Externalized Fingerprint Matching

Claude Barral
Gemplus Card International
Applied Research & Security Centre
La Vigie. Avenue des Jujubiers
La Ciotat, F-13705, France

Jean-Sébastien Coron, David Naccache
Gemplus Card International
Applied Research & Security Centre
34 rue Guynemer
Issy les Moulineaux CEDEX, F-92447, France

{claude.barral,jean-sebastien.coron,david.naccache}@gemplus.com

Abstract. The 9/11 tragedy triggered an increased interest in biometric passports. According to several sources [2], the electronic ID market is expected to increase by more than 50% *per annum* over the three coming years, excluding China.

To cost-effectively address this foreseen explosion, a very inexpensive memory card (phonecard-like card) capable of performing fingerprint matching is paramount.

This paper presents such a solution. The proposed protocol is based on the following idea: the card stores the user's fingerprint information to which random minutiae were added at enrolment time (we denote this scrambled template by t). The card also stores a binary string w encoding which of the minutiae in t actually belong to the holder. When an identification session starts, the terminal reads t from the card and, based upon the incoming scanner data, determines which of the minutiae in t are genuine. The terminal forms a candidate w' and sends it to the card. All the card needs to do is test that the Hamming weight of $w \oplus w'$ is smaller than a security threshold d.

It follows that the card only needs to embark passive data storage capabilities, one exclusive-or gate, a shift register, a counter and a comparator (less than 40 logical gates).

1 Introduction

Since the 9/11 tragedy fingerprints have rallied significant support as the biometric technology that will probably be most widely used in the future.

The fingerprint's strength is its acceptance, convenience and reliability. It takes little time and effort for somebody using a fingerprint identification device to have his or her fingerprint scanned. Studies have also found that using fingerprints as an identification means is the least intrusive of all biometric techniques. Verification of fingerprints is also fast and reliable. Users experience fewer errors in matching when they use fingerprints versus many other biometric methods. In addition, fingerprint identification devices usually require very little space on a desktop or in a machine. Several companies have produced capture units (scanners) smaller than a deck of cards.

Generally, a fingerprint biometric system comprises four main modules:

- A capture unit, which acquires the raw biometric fingerprint data D of an individual (typically a bitmap of the finger's ridges).

- A feature extraction module f in which the acquired biometric data is processed to extract a feature-set $f(D)$ that models D. Typically $f(D)$ is the position and orientation of ridge bifurcations and ridge endings in D (points called *minutiae*, see figures 1 and 2). $f(D)$ is usually obtained after several signal processing steps (Figure 3) consisting in filtering D, thinning it and extracting minutiae from the thinned image using an *ad-hoc* algorithm.

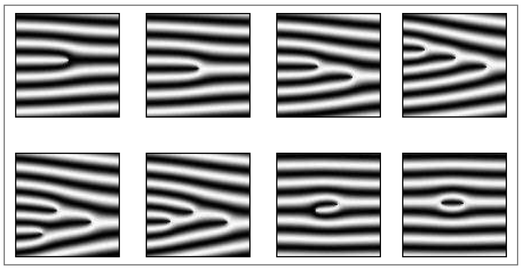

Figure 1. Different Minutia Types

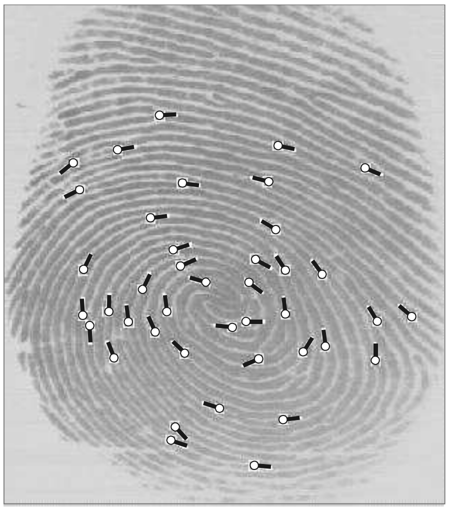

Figure 2. Minutiae in a Fingerprint

From a practical standpoint, a raw D requires 64K bytes[1]. The complexity of f varies greatly according the algorithm used. 120 MIPS per fingerprint is a typical benchmark figure for f. The size of $f(D)$ is typically comprised between 300 and 3000 bytes.

- A matching module μ in which an extracted feature-set $f(A)$ can be compared to a reference pattern $f(B)$. This comparison process (figure 4) outputs a score $0 \leq \mu(f(A), f(B)) \leq 1$.
- A decision-making module in which the user's claimed identity is either accepted or rejected based on the matching score: if $\mu(f(A), f(B)) > \alpha$ return `accept` else return `reject`. α is an application-dependent security parameter.

[1] 500 dpi resolution, 256×256 pixel image 256 grey-scale (*i.e.* one byte per pixel).

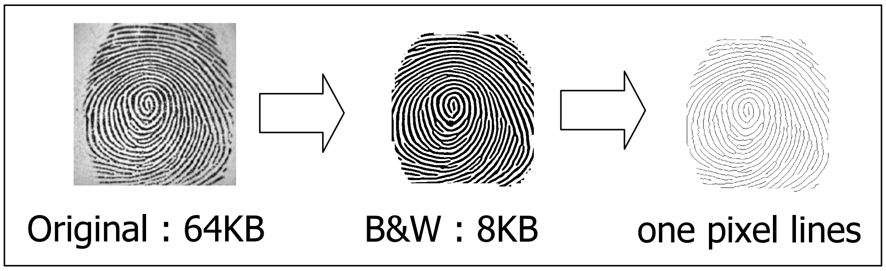

Figure 3. Processing a Fingerprint Bitmap

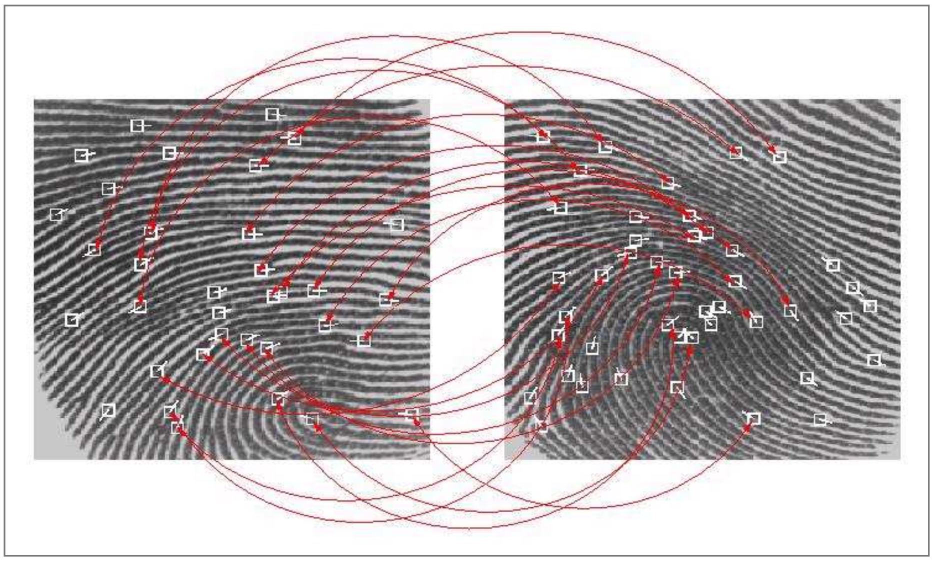

Figure 4. Fingerprint Matching

Traditionally, the performance of a biometric system is described by the probability distributions of genuine and impostor matching scores. A genuine matching score is obtained when two feature sets corresponding to the same individual are compared, and an impostor matching score is obtained when feature sets belonging to two different individuals are compared. When a matching score exceeds α, the two feature sets are declared as belonging to the same individual; otherwise, they are assumed to belong to two different individuals.

Thus, there are two types of errors associated with a biometric system:

– A false accept, which occurs when an impostor matching score happens to exceed α.

– A false reject, which occurs when a genuine matching score doesn't exceed α.

A Receiver Operating Characteristic (ROC) curve plots the False Reject Rate (FRR - the percentage of genuine scores that do not exceed α) against the False Accept Rate (FAR - the percentage of impostor scores that exceed α) for different α values. The α that best suits a system depends on the nature of the application. In forensic applications, for example, a low FRR is preferred, while for access to facilities such as nuclear plants, a low FAR is desired.

2 The Matching Problem

Minutiae matching is certainly the best known and most widely used method for fingerprint matching. We refer the reader to [3] for a definition of the matching problem that we recall here:

2.1 Problem formulation

Let $f(D)$ and $f(D')$ be the representation of the template and input fingerprint, respectively. Here the representation f is a feature vector (of variable length) whose elements are the fingerprint minutiae. Each minutia may be described by a number of attributes, including its location in the fingerprint image, orientation, type (*e.g.* ridge termination or ridge bifurcation), a weight based on the quality of the fingerprint image in the neighborhood of the minutia, and so on. Most common minutiae matching algorithms consider each minutia m as a triplet $\{x, y, \theta\}$ that indicates the x, y minutia location coordinates and the minutia angle θ:

$$\begin{aligned} f(D) &= \{m_1, m_2, \cdots, m_n\}, \quad & m_i &= \{x_i, y_i, \theta_i\}, \quad & i &= 1 \cdots, n \\ f(D') &= \{m'_1, m'_2, \cdots, m'_{n'}\}, \quad & m'_i &= \{x'_i, y'_i, \theta'_i\}, \quad & i &= 1 \cdots, n' \end{aligned}$$

where n and n' denote the number of minutiae in $f(D)$ and $f(D')$, respectively.

A minutia $m'_j \in f(D')$ and a minutia $m_i \in f(D)$ are considered matching, if the *spatial distance* (sd) between them is smaller than a given tolerance r_0 and the *direction difference* (dd) between them is smaller than an angular tolerance θ_0:

$$\begin{aligned} \mathrm{sd}(m'_j, m_i) &= \sqrt{(x'_j - x_i)^2 + (y'_j - y_i)^2} \leq r_0 \quad (1) \\ \text{and} \\ \mathrm{dd}(m'_j, m_i) &= \min(|\theta'_j - \theta_i|, 360° - |\theta'_j - \theta_i|) \leq \theta_0 \quad (2) \end{aligned}$$

Equation (2) takes the minimum of $|\theta'_j - \theta_i|$ and $360° - |\theta'_j - \theta_i|$ because of the circularity of angles (the difference between angles of 2° and 358° is only 4°). The *tolerance boxes* (or hyper-spheres) defined by r_0 and θ_0 are necessary to compensate for the unavoidable errors made by feature extraction algorithms and to account for the small plastic distortions that cause the minutiae positions to change.

Aligning the two fingerprints is a mandatory step in order to maximize the number of matching minutiae. Correctly aligning two fingerprints requires *displacement* (in x and y) and *rotation* (θ) to be recovered, and frequently involves other geometrical transformations:

- *scale* has to be considered when the resolution of the two fingerprints may vary (*e.g.* the two fingerprint images have been taken by scanners operating at different resolutions);
- other *distortion-tolerant* geometrical transformations could be useful to match minutiae in case one or both of the fingerprints is affected by severe distortions.

In any case, tolerating a higher number of transformations results in additional degrees of freedom to the minutiae matcher: when a matcher is designed, this issue needs to be carefully evaluated, as each degree of freedom results in a huge number of new possible alignments which significantly increases the chance of incorrectly matching two fingerprints from different fingers.

Let map(.) be the function that maps a minutia $m'_j \in f(D')$ into m''_j according to a given geometrical transformation; for example, by considering a displacement of $[\Delta x, \Delta y]$ and a counterclockwise rotation θ around the origin[2]:

$$\mathrm{map}_{\Delta x, \Delta y, \theta}(m'_j) = m''_j = \{x''_j, y''_j, \theta'_j + \theta\}$$

where

[2] The origin is usually selected as the minutiae centroid (*i.e.* the average point); before the matching step, minutiae coordinates are adjusted by subtracting the centroid coordinates.

$$\begin{bmatrix} x''_j \\ y''_j \end{bmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{bmatrix} x'_j \\ y'_j \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Let $\zeta(.)$ be an indicator function that returns 1 when the minutiae m''_j and m_i match according to Equations (1) and (2):

$$\zeta(m''_j, m_i) = \begin{cases} 1 & \text{if} \quad \mathrm{sd}(m''_j, m_i) \leq r_0 \quad \text{and} \quad \mathrm{dd}(m''_j, m_i) \leq \theta_0 \\ 0 & \text{otherwise} \end{cases}$$

The the matching problem can be formulated as:

$$\underset{\Delta x, \Delta y, \theta, P}{\text{maximize}} \sum_{i=1}^{n} \zeta(\mathrm{map}_{\Delta x, \Delta y, \theta}(m'_{P(i)}), m_i) \qquad (3)$$

where $P(i)$ is an unknown function that determines the pairing between $f(D)$ and $f(D')$ minutiae; in particular, each minutia has either exactly one mate in the other fingerprint or has no mate at all:

1. $P(i) = j$ indicates that the mate of the $m_i \in f(D)$ is $m'_j \in f(D')$
2. $P(i) = \perp$ indicates that $m_i \in f(D)$ has no mate in $f(D')$
3. an $m'_j \in f(D')$ such that $\forall i = 1, \cdots, n \quad P(i) \neq j$ has no mate in $f(D)$
4. $\forall i = 1, \cdots, n \;\; \forall k = 1, \cdots, n' \;\Rightarrow P(i) \neq P(k)$ or $P(i) = P(k) = \perp$ (this requires that each minutia in $f(D')$ is associated with at most one minutia in $f(D)$).

Note that, in general, $P(i) = j$ does not necessarily mean that minutiae m'_j and m_i match in the sense of Equations (1) and (2) but only that they are the most likely pair under the current transformation.

Expression (3) requires that the number of minutiae mates be maximized, independently of how strict these mates are; in other words, if two minutiae comply with Equations (1) and (2), then their contribution to expression (3) is made independently of their spatial distance and of their direction difference.

Solving the minutiae matching problem (expression (3)) is trivial when the correct alignment $(\Delta x, \Delta y, \theta)$ is known; in fact, the pairing (*i.e.* the function P) can be determined by setting for each $i = 1, \cdots, n$:

- $P(i) = j$ if $m''_j = \mathrm{map}_{\Delta x, \Delta y, \theta}(m'_j)$ is closest to m_i among the minutiae.

$$\left\{ m''_k = \mathrm{map}_{\Delta x, \Delta y, \theta}(m'_k) \mid k = 1, \cdots, n, \quad \zeta(m''_k, m_i) = 1 \right\}$$

– $P(i) = \perp$ if $\forall k = 1, \cdots, n, \quad \zeta(\text{map}_{\Delta x, \Delta y, \theta}(m'_k), m_i) = 0$

To comply with constraint 4 above, each minutia m''_j already mated has to be marked, to avoid mating it twice or more. Figure 5 shows an example of minutiae pairing given a fingerprint alignment.

To achieve the optimum pairing (according to Equation (3)), a slightly more complicated scheme should be adopted: in fact, in the case when a minutia of $f(D')$ falls within the tolerance hyper-sphere of more than one minutia of $f(D)$, the optimum assignment is that which maximizes the number of mates (refer to Figure 6 for a simple example).

The maximization in (3) can be easily solved if the function P (minutiae correspondence) is known; in this case, the unknown alignment $(\Delta x, \Delta y, \theta)$ can be determined in the least square sense. Unfortunately, in practice, neither the alignment parameters nor the correspondence function P are known and therefore, solving the matching problem is very hard. A brute force approach, that is, evaluating all the possible solutions (correspondences and alignments) is prohibitive as the number of possible solutions is exponential in the number of minutiae (the function P is more than a permutation due to the possible $\perp$ values). Hence heuristics are used.

In figure 5 minutiae of $f(D')$ mapped into $f(D)$ coordinates for a given alignment. Minutiae of $f(D)$ are denoted by $\odot$s, whereas $f(D')$ minutiae are denoted by $\times$s. Note that $f(D')$ minutiae are referred to as m'', because what is shown in the figure is their mapping into $f(D)$ coordinates. Pairing is performed according to the minimum distance. The dashed circles indicate the maximum spatial distance. The gray circles denote successfully mated minutiae; minutia m_1 of $f(D)$ and minutia m''_3 of $f(D')$ have no mates, minutiae m_3 and m''_6 cannot be mated due to their large direction difference.

In figure 6, if m_1 were mated with m''_2 (the closest minutia), m_2 would remain unmated; however, pairing m_1 with m''_1, allows m_2 to be mated with m''_2, thus maximizing Equation (3).

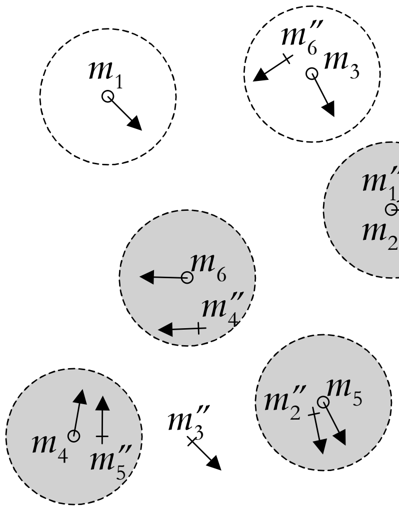

Figure 5. Mating

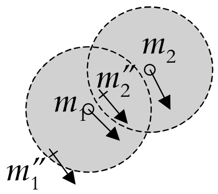

Figure 6. Mating with a Second-Closest

3 Fingerprint Match-On-Card

3.1 What Is a Smart-Card?

The physical support of a conventional smart-card is a plastic rectangle printed with information concerning the application or the issuer, as well as readable information about the card holder (for instance, a validity date or a photograph). This support can also carry a magnetic stripe or a bar-code.

ISO Standard 7816 specifies that the micromodule must contain an array of eight contacts but only six of these are actually connected to the chip, which is usually not visible. The contacts are assigned to power supplies (V_{cc} and V_{pp}), ground, clock, reset and a serial data communication link commonly called I/O. ISO is currently considering various requests for re-specification of the contacts; notably for dual USB/7816 support.

While for the time being card CPUs are mainly 8 or 16-bit microcontrollers[3] new 32-bit devices has recently become available.

From a functional standpoint a smart card is a miniature computer. A small on-board RAM serves as a temporary storage of calculation results and the card's microprocessor executes a program etched into the card's ROM at the mask-producing stage. This program cannot be modified or read-back in any way.

For storing user-specific data individual to each card, cards contain EEPROM (Electrically Erasable and Programmable ROM) or flash memory, which can be written and erased hundreds of thousands of times. Java cards even allow the import of executable programs (applets) into their nonvolatile memory according to the card holder's needs.

Finally, the card contains a communication port (serial via an asynchronous link) for exchanging data and control information with the external world. A common bit rate is 9,600 bits per second, but much faster ISO-compliant throughputs are commonly used (from 19,200 up to 115,200 bits per second). The advent of USB cards opens new horizons and allows data throughput to easily reach one megabit per second.

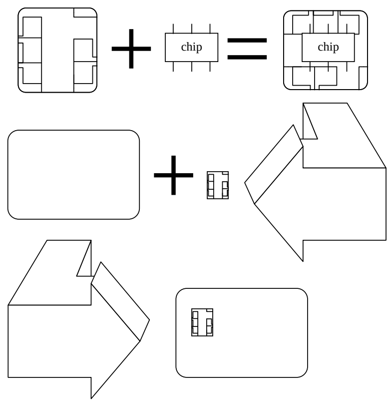

Figure 7. Smart-Card Manufacturing

To prevent information probing, all these elements are packed into one single chip. If this is not done, the wires linking the system components to each another could become potential passive or active penetration routes [1]. The different steps of smart card manufacturing are shown in figure 7: wire bonding (chip + micromodule) and potting (chip + micromodule + plastic).

[3] The most common cores are Motorola's 68HC05 and Intel's 80C51.

3.2 Biometric Smart-Cards

Biometric smart-cards has the capacity to store a template $f(D)$ in EEPROM and perform both matching and decision-making when presented with a candidate D' (or $f(D')$ if the algorithm f is public[4]).

Typically, an accept will 'open' the card and permit access to some of its EEPROM files, enable the generation of a digital signature or debit a purse.

It is customary to require that these steps take place in less than a second (convenience). When coded in a card a matching algorithm would use at least 2,000 bytes of RAM. Code would usually occupy 2,000 to 12,000 ROM bytes.

Code complexity (matching involves many floating-point trigonometric operations) and RAM consumption are two decisive cost factors in the design of such solutions.

The following section provides a novel solution to this problem. The solution, called *Externalized Fingerprint Matching*, allows to implement μ in simple (microprocessor-less) memory cards. This is particularly important for addressing cost-effectively very large markets (*e.g.* China, 1.3 billion inhabitants) and for deploying disposable biometric IDs such as visas, hotel room keys or visitor/subcontractor badges.

4 Externalizing the Fingerprint Matching

The new idea consists in adding *false minutiae* to $f(D)$ and *reversing the burden of proof* to have the *card challenge the reader* to find out, based on the acquisition coming from the scanner, which minutiae are genuine:

4.1 Enrolment

The enrolment protocol is the following:

1. The issuer extracts $f(D)$, picks a set of random minutiae r and merges it into $f(D)$. We denote the result of this operation (illustrated in figures 8 and 9) by $t = f(D) \cup r$.

[4] Most match-on-card algorithms are proprietary but usually available under NDA. The following companies sell, license or develop match-on-card code: Precise biometrics, Veridicom, Gemplus, Siemens biometrics, Ikendi, Dermalog, Identix, Fingerprint cards AB.

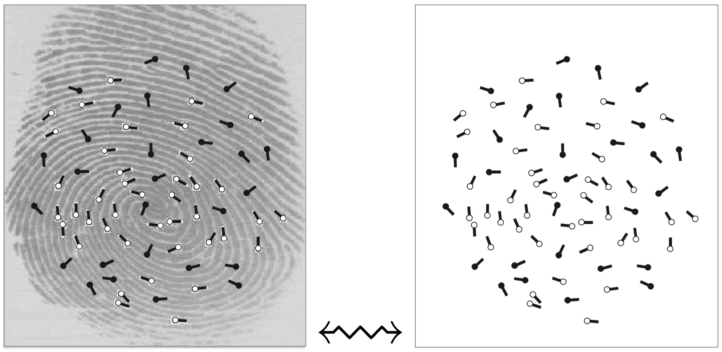

Figure 8. Fingerprint Scrambling with False Minutiae

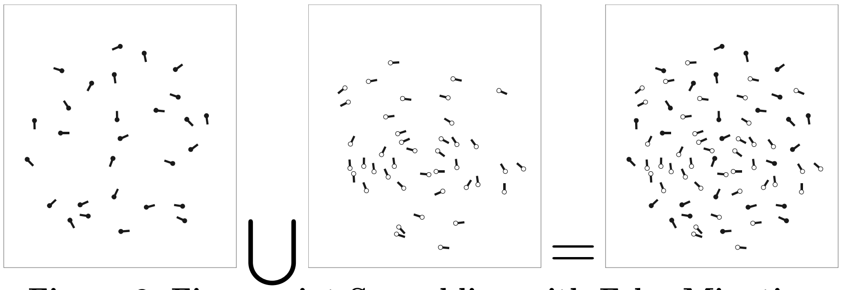

Figure 9. Fingerprint Scrambling with False Minutiae

2. The issuer encodes t as a binary string u where bit $u_i = 1$ if the i-th minutia in t belongs to $f(D)$ and $u_i = 0$ if the i-th minutia in t belongs to r.
3. The issuer signs, using a public-key signature scheme the data $\{t, u, d\}$ where d is a security parameter which choice is discussed below. Let σ be the signature of $\{t, u, d\}$.
4. The issuer delivers an identity card containing $\{t, u, d, \sigma\}$. The card allows the free reading of t and d.

4.2 Identification

The identification protocol is the following:

1. The terminal receives from the scanner a fingerprint candidate D'.
2. The terminal reads t from the card and partitions t (based upon D') into two sets $t = t_{\text{true}} \cup t_{\text{false}}$. The terminal encodes this partition as a binary string u' where bit $u'_i = 1$ if the i-th minutia in t belongs to t_{true} and $u'_i = 0$ if the i-th minutia in t belongs to t_{false}. The terminal sends u' to the card.

3. The card computes $w = u \oplus u'$. If the Hamming weight of w (denoted $H(w)$) is smaller than d the card outputs σ and u. At this step the card considers that the presented finger is legitimate.
4. The terminal verifies σ with respect to the issuer's public-key and if σ is correct and $H(u \oplus u') \leq d$ then the terminal considers that the scanned finger and the card match each other and are approved by the issuer.

4.3 Evaluating the Protocol's FAR

The security of the above protocol is determined as follows.

The correct fingerprint is characterized by the reference vector u whose length is $n+m$ and whose Hamming weight is n. Since u is unknown to the attacker we assume that it has a random distribution over the vectors of weight n.

Assume that the Hamming weight of u', the vector submitted by the attacker, is equal to $n+k$, where k is a non-negative integer. Letting $w = u' \oplus u$ we have $w = w_1 \vee w_2$ where $w_1 = u \wedge \neg u'$ and $w_2 = \neg u \wedge u'$. Let $i = H(w_1)$.

We have $H(u') = n+k = H(u)+H(w_2)-H(w_1)$, which gives $H(w_2) = i+k$, whereby $H(w) = H(w_1) + H(w_2) = 2i + k$. Since $H(u') = n+k$, the number of possible choices for w_2 is $\binom{n+k}{i+k}$ and the number of possible choices for w_1 is $\binom{m-k}{i}$.

The number of possible u vectors for a given integer i is therefore:

$$R(n,m,k,i) = \binom{n+k}{i+k} \times \binom{m-k}{i}$$

Summing over all possible i, we obtain the probability over u, denoted $P(n,m,k)$ that the attack succeeds with a candidate u' of weight $n+k$:

$$P(n,m,k) = \frac{\sum_{i=0}^{(d-k)/2} R(n,m,k,i)}{\binom{m+n}{n}}$$

If k is negative, we obtain the probability:

$$P(n,m,k) = \frac{\sum_{i=0}^{(d+k)/2} R'(n,m,k,i)}{\binom{m+n}{n}}$$

Where:

$$R'(n,m,k) = \binom{n+k}{i} \times \binom{m-k}{i-k}$$

Eventually, the FAR is the maximum probability, over all possible k, that a candidate u' is accepted:

$$\text{FAR} = \max_{k=-n}^{m} P(n,m,k)$$

Letting $\text{FAR} = 10^{-e}$ typical $\{n,m\}$ values for $e=5$ and $d=0$ would be: $\{6,17\}, \{7,14\}, \{8,12\}, \{9,11\}, \{10,10\}, \{11,9\}$. Variations in d affect the FAR as shown in the graphics below:

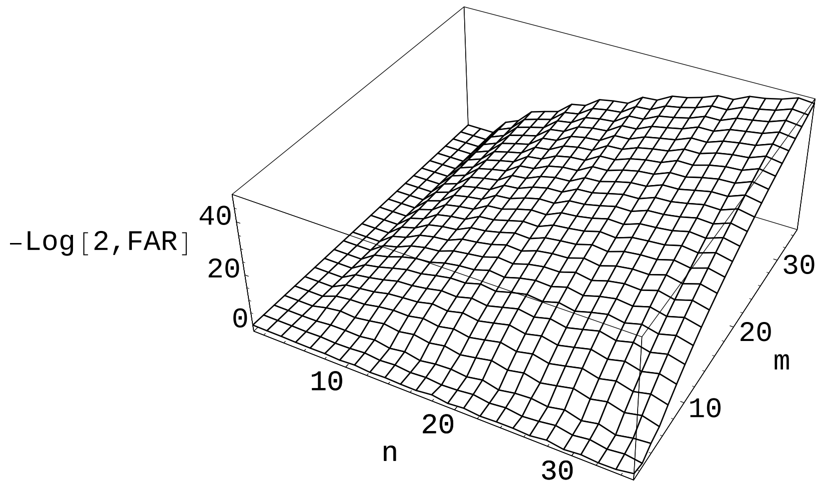

Figure 10. FAR for $d = 4$.

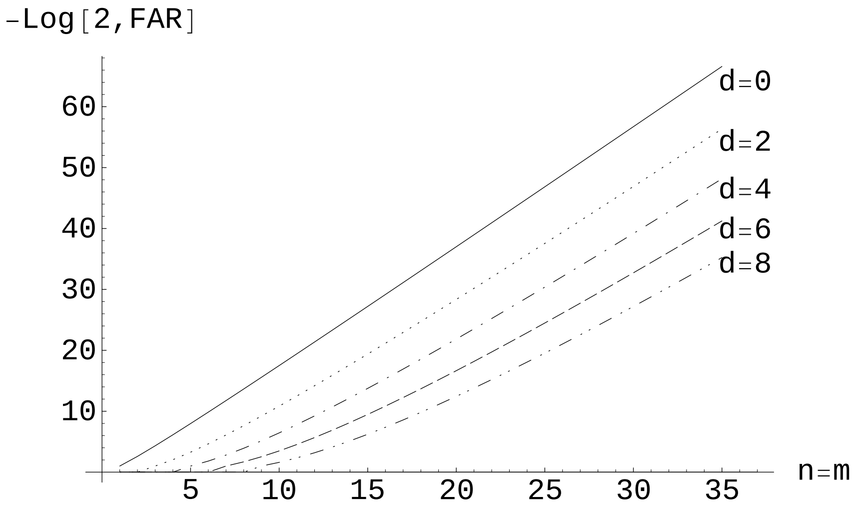

Figure 11. FAR for $m = n$ and different d values.

Note that the above calculations rely on the following two assumptions:

Assumption 1. Spatial Uniformity Assumption: *The probability to find a minutia at any given $\{x, y\}$ coordinate in $f(D)$ is constant.*

In other words, the simplified FAR estimate assumes[5] that there are no denser or scarcer areas in $f(D)$ and that minutiae are independent of each other *i.e.*, knowing that a minutia m exists at a given $\{x, y\}$ location does not provide any information about the would-be existence (or type) of minutiae at m's neighborhood.

Assumption 2. Biometric Scrambling Assumption: There exists an probabilistic algorithm $\mathcal{A}$ taking as input $f(D)$ and outputting a $t = f(D) \cup r$ such that partitioning t into the original subsets $f(D)$ and r, even approximately, is intractable.

An alternative fingerprint scrambling model is given in the appendix.

5 Implementation and Applications

The protocol was implemented as a Javacard applet on a Gemplus GemXpresso Pro smart card using Ikendi Software AG's minutiae extraction engine.

[5] *"... Since u is unknown to the attacker we assume that it has a random distribution over the vectors of weight n...*

For the reader terminal emulation, the demonstrator is uses a Pentium III at 500 MHz and a Gemplus GemPC Touch 430 smart card reader with an embedded fingerprint sensor (silicon sensor using capacitive technology), shown below.

Figure 12. GemXpresso Pro and GemPC Touch 430.

The entire fingerprint matching process takes less than a second. The applet's size is about 510 bytes and the card's processing time is 26 ms, that break-down as follows:

Protocol phase	**Duration**
The terminal asks permission to send u'	6 ms
The card prepares to receive u'	8 ms
The terminal sends u'	6 ms
The card compares u and u'	4 ms
The card returns true or false	2 ms

5.1 National Identity and Access Control to Facilities

In a typical national ID application, a law enforcement agent using a portable secret-less biometric reader must ascertain that a physically present individual is associated to a data string Q. In most cases, Q represents information such as the ID card number, the surname, given names, nationality, height, place of birth, date of birth, dates of issue and expiry, color of eyes, residence *etc.* In the sequel we assume that σ also signs Q

Given that the portable reader is under the agent's total control (*i.e.* provides end-to-end security from the capture unit to the decision taking and display module) the display of Q on the reader's screen provides the officer with a binding between the physically present individual and Q.

Note (as is the case with *all other* match-on-card protocols) that biometry alone cannot provide a binding between the ID (physical support) and the individual but only between Q (the information) and the individual. To provide *also* a binding between the ID and the individual the ID must be enriched with active digital signature or zero-knowledge capabilities.

5.2 Internet Login and On-Line Access

As is the case with passwords and other biometric protocols, one *cannot* require resistance against parties who witnessed a successful biometric identification session (or participated in it). However, we do require that a party who never witnessed a successful session will be unable to mimic the legitimate user and his card, even under the (very adversarial) assumption that the attacker managed to steal the user's ID card.

Given that the card will only reveal w when the interrogator proves to it that he knows already an extremely close approximation of w, the thief will not be able to retrieve $f(D)$ from the ID card [6] and mimic the user's presence on the other side of the communication line.

As is the case with passwords and other biometric protocols, a remote user can always voluntarily 'delegate' (lend) his D and σ to friends or colleagues. To prevent this and ascertain that the ID card is physically present at the other side of the line, the ID card must be enriched with active public-key capabilities. Note that even such a capability will never ascertain that the user did not voluntarily give the physical ID plus D to the friend or the colleague.

5.3 Access Control to Card Inner Data or Card Functions

In many settings, one wishes to bind the enabling of an *on-card* function to the user's presence. A typical example is an electronic purse where a debit function is activated only after successfully recognizing the user's D. This provides an excellent protection against card theft and subsequent illegal debit.

Note that unless the capture unit is embedded in the card (such capture units are marketed by several suppliers today), a user can, again, voluntarily lend his D to a friend (although in most cases debit operations are done in front of merchants). Other applications of access control to inner card data consist in accessing private files on a memory stick or medical data in a health cards.

[6] Should the FAR be high (*e.g.* $\simeq 2^{-40}$), we recommend to protect the card against exhaustive search using a ratification counter. *e.g* lock the card after 20 successive unsuccessful fingerprint verifications.

5.4 Conclusion

The above shows that although extremely economic (from an on-board resource consumption perspective), the protocol presented in this paper provides equivalent functionalities to other match-on-card techniques in all typical use-cases.

References

1. O. Kommerling and M. Kuhn, *Design principles for tamper-resistant smartcard processors*, Proceedings of USENIX Workshop on Smartcard Technology, 1999, pp. 9–20.
2. A. Robert, La Tribune, Les cartes à puce se cherchent une identité, page 59b, October 10, 2003.
3. D. Maltoni, D. Maio, A. Jain, S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer, New York, 2003.

APPENDIX A
SIMPLIFIED MINUTIAE SCRAMBLING MODEL

In the simplified minutiae scrambling model, we let n be the number of minutiae in $f(D)$ and $k \leq n$ be the total number of minutiae in the resulting template t, that is, true and false minutiae. Again, we let d be a security parameter which choice is discussed below.

Definition 1. *A Biometric Scrambling Algorithm is an algorithm taking as input a set $f(D)$ of n minutiae and outputting a template t and a randomly distributed k-bit string u, such that the i-th minutia in t belongs to $f(D)$ iff $u_i = 1$.*

Assumption 3. Simplified Assumption: There exists an (efficient) Biometric Scrambling algorithm $\mathcal{A}$ with the following two properties:
- Given $f(D)$ and a t generated by $\mathcal{A}$, one can obtain a u' such that $H(u' \oplus u) \leq d$ with probability greater than β.
- Given only t the success probability of an attacker in outputting u' such that $H(u' \oplus u) \leq d$ is $\epsilon_{\text{guess}} + \mathtt{negl}$, where $\mathtt{negl}$ is a negligible function of the parameters, and

$$\epsilon_{\text{guess}} = 2^{-k} \sum_{i=0}^{d} \binom{k}{i}$$

A.1. Enrolment

The enrolment protocol is the following:

1. The issuer extracts $f(D)$ and sets $t \leftarrow \{\}$.
2. The issuer generates a random k-bit string $u = u_1, \ldots, u_k$.
3. For $i = 1$ to k:
   (a) if $u_i = 1$, the issuer adds to the template t a random (and not already selected) minutia from $f(D)$.
   (b) if $u_i = 0$, the issuer adds to t a random minutia.
4. The issuer signs the data $\{t, u, d\}$. Let σ be the signature of $\{t, u, d\}$.
5. The issuer delivers an identity card containing $\{t, u, d, \sigma\}$. The card allows the free reading of t and d.

Identification is identical to 4.2.

A.2. Security

The second property of the Simplified Assumption ensures that the success probability of an attacker is only negligibly greater than the success probability obtained by just randomly "guessing" the random string u.

The following theorem proves that the identification protocol is secure under the Simplified Assumption.

Theorem 1. *Without the knowledge of $f(D)$ an attacker's success probability is smaller than $\epsilon_{guess} + \texttt{negl}$.*

Proof. Assume that an attacker $\mathcal{F}$ manages to pass the identification protocol without knowing $f(D)$, with probability ϵ'. Then, using $\mathcal{F}$, given a template t, one can obtain u' such that $H(u' \oplus u) \leq d$, without knowing d, with probability ϵ'. By the Simplified Assumption, this can only be done with probability lesser than $\epsilon_{\text{guess}} + \texttt{negl}$, which gives $\epsilon' \leq \epsilon_{\text{guess}} + \texttt{negl}$. □

A.3. Evaluating the FAR and FRR

The FAR is, by definition, the probability that a wrong fingerprint is declared genuine. Therefore, the FAR is smaller than the attacker's success probability :

$$\text{FAR} \leq 2^{-k} \sum_{i=0}^{d} \binom{k}{i} + \texttt{negl}$$

Neglecting the term $\texttt{negl}$, the following table lists various $\{k, d\}$ choices and their corresponding FARs.

$-\log_{10}(\text{FAR})$	2	3	3	4
k	10	20	26	30
d	2	3	5	5

The FRR being the percentage of correct fingerprints that do not pass the identification algorithm, the FRR is equal to $1 - \beta$, where β in the probability introduced in the Simplified Assumption. $1 - \beta$ must be small.