# Studying Potential Side Channel Leakages on an Embedded Biometric Comparison System<sup>\*</sup>

Maël Berthier<sup>1</sup>, Yves Bocktaels<sup>1</sup>, Julien Bringer<sup>1</sup>, Hervé Chabanne<sup>1,2</sup>, Taoufik Chouta<sup>2</sup>, Jean-Luc Danger<sup>2</sup>, Mélanie Favre<sup>1</sup>, Tarik Graba<sup>2</sup>

<sup>1</sup> Morpho

<sup>2</sup> Télécom ParisTech

Identity and Security Alliance (The Morpho and Télécom ParisTech Research Center)

**Abstract.** We study in this work the potential side channel leakages of a hardware biometric comparison system that has been designed for fingerprints.

An embedded biometric system for comparison aims at comparing a stored biometric data with a freshly acquired one without the need to send the stored biometric data outside the system. Here one may try to retrieve the stored data via side channel, similarly as for embedded cryptographic modules where one may try to exploit side channel for attacking the modules.

On one hand, we show that we can find partial information by the means of simple Side Channel Analysis that may help to retrieve the stored 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.

### 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<sup>3</sup> inside the smart card [8,16,6,10]. 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 1.41GHz working frequency, a maximum of 4MB RAM usage and matching execution time limited to 0.3s. Even with this restrictions, the available resources on these platforms are far better than what we can find in a common smart

<sup>\*</sup> This work was partially funded by the French ANR project BMOS.

<sup>&</sup>lt;sup>3</sup> The comparison algorithm is often also called a matching algorithm.

card used for authentication (around 30MHz frequency and 5KB RAM in [6]). Recently, to overcome the limited resources of a smart card when a comparison algorithm is implemented in software, [7] 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 [18]), but we focus on this work 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 [11], a lot of other vulnerabilities were studied mainly related to power consumption [12] and electromagnetic emanations [14].

Concerning the security of biometric matching systems, authors of [15] identified 8 points of vulnerability that an attacker may exploit. In fact, a generic biometric system can be divided into four main modules (see Figure 1): the *sensor* taking a raw image of the fingerprint, the *extractor* that performs preprocessing 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 [19,13]. These algorithms produce synthetic templates iteratively adapted to the score they produce. We can as well cite a timing analysis on fingerprint matching [9] 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 [5] but, to our knowledge, this has not been studied much further. The ThumbPod project [18] has designed an FPGA implementation (cf. for instance [20,17]) that resists to side channel leakage thanks to dual rail techniques but the biometric algorithm used [21] is not a standard one contrary to the one used in [7] and the study made was not specific to the biometric leakages.

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. We illustrate our work on the hardware biometric comparison solution described in [7]. 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 [7] could be improved to lead to a secure biometric coprocessor, thus avoiding sensitive leakages.

The article is structured as follows. In section 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 section 4 deals with their exploitation mainly based on a hill climbing strategy. Finally, we give some countermeasures in section 5.

# 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 { $x(8bits), y(8bits), \theta(6bits)$ }. The angle  $\theta$  is the ridge ending or bifurcation orientation. 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  $\mathbb{S} = (\{(x_0, y_0, \theta_0)\}, \dots, \{x_n, y_n, \theta_n\}).$ 

#### 2.2 The studied fingerprint matching module

In [7], 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 [7]'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 second) to enable efficient side channel captures.

**Registration** Registration 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. The number of possible transformations is too important to store the whole histogram in a smart card. The histogram construction is adapted by dividing the research space in many subspaces and votes are only done with respect to processed subspace. This allows to reduce the size of the embedded memory to the size of a subspace.

Subspace histograms are then calculated in an increasing rotation angle  $(\Delta_{\theta})$ from  $\Delta_{\theta_{min}}$  to  $\Delta_{\theta_{max}}$ . The same embedded memory is thus used for all subspace histograms. Between two consecutive subspaces, the most voted  $(\Delta_x, \Delta_y, \Delta_{\theta})$ triplet is saved into an internal register, so it can be compared to the next subspace, and the memory buffer is completely reset.

The drawback of this optimization is the necessity to recalculate and 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 subhistogram construction is not done on the whole reference minutia set. Thus, for each minutia of the input set, the sole minutiae of the reference, such that the difference in orientation angles  $(\Delta_{\theta})$  belongs to the subspace, are tested. To optimize research of these particular reference minutiae, the reference set is sorted regarding the minutiae orientation angle. A mapping array is added, with the orientation angle as key, to point directly to the first and last minutiae (noted as  $F_{\theta_i}$  and  $L_{\theta_i}$ ) with this particular orientation angle. A special *NONE* value is used if no reference minutiae has this orientation angle. The mapping array is called *set\_access*. Figure 2 depicts the iterative registration process.



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

Algorithm 1 describes the histogram construction for the subspace during the registration phase with  $m_{ref}$  denoting 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).

```
for each \Delta_{\theta i} \in [\Delta_{\theta_{min}}, \Delta_{\theta_{max}}] do

for each subspace do

for each 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

else

| Continue // No processing activity

| end

end

Update best \{\Delta_{\theta}, \Delta_x, \Delta_y\} registers if greater triplet is voted

Erase subspace memory

end
```

Algorithm 1: Subspaces histogram built during registration phase

**Pairing** In the pairing phase the affine transformation found during registration is applied on the input set. Then similarity measure is used to associate pairs of input and reference minutiae. Thus, coordinates of each input set point are iteratively compared to all the points of the reference set. 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} \in input \ set \ do
    \max_{3D} score = 0
    \operatorname{pair}[m_{in}] = \{\operatorname{none}, 0\}
    tmp\_best\_ref = \{none, 0\}
    foreach m_{ref} \in reference \ set \ do
         pair_{score} = Gauss(Dist_{\theta}, Dist_X, Dist_Y)
         if max_3D_score \leq pair_{score} (1) then
             \max_{3D\_score} = pair_{score} (2)
             // Save m_{ref} with best score in temporary register
             tmp\_best\_ref = \{m_{ref}, pair_{score}\}(3)
         else
         | Nop
         end
    end
    // Associate m_{in} to m_{ref} and save corresponding score
    \operatorname{pair}[m_{in}] = \operatorname{tmp\_best\_ref}
end
Compute final score using local scores in pair
                           Algorithm 2: Pairing phase
```

### 2.3 Assumptions on the matching system

The studied biometric matching system structure is compliant to the one pictured in Figure 1 but we can additionally make the following assumptions<sup>4</sup> 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;
- The final score is outputted by the system;
- There is no protection of the implementation:
  - There are no side channel countermeasures;
  - There is no retry counter.

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

<sup>&</sup>lt;sup>4</sup> Note that the scope here is not to discuss the security of any existing Match-On-Card products.

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 Figure 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 fingerprint.

#### 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.

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



Fig. 3. Information leakage on pairing step

consumption due to the process activity inequality. This difference can be seen on the power consumption trace of the registration part (Figure 4). Figure 5 shows the angle values of all the reference fingerprint minutiae.



Fig. 4. Power consumption during registration with a single minutia input fingerprint



Fig. 5. Reference fingerprint minutiae angle values

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 Figure 5). 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 matching with an increasing angle value and kept the trace for each match. Figure 6 shows the traces of 3 different matchings with an increasing angle value (not consecutive).



Fig. 6. Shifting drops in power consumption with 3 increasing input minutia angle values

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 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 on Figure 7, 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 Exploitation

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

# 4.1 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 ( $\approx 20$  to 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



Fig. 7. Comparison between the number of minutiae neighbors in reference fingerprint and the value of drop shift

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 [19] used the Hill Climbing (HC) heuristic to find modifications that increase the comparison score between synthetic minutiae set 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). These two conditions are verified in our case following the assumptions explained in Section 2.3.

#### 4.2 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 Section 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 [19] 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.



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

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 passes the threshold score at 4000 iterations instead of 8000 iterations for the other two modes. To achieve this improvement, we needed only 64 matching executions, which can be added to the 4000 iterations to give the total number of needed matching executions in order to construct an approximation of the reference fingerprint data.

Keeping the assumptions from Section 2.3 verified, we are able to succeed a Hill Climbing with half the matching iterations otherwise needed. This means that some specific countermeasures have to be implemented to protect the biometric comparison coprocessor from that kind of leakages.

### 5 Countermeasures

The previous part shows that the most interesting leakage information comes from the sorted-by-angle parse on reference minutiae during the registration step. To get rid of these angles dependencies would make the HC improvement not feasible anymore. In this section, we will describe one countermeasure for each threat previously identified.

#### 5.1 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 pretranslation-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  $\Delta_{\theta_{min}} \dots \Delta_{\theta_{max}}$ .  $rot_a \oplus mask$  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!.

#### 5.2 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.

#### 5.3 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 Figure 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.

## 6 Conclusion

In this paper, we analyzed the potential information leakages of a hardware biometric comparison module. We pointed out that we can find out relevant information of the stored fingerprint template by the means of side channel analysis. These informations can be used to facilitate hill climbing attacks. Fortunately, there are some simple countermeasures that can be used to thwart the information leakages. Our future work will investigate further the possible side channel analysis strategies and will also cover the design study of a secure biometric coprocessor by including such kind of countermeasures.

## 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.
- C. Barral and S. Vaudenay. A protection scheme for moc-enabled smart cards. In Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the, pages 1–6. IEEE, 2006.

- S. Bistarelli, F. Santini, and A. Vaccarelli. An asymmetric fingerprint matching algorithm for java card <sup>tm</sup>. *Pattern Anal. Appl.*, 9(4):359–376, 2006.
- T. Chouta, J.-L. Danger, L. Sauvage, and T. Graba. A small and high-performance coprocessor for fingerprint match-on-card. In DSD, pages 915–922. IEEE, 2012.
- T. Cucinotta, R. Brigo, and M. D. Natale. Hybrid fingerprint matching on programmable smart cards. In S. K. Katsikas, J. Lopez, and G. Pernul, editors, *Trust-Bus*, volume 3184 of *Lecture Notes in Computer Science*, pages 232–241. Springer, 2004.
- J. Galbally, S. Carballo, J. Fiérrez, and J. Ortega-Garcia. Vulnerability assessment of fingerprint matching based on time analysis. In J. Fiérrez-Aguilar, J. Ortega-Garcia, A. Esposito, A. Drygajlo, and M. Faúndez-Zanuy, editors, COST 2101/2102 Conference, volume 5707 of Lecture Notes in Computer Science, pages 285–292. Springer, 2009.
- M. Govan and T. Buggy. A computationally efficient fingerprint matching algorithm for implementation on smartcards. In *Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007. First IEEE International Conference on*, pages 1–6. IEEE, 2007.
- P. C. Kocher, J. Jaffe, and B. Jun. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 1996. (PDF).
- P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
- M. Martinez-Diaz, J. Fierrez-Aguilar, F. Alonso-Fernandez, J. Ortega-Garcia, and J. Siguenza. Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification. In *Carnahan Conferences Security Technology, Proceedings 2006 40th Annual IEEE International*, pages 151–159. IEEE, 2006.
- J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smard Cards. In I. Attali and T. P. Jensen, editors, *E-smart*, volume 1240 of *LNCS*, pages 200–210. Springer-Verlag, 2001. ISSN 0302-9743.
- N. K. Ratha, J. H. Connell, and R. M. Bolle. An analysis of minutiae matching strength. In Proc. 3rd AVBPA, pages 223–228, 2001.
- J. Reisman, U. Uludag, and A. Ross. Secure fingerprint matching with external registration. In T. Kanade, A. K. Jain, and N. K. Ratha, editors, *AVBPA*, volume 3546 of *Lecture Notes in Computer Science*, pages 720–729. Springer, 2005.
- K. Tiri, D. Hwang, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede. Aes-based cryptographic and biometric security coprocessor ic in 0.18mu;m cmos resistant to side-channel power analysis attacks. In VLSI Circuits, 2005. Digest of Technical Papers. 2005 Symposium on, pages 216–219, 2005.
- 18. UCLA. Thumbpod: a next generation biometrically secure wireless embedded system. http://www.emsec.ee.ucla.edu/thumbpod.
- U. Uludag and A. K. Jain. Attacks on biometric systems: a case study in fingerprints. In E. J. Delp and P. W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents*, volume 5306 of *Proceedings of SPIE*, pages 622–633. SPIE, 2004.
- S. Yang, K. Sakiyama, and I. Verbauwhede. Efficient and secure fingerprint verification for embedded devices. *EURASIP Journal on Advances in Signal Processing*, 2006(1):058263, 2006.
- S. Yang and I. Verbauwhede. Automatic secure fingerprint verification system based on fuzzy vault scheme. In *IEEE International Conference on Acoustics*, Speech, and Signal Processing (ICASSP 2005), pages 609–612, 2005.