the input and reference sets.

Note that this approach in three steps for fingerprint comparison is quite classical. Consequently our side-channel analysis and associated results discussed in the remaining of the paper can probably be also adapted to other comparison algorithms that rely on the standard minutiae representation.

### 2.3 Assumptions on the Matching System

The studied biometric matching system structure is compliant to the one pictured in Fig. 1 but we can additionally make the following assumptions[2] on it, in order to simplify the study, as we aim to define recommendations for designing a secure biometric hardware coprocessor:

- We have full control of the inputs;
- There is no protection of the implementation:
  - There are no side channel countermeasures;
  - There is no retry counter (i.e. any number of attempts is possible).

[2] Note that the scope here is not to discuss the security of any existing Match-On-Card products.

All these points will greatly help us to study the information leakages of the design.

## 3 Information Leakage

The studied biometric hardware module behaves as follows. The private reference fingerprint template is stored in the module and the input fingerprint template is sent directly to the matcher. This means that the attacker has a complete control on the submitted fingerprint (the one sent as input to the module). During the matching execution, both reference and submitted fingerprint are manipulated, generating secret dependent variations on power consumption.

As an analogy with usual side channel analysis on cryptographic processes, we will study here the impact of manipulating a secret data (reference fingerprint is used here instead of the secret key for classical side channel analysis) and a chosen data (a chosen fingerprint sent to the comparison algorithm is used here instead of a plain text message for classical side channel analysis). However there are several differences:

The size of the secrets space, for example on an AES (Advanced Encryption Standard) is $2^{128}$, with a 128-bit key. For our fingerprint comparison scenario, each minutia is represented on 22 bits (8 bits for $x$ and $y$, and 6 bits for the angle), which means that with an average minutiae number of 20, the average secrets space size is upper bounded by $2^{440}$.

On the other hand, a single bit difference on the secret key in cryptography directly leads to a rejection while an error on fingerprint acquisition is allowed (more or less minutiae, slight shift on position or angle of a minutia...). Thus the attacker may gain an interesting advantage by adapting the submitted fingerprints during an attack.

In the sequel we use Simple Power Analysis (SPA) in order to identify some patterns in power consumption which give information about what is executed on the target chip. As usual, this is made by measuring current that flows from the power supply to the attacked device.

### 3.1 SPA on Pairing Phase

In the second part of the matching execution, each minutia of the reference fingerprint is compared with all the transformed input fingerprint minutiae. On Fig. 3, we can see that the pairing phase is composed of $Size_{in}$ similar patterns that correspond to the iterations of the pairing loop. If we zoom on a single loop iteration, we can identify $Size_{ref} + 1$ steps. For each input minutia (the outer loop), there are $Size_{ref}$ accesses to reference minutiae plus one access to the input minutia access. A simple count gives the size of the reference minutiae set.

### 3.2 SPA on Registration Phase

As we can see from the Algorithm 1, there is a difference of process activity if the *set_access* value for a specific angle is *NONE* or not.

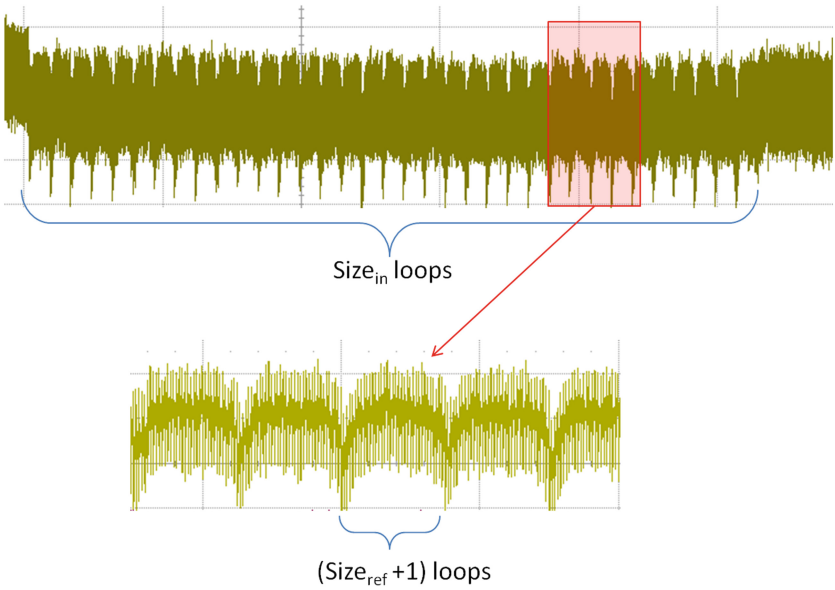



**Fig. 3.** Information leakage on pairing step

Since we have full control on the input fingerprint, we tried to submit a single minutia as a fingerprint input to reveal some activity which only depends on the reference fingerprint. The coordinates of the single minutia are not important, but we set the angle value at 0, to start from the first angle. For each computed transformation, if all the corresponding differences in orientation angles $\Delta_{\theta i}$ are out of bounds (i.e. $[\theta_{in} - 1, \theta_{in} + 1]$), there will be a noticeable difference in power consumption due to the process activity inequality. This difference can be seen on the power consumption trace of the registration part (Fig. 4). The angle values of the reference fingerprint minutiae were distributed as follows:

$$\begin{aligned}&|19\ 20\ 20\ 20\ 22\ 23\ 23\ 24\ 25\ 26\ 26\ 27\ 27\ 27\ 28\ 28\ 28\ 30\ 30\ 31\ 31\ 32\ 32\ 34\\&|50\ 53\ 53\ 54\ 54\ 55|62\end{aligned} \tag{2}$$

We can see some drops in the power consumption which correspond to the angle area where there is no minutiae matching in the reference fingerprint (red lines in (2) vs. red markers in Fig. 4). This is due to the affine transformation of the input fingerprint (single minutia) that does not match with a reference one.

We then tried to analyze the dependence between the angle of the submitted fingerprint minutia and these drops on the trace. We processed several matchings with an increasing angle value and kept the trace for each match. Figure 5 shows the traces of 3 different matchings with an increasing angle value (not consecutive).

The drops are shifted to the left when we start the registration with a higher angle value. Increasing the angle of the input minutia from $i$ to $i + 1$ will cause

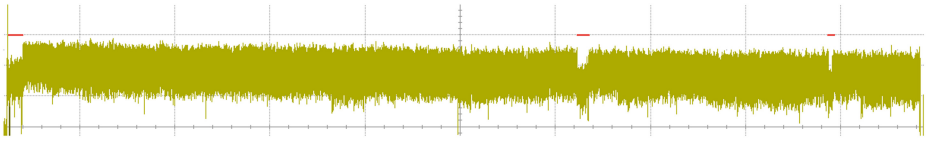


**Fig. 4.** Power consumption during registration with a single minutia input fingerprint (color figure online)

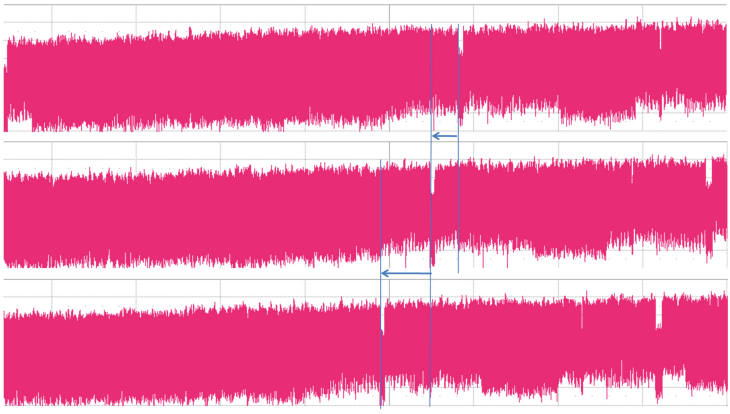


**Fig. 5.** Shifting drops in power consumption with 3 increasing input minutia angle values

a shift in the starting reference minutia from the angle $i-1$ to $i$. This means that we can get the number of minutiae for each angle value by increasing the angle value of the input minutia.

As we can see in Fig. 6, there is a strong dependence between the number of minutiae for a chosen angle in the reference fingerprint and the drops shift in power consumption. By observing the drops delay between two consecutive angle values of input minutia for each possible angle value, we are able to get the distribution of the reference fingerprint minutiae angles (the number of minutiae concerned by the $i^{th}$ angle value among the total number of minutiae). There are only 64 matchings to perform.

## 4 Side Channel Attack on the Comparison Score

A traditional approach to enhance privacy is to hide the score that can be exploited by Hill Climbing attacks in favor of a boolean answer. Therefore, we investigate side channels in order to retrieve the score when not directly available and thus we are able to climb back to the reference minutiae set.

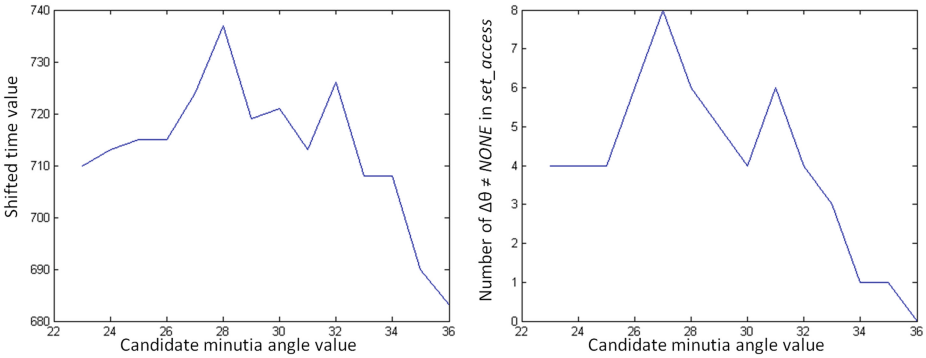



**Fig. 6.** Comparison between the number of minutiae neighbors in reference fingerprint and the value of drop shift

### 4.1 Introduction to Template Attack Combined with PCA

**Template Attack.** Template attack is a powerful statistical tool that is based on the leakage profiling of a similar device, which allows to retrieve the secret with less traces than differential attacks, or where these last ones simply fail [8]. Thus it is assumed that the adversary is in possession of an open similar device on which the *learning* can be done. Thus a state/operation $\mathcal{S}_i$ is characterized by computing its template $\mathcal{T}_{\mathcal{S}_i} = \{\mu_{\mathcal{S}_i}, Cov_{\mathcal{S}_i}\}$ consisting of the mean and the covariance matrix of the leakage traces respectively. In order to decrease the learning stage complexity, the computation is restricted to relevant leakage points as it will be discussed further in Sect. 4.2.

Therefrom, when another similar device is attacked, the adversary aims to reveal an unknown $\mathcal{S}_x$ by computing the maximum likelihood. Computation of the likelihood is done as the following:

$$p(\mathcal{L}|\mathcal{S} = \mathcal{S}_i) = \prod_{j=1}^{T} p(\mathcal{L}_j|\mu_{\mathcal{S}_i}, Cov_{\mathcal{S}_i}), \text{ where} \tag{3}$$

$$p(\mathcal{L}_j|\mu_{\mathcal{S}_i}, Cov_{\mathcal{S}_i}) = \frac{1}{\sqrt{(2\pi)^N |Cov|}} \times e^{-\frac{1}{2}(\mathcal{L}_j - \mu_{\mathcal{S}_i})^T Cov^{-1} (\mathcal{L}_j - \mu_{\mathcal{S}_i})}. \tag{4}$$

Where $i \in \{1, \ldots, \lambda\}$, with $\lambda$ the total number of possible states. $T$ is the set of leakage traces $\mathcal{L}$ each one of $N$ samples.

**Projection on Principal Components.** Computation magnitude of the maximum likelihood increases according to the number of used samples in leakage traces, which may result in significant resources loads. Inversion of the covariance matrix can also be one of the barriers prohibiting a direct computation of the likelihood. This may be due to the potential linearity between neighbour samples. Therefrom the adversary can consider the Principal Component Analysis [13] in order to keep only relevant informations (i.e., with maximum

variance). This operation is done by projecting templates and leakage traces into low dimensional subspaces. Computation of the principal components and projection matrices is out of the scope of this paper (see [5]). Thus, in our attack we use PCA to avoid covariance matrix issues.

### 4.2 Profiling and Attacking the Score Computation

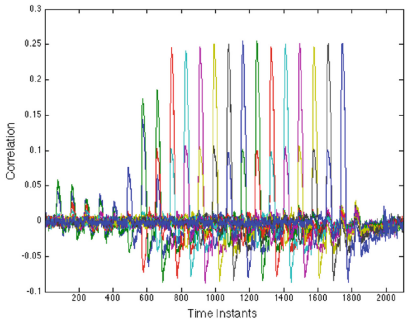
**The Hardware Implementation.** For our analysis we focused on profiling the score register consumption. The score computation stated in Eq. 1 is processed as follows: first, the register that will hold the final score is used to accumulate all local scores. In fact, this accumulation requires a 22-bit register and consists of the computation of the division nominator. Second, the accumulated scores are normalized by the $maxPairs$ denominator (see Sect. 2.2) by using a restoring division. This technique is a naive Euclidean approach that processes successive subtractions and comparisons, and outputs one quotient bit at each clock cycle. The binary version of this approach relies on successive left shifts of the nominator register which allows to reuse this register to store the quotient bits successively in the LSB. Thus, at the first clock cycle of this computation the score MSB is output and so on. Interestingly, the restoring division is one of the standard implementation that is adopted by many processors and hardware designers.

**The Learning Phase.** To perform the learning phase $10k$ traces were used. As the target register is an LFSR, we assume the Hamming distance between two consecutively computed bits as the leakage model. This results in two classes for each of the 22 bits. In practice, in order to determine relevant leakage moments, we compute the correlation coefficient between the $i^{th}$ bit model over all samples. Figure 7a shows the correlation traces for the 16 LSB, in order of computation. In fact, it turns out that the first 6 MSBs of the acquisition campaign have an unbalanced parity of 0 and 1 *(more than 90 % equal to* 0). Hence, for a proof of concept, we consider that profiling and attacking remaining bits is sufficient.

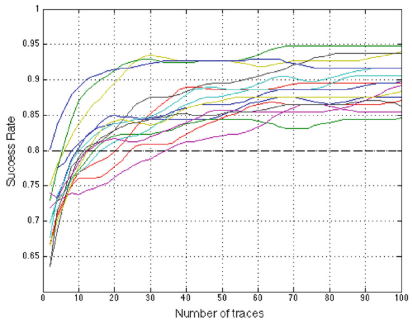
**The Attack Phase.** The success rate metric is a simple statistical tool giving the average of successful attacks on different sets of traces of different sizes. In other words, it allows to determine the average of what an adversary can achieve or expect with a certain amount of traces. For our attack, the amount of traces to reach a success rate of 80 %, varies according to the targeted bit from a single trace to 34 traces (the LSB need less traces). This is due to the low SNR consequence of the intrinsic and ambient noise. Indeed, the activity leaked by one register bit is low relatively to the rest of the device activity. In Fig. 7b we plot the success rate with 100 attack retries on independent sets of traces.

## 5 Exploitation

We emphasized in the previous sections different information that are observed through side channel from the comparison algorithm execution. We will explain here an advanced strategy to exploit those information.




(a) Localization of relevant samples




(b) Success rate of the template attack

**Fig. 7.** The proling and the attack results

### 5.1 Hill Climbing Attack

One of the possible attacks on a biometric system is to reach a positive verification using synthetic input minutiae sets rather than using the genuine user fingerprint. A brute-force attack is very hard unless the verification system has a significant discrimination error rate (false acceptance rate). This is due to the amount of minutiae points in a fingerprint template (≈20–100) which results in a possibility space of $2^{2200}$ ($(2^{(8+8+6)})^{100}$) in the worst case. Note that, in this rough estimation, we consider that the attacker has no knowledge on fingerprint geometry and will take into account the whole possibility space. Fingerprints with minutiae at the edges or with identical minutiae are hence considered.

A more efficient strategy exists: Authors of [22] used the Hill Climbing (HC) heuristic to find modifications that increase the comparison score between synthetic minutiae sets and the targeted reference set. It considers a starting set of minutiae points which is iteratively modified and sent back to the matcher module for score evaluation. An applied modification is kept only when the score increases. Possible modifications on a minutiae set are:

1. Randomly translate or rotate a randomly selected minutia;
2. Add a minutia;
3. Replace a randomly selected minutia;
4. Delete a randomly selected minutia.

The heuristic stops when the synthetic set reaches the matcher validation score for which sets are considered as sufficiently close to reference data. Thus, the attack on the matcher combined with this private reference fingerprint template is considered as a success.

Of course, the HC approach assumes that the attacker has a direct access to the matcher input (i.e. the attacker is able to choose the input fingerprint) and that the matching score is known (not only the binary OK/NOK result). The first condition is verified in our case following the assumptions explained in

Sect. 2.3. And we explained in the previous section how to retrieve the matching score, thus we assume below that the score could be known.

### 5.2 Hill Climbing Improvement

In the previous description of Hill Climbing, the added and modified minutiae are randomly chosen. This means that there are $2^{22} = 256 \times 256 \times 64$ possibilities each time we have to add or modify a single minutia. Our study on power consumption, as discussed in Sect. 3, gave some interesting information about the reference fingerprint template: the number of minutiae per angle. The most important information here is to have the distribution of the minutiae among the 64 angles.

A simple way to use this knowledge is to pick a minutia according to a distribution table. This distribution table, containing an associated probability for each angle, is created thanks to the shift timings values from the previous study. For each angle (64 matching executions) we store the time shift value among the total of all the 64 time shifts (which correspond to the registration step time).

To evaluate the improvement, we compared 3 different levels of Hill Climbing:

- Without optimization: new minutiae are picked randomly (equivalent to the HC in [22] with a single initial guess).
- List mode: new minutiae are picked from a list of existing angles, but there is no associated probability.
- Distribution mode: new minutiae are picked from a distribution table, with probabilities deduced from side channel analysis and thus approximately corresponding to those of the reference fingerprint template.

Figure 8 shows the result of these 3 modes on 4 times averaged Hill Climbing. It describes the score (vertical scale) among the matching iterations (horizontal scale). The horizontal line depicts the score threshold above which the synthesized fingerprint is accepted as corresponding to the reference one.

The distribution mode reaches the threshold score with 4000 iterations instead of 8000 iterations for the two other modes. To achieve this improvement, only 64 matching executions are necessary.

It has to be recalled (see Sect. 4.2) that the extraction of the score using the profiled attack, needs roughly 34 traces at each iteration. This gives an idea of the total number of traces needed to construct an approximation of the reference fingerprint data.

Keeping the assumptions from Sect. 2.3 verified, we are able to succeed a Hill Climbing with half the matching iterations otherwise needed. This improvement is possible thanks to the information leaked via side channel while executing biometric comparisons. We bet that deeper analysis of the side channel leakage would probably lead to further improvement. This means that some specific countermeasures have to be implemented to protect the biometric comparison coprocessor from that kind of leakages.

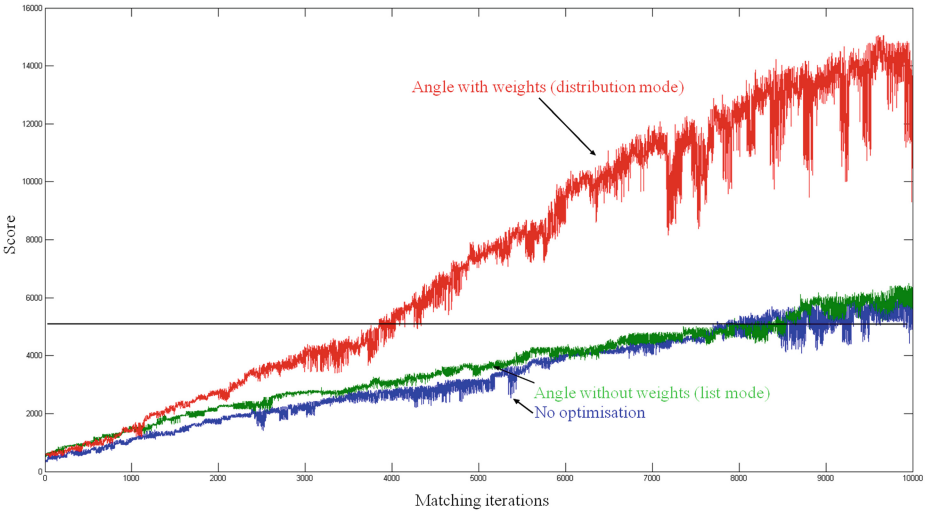



**Fig. 8.** Hill Climbing result for 3 modes, each replayed and averaged 4 times

## 6 Countermeasures

In previous sections we presented an approach that may allow an adversary to break through the simple countermeasure of hiding the score in order to perform the HC again. We also showed how it can be possible to reduce the HC needed iterations by a further exploitation of side channels. In this section, we will describe one countermeasure for each threat previously identified. Many of them require random numbers, which could be assumed coming from a random generator from the smart card in which the biometric coprocessor is attached to.

### 6.1 Protecting the Score Computation

Manipulation of the comparison score, whether by normalization (our case) or other approaches may leak sensitive information leading to its recovery. As stated in Sect. 2.2, in order to produce a binary answer, normalization of the accumulated score is followed by a comparison to acceptance threshold $Score_{Th}$. This operation can be expressed in a different way in manner to avoid normalization. Thus the accumulated score is compared to a dynamically adjusted threshold. This approach avoids the usage of a register which makes the combinatorial path the main source of leakage. The answer computation is thus:

Here, the number of minutiae in the reference set $size_{ref}$ is not considered as a sensitive information. Thus, even if the decision is made in two cycles (i.e. $DyScore_{Th}$ is stored in a register) and the adversary succeeds in retrieving $size_{ref}$, this last one is of a low entropy.

$AccuScore = \sum_{i=0}^{size_{in}} pair[i]$
$DyScore_{Th} = Score_{Th} \times \mathrm{Max}(size_{in}, size_{ref})$
**if** $DyScore_{Th} \leqslant AccuScore$ **then**
  Answer = 1
**else**
  Answer = 0
**end**

**Algorithm 3.** Hiding score computation

### 6.2 Randomization of the Registration Phase by Masking

A first method to protect the information leakage during the registration part is to start the registration from a randomly chosen rotation angle instead of going systematically from $\Delta_{\theta_{min}}$ to $\Delta_{\theta_{max}}$. This random offset value has to be different for each fingerprint comparison to avoid the correlation between the processing order and the orientation of the input minutiae. The same result can be obtained by applying on the reference fingerprint a randomly chosen pre-translation-rotation. This countermeasure would solve the incremental minutia angle parse threat, but would not be efficient enough because reference minutiae are still parsed in a sorted angle order. On average, 400 attempts of matching with a same single minutia will give the 64 angles distribution, with a 90 % success rate. In this case, the probability of obtaining the original minutiae parse sequence is 1/64.

A better countermeasure is to completely randomize the processing sequence regarding the orientation angle. An efficient way to achieve this is to use a randomly generated $mask$ to change the sequence order. There are 64 rotation angles to test, thus a $log_2(64)$-bit length vector $rot\_a$ is used to iterate through the sequence $\Delta_{\theta_{min}}..\Delta_{\theta_{max}}$. $rot\_a \oplus mask$ will give a random permutation of the original sequence. As the angle parse order is changed (and not only the angle start value), the drops on which we measure the time shift are split and other may appear. In that way, the angles distribution of the reference fingerprint is impossible to retrieve. The hardware cost of such a countermeasure is very small, and the probability of obtaining the original minutiae parse sequence is increased to 1/64!.

### 6.3 Input Fingerprint Requirements

The observation of the angles distribution of the reference minutiae is eased by the fact that we are allowed to send and match a single minutia fingerprint, or a fingerprint with several occurrences of the same minutia repeated. An either simple countermeasure would be to disable matching if the submitted fingerprint does not fulfill some basic requirements like:

- A minimum and maximum number of minutiae.
- No duplicated minutiae.

### 6.4 Random Additional Cycles During Pairing Phase

The pairing phase leaks some information about the reference fingerprint minutiae number. This information alone is not enough to improve a Hill Climbing attack, but it can still be protected with a low cost countermeasure.

As we have seen on Fig. 3, it is easy to count the number of cycles inside a reference minutia loop, and hence get the minutiae number of the reference fingerprint. Adding a random number of extra cycles per reference minutia loop would break this leakage and create a random delay effect on the whole pairing step. The idea is to choose a single random value $Rng\_FP$ which will be common to all reference minutiae loops, and an additional one $Rng\_minu_i$, different for each loop.

For instance, if $Rng\_FP$ is chosen with a maximum of 20 % of the reference minutiae number ($Rng\_FP \in [0; 0.2 * size_{ref}]$), and $Rng\_minu_i$ are chosen with max of 10 %, the average global extra computation time on pairing step will be 15 %. This is a low cost countermeasure as the pairing step represents less than 10 % of the whole matching process.

## 7 Conclusion

In this paper, we analyzed, for the first time, the potential information leakages of a hardware biometric comparison module that relies on state of the art fingerprint comparison techniques. We pointed out that we can find out relevant information of the private reference fingerprint template by the means of side channel analysis. These informations, together with a template attack to retrieve the value of the comparison score, enable us to mount an improved hill climbing attack to approach the reference template. This shows the need to protect the implementation. Fortunately, there are some simple countermeasures that can be used to thwart the information leakages. Our future work will thus cover the design study of a secure biometric coprocessor by including such kind of countermeasures.

**Acknowledgment.** This work has been partially funded by the French ANR project BMOS and by the European FP7 BEAT project (SEC-2011-284989). The authors would like to thank the other BMOS partners, especially Thibault Porteboeuf from Secure-IC, for their help on the FPGA prototype.

## References

1. http://www.rcis.aist.go.jp/special/SASEBO/SASEBO-GII-en.html
2. Fingerprint Verification Competition. http://biolab.csr.unibo.it/FVCOnGoing/
3. Fingerprint Verification Competition (2004). http://bias.csr.unibo.it/fvc2004/
4. Iso/iec 19794-2 information technology - biometric data interchange formats - part 2: Finger minutiae data

5. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
6. Barral, C., Vaudenay, S.: A protection scheme for moc-enabled smart cards. In: 2006 Biometrics Symposium: Special Session on Research at the Biometric Consortium Conference, pp. 1–6. IEEE (2006)
7. Bistarelli, S., Santini, F., Vaccarelli, A.: An asymmetric fingerprint matching algorithm for java card ™. Pattern Anal. Appl. **9**(4), 359–376 (2006)
8. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
9. Chouta, T., Danger, J.-L., Sauvage, L., Graba, T.: A small and high-performance coprocessor for fingerprint match-on-card. In: DSD, pp. 915–922. IEEE (2012)
10. Cucinotta, T., Brigo, R., Di Natale, M.: Hybrid fingerprint matching on programmable smart cards. In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2004. LNCS, vol. 3184, pp. 232–241. Springer, Heidelberg (2004)
11. Galbally, J., Carballo, S., Fierrez, J., Ortega-Garcia, J.: Vulnerability assessment of fingerprint matching based on time analysis. In: Fierrez, J., Ortega-Garcia, J., Esposito, A., Drygajlo, A., Faundez-Zanuy, M. (eds.) BioID MultiComm2009. LNCS, vol. 5707, pp. 285–292. Springer, Heidelberg (2009)
12. Govan, M., Buggy, T.: A computationally efficient fingerprint matching algorithm for implementation on smartcards. In: First IEEE International Conference on Biometrics: Theory, Applications, and Systems, 2007, BTAS 2007, pp. 1–6. IEEE (2007)
13. Jolliffe, I.: Principal Component Analysis. Wiley Online Library, New York (2005)
14. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
15. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
16. Martinez-Diaz, M., Fierrez-Aguilar, J., Alonso-Fernandez, F., Ortega-Garcia, V., Siguenza, J.: Hill-climbing and brute-force attacks on biometric systems: a case study in match-on-card fingerprint verification. In: Proceedings 2006 40th Annual IEEE International Carnahan Conferences Security Technology, pp. 151–159. IEEE (2006)
17. Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
18. Ratha, N.K., Connell, J.H., Bolle, R.M.: An analysis of minutiae matching strength. In: Bigun, J., Smeraldi, F. (eds.) AVBPA 2001. LNCS, vol. 2091, pp. 223–228. Springer, Heidelberg (2001)
19. Reisman, J., Uludag, U., Ross, A.: Secure fingerprint matching with external registration. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 720–729. Springer, Heidelberg (2005)
20. Tiri, K., Hwang, D., Hodjat, A., Lai, B.-C., Yang, S., Schaumont, P., Verbauwhede, I.: AES-based cryptographic and biometric security coprocessor IC in 0.18- μm CMOS resistant to side-channel power analysis attacks. In: 2005 Symposium on VLSI Circuits 2005. Digest of Technical Papers, pp. 216–219 (2005)
21. UCLA. Thumbpod: a next generation biometrically secure wireless embedded system. http://www.emsec.ee.ucla.edu/thumbpod

22. Uludag, U., Jain, A.K.: Attacks on biometric systems: a case study in fingerprints. In: Delp, E.J., Wong, P.W. (eds.) Security, Steganography, and Watermarking of Multimedia Contents. Proceedings of SPIE, vol. 5306, pp. 622–633. SPIE (2004)
23. Yang, S., Sakiyama, K., Verbauwhede, I.: Efficient and secure fingerprint verification for embedded devices. EURASIP J. Adv. Signal Process. **2006**(1), 058263 (2006)
24. Yang, S., Verbauwhede, I.: Automatic secure fingerprint verification system based on fuzzy vault scheme. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005), pp. 609–612 (2005)