

# Secure Biometric Authentication for Weak Computational Devices

Mikhail J. Atallah<sup>1</sup>, Keith B. Frikken<sup>1</sup>  
Michael T. Goodrich<sup>2</sup>, and Roberto Tamassia<sup>3</sup>

<sup>1</sup> CERIAS and Department of Computer Sciences, Purdue University  
mja@cs.purdue.edu, kbf@cs.purdue.edu

<sup>2</sup> Dept. of Computer Science, Univ. of California, Irvine  
goodrich@ieee.org

<sup>3</sup> Dept. of Computer Science, Brown Univ.  
rt@cs.brown.edu

**Abstract.** This paper presents computationally “lightweight” schemes for performing biometric authentication that carry out the comparison stage without revealing any information that can later be used to impersonate the user (or reveal personal biometric information). Unlike some previous computationally expensive schemes—which make use of the slower cryptographic primitives—this paper presents methods that are particularly suited to financial institutions that authenticate users with biometric smartcards, sensors, and other computationally limited devices. In our schemes, the client and server need only perform cryptographic hash computations on the feature vectors, and do not perform any expensive digital signatures or public-key encryption operations. In fact, the schemes we present have properties that make them appealing even in a framework of powerful devices capable of public-key signatures and encryptions. Our schemes make it computationally infeasible for an attacker to impersonate a user even if the attacker completely compromises the information stored at the server, including all the server’s secret keys. Likewise, our schemes make it computationally infeasible for an attacker to impersonate a user even if the attacker completely compromises the information stored at the client device (but not the biometric itself, which is assumed to remain attached to the user and is not stored on the client device in any form).

**Keywords:** biometrics, authentication, smart cards, cryptographic hash functions.

---

<sup>1</sup> Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park’s e-enterprise Center.

<sup>2</sup> This work was supported in part by Grants CCR-0312760, CCR-0311720, CCR-0225642, and CCR-0098068 from the National Science Foundation.

<sup>3</sup> This work was supported in part by Grants CCR-0311510, CCR-0098068, and IIS-0324846 from the National Science Foundation and by a research gift from Sun Microsystems.

## 1 Introduction

Biometric-based identification starts with a physical measurement for capturing a user’s biometric data, followed by the extraction of features from the measurement, and finally a comparison of the feature vector to some previously-stored reference vector. While biometric-based identification holds the promise of providing unforgeable authentication (because the biometric is physically attached to the user), it has a number of practical disadvantages. For example, the storage of reference vectors presents a serious privacy concern, since they usually contain sensitive information that many would prefer to keep private. Even from a security standpoint, biometric information must be stored and transmitted electronically, and, as the old adage goes, a user only gets nine chances to change her fingerprint password (and only one chance to change a retinal password). Thus, we would like to protect the privacy of biometric reference vectors.

One of the major difficulties in biometric information is that, even when it comes from the same individual, it is variable from one measurement to the next. This means that standard encryption of the reference vector is not sufficient to achieve the desired properties. For, even when the reference vector is stored in encrypted form, it appears as though the comparison step (comparing a recently-read biometric image to the reference vector) needs to be done in the clear. That is, standard techniques of comparing one-way hashes (or encryptions) of a stored password and an entered password cannot be used in the context of biometric authentication, as two very similar readings will produce very different hash (or encrypted) values. Unfortunately, this cleartext comparison of biometric data exposes sensitive information to capture by an adversary who obtains one of the two in-the-clear comparands, e.g., through spy-ware at the client or at the server. Moreover, in addition to this comparison-step vulnerability, encrypting the reference vector is obviously not sufficient to protect biometric data from an adversary who learns the decryption key, as could be the case with a dishonest insider at a financial institution. We next review previous work in overcoming these difficulties.

### 1.1 Related Work

There is a huge literature on biometric authentication, and we briefly focus here on the work most relevant to our paper. There are two broad approaches: The one where the comparison is done at the remote server, and the one where the comparison is done at the client end (the portable device where the biometric measurement is done). Most of the recent work has focused on the second category, ever since the landmark paper of Davida et al. [9] proposed that comparisons be done at the client end (although their scheme is also useful in the first case, of remote comparison at the server end). Many other papers (e.g., [4, 3, 13], to mention a few) build on the *wallet with observer* paradigm introduced in Chaum et al. [6] and much-used in the digital cash literature; it implies that there is a tamper-proof device available at the client’s end where the comparison is made. The “approximate equality” in biometric comparisons is a challenge

faced by any biometric scheme, and many ways have been proposed for overcoming that difficulty while preserving the required security properties. These include the use of error-correcting codes [9, 10, 16, 8], fuzzy commitments and fuzzy vaults [16, 15, 7] (for encrypting the private key on the smartcard using fingerprint information), fuzzy extractors [11], and the use of secure multi-party computation protocols [17]. Some carry out the comparisons “in the clear” (after decryption), whereas others manage to avoid it. They all rely, in varying degrees, to one (or more) of the following assumptions: That the portable device is tamper-resistant (“wallet with observer” based papers), that the portable device is powerful enough to carry out relatively expensive cryptographic computations (public-key encryption, homomorphic encryption), and sequences of these for in carrying out complex multi-step protocols. See [12, 5] for a review and a general discussion of the pitfalls and perils of biometric authentication and identification (and how to avoid them), and [18] for a rather skeptical view of biometrics.

In spite of some drawbacks and practicality issues, these schemes have shown the theoretical (and, for some, practical) existence of secure and private biometric authentication.

## 1.2 Motivation for Our Approach

Just like the tiny (and weak) embedded microprocessors that are now pervasive in cars, machinery, and manufacturing plants, so will biometrically-enabled electro-mechanical devices follow a similar path to pervasiveness (this is already starting to happen due to security concerns). Not only inexpensive smartcards, but also small battery-operated sensors, embedded processors, and all kinds of other computationally weak and memory-limited devices may be called upon to carry out biometric authentication. Our work is based on the premise that biometric authentication will eventually be used in a such a pervasive manner, that it will be done on weak clients and servers, ones that can compute cryptographic hashes but not the more expensive cryptographic primitives and protocols; in a battery-powered device, this may be more for energy-consumption reasons than because the processor is slow. This paper explores the use of such inexpensive primitives, and shows that much can be achieved with them.

Our solutions have other desirable characteristics, such as not relying on physical tamper-resistance alone. We believe that relying entirely on tamper-resistance is a case of “putting too many eggs in one basket”, just as would be a complete reliance on the assumed security of a remote online server – in either case there is a “single point of failure”. See [2, 1] on the hazards of putting too much faith in tamper-resistance. It is desirable that a system’s failure requires the compromise of *both* the client and remote server.

## 1.3 Lightweight Biometric Authentication

As stated above, this paper explores the use of lightweight computational primitives and simple protocols for carrying out secure biometric authentication. As

in the previous schemes mentioned above, our security requirement is that an attacker should not learn the cleartext biometric data and should not be able to impersonate users by replaying encrypted (or otherwise disguised) data. Indeed, we would like a scheme to be resilient against insider attacks; that is, a (dishonest) insider should be unable to use data stored at the server to impersonate a user (even to the server). We also want our solutions to be simple and practical enough to be easily deployed on weak computational devices, especially at the client, which could be a small smartcard biometric reader. Even so, we don't want to rely on tamper-resistant hardware to store the reference vector at the client. Ideally, we desire solutions that make it infeasible for an attacker to impersonate a user even if the attacker steals the user's client device (e.g., a smartcard) and completely compromises its contents.

Even though the main rationale for this kind of investigation is that it makes possible the use of inexpensive portable units that are computationally weak (due to a slow processor, limited battery life, or both), it is always useful to provide such faster schemes even when powerful units are involved.

#### 1.4 Our Contributions

The framework of this paper is one where biometric measurement and feature extraction are done in a unit we henceforth refer to as the *reader*, which we sometimes refer to informally as the "smartcard," although this physical implementation of the reader is just one of many possibilities. It is assumed that the client has physical possession of the reader and, of course, the biometric itself. The alignment and comparison of the resulting measured feature information to the reference feature information is carried out at the *comparison unit*. Both the reference feature information and the comparison unit are assumed to be located at the *server* (at which authentication of the client is desired). Since authentication for financial transactions is a common application of biometric identification, we sometimes refer to the server informally as the "bank."

We present schemes for biometric authentication that can resist several possible attacks. In particular, we allow for the possibility of an attacker gaining access to the communication channel between reader and comparison unit, and/or somehow learning the reference information stored at the comparison unit (reference data is write-protected but could be read by insiders, spyware, etc.). We also allow for the possibility of an attacker stealing the reader from the client and learning the data stored on the reader. Such an attack will, of course, deny authentication service to the client, but it will not allow the attacker to impersonate the user, unless the attacker also obtains a cleartext biometric measurement from the user or the stored reference information at the server. To further resist even these two latter coordinated multiple attacks, the reader could have its data protected with tamper-resistant hardware, but we feel such coordinated multiple attacks (e.g., of simultaneously compromising the reader and the server) should be rare. Even so, tamper-resistant hardware protecting the memory at the reader could allow us to resist even such coordinated attacks.

Given such a rich mix of attacks that we wish to resist, it is desirable that the authentication protocol between reader and comparator not compromise the security or privacy of biometric information. We also require that compromise of the reference data in the comparator does not enable impersonation of the user. These security and privacy requirements pose a challenging problem because biometrics present the peculiar difficulty that the comparisons are necessarily inexact; they are for approximate equality. We give solutions that satisfy the following properties:

1. The protocols use cryptographic hash computations but not encryption. All the other operations used are inexpensive (no multiplication).
2. Information obtained by an eavesdropper during one round of authentication is useless for the next round, i.e., no replay attacks are possible.
3. User information obtained from the comparisons unit by an adversary (e.g., through a corrupt insider or spyware) cannot be used to impersonate that user with that server or in any other context.
4. If a card is stolen and all its contents compromised, then the thief cannot impersonate the user.

Our solutions are based on a *decoupling* of information between the physical biometric, the reader, and the server, so that their communication and storage are protected and private, but the three of them can nevertheless perform robust biometric authentication. Moreover, each authentication in our scheme automatically sets up the parameters for the next authentication to also be performed securely and privately. Our scheme has the property that one smartcard is needed for each bank; this can be viewed as a drawback or as a feature, depending on the application at hand – a real bank is unlikely to trust a universal smartcard and will insist on its own, probably as an added security feature for its existing ATM card infrastructure. On the other hand, a universal card design for our framework of weak computational clients and servers (i.e., a card that works with many banks, as many of the above-mentioned earlier papers achieve) would be interesting and a worthwhile subject of further research. For now, our scheme should be viewed as a biometric supplement to, say, an ATM card’s PIN; the PIN problem is trivial because the authentication test is of exact equality – our goal is to handle biometric data with the same efficiency and results as if a PIN had been used. Our scheme is not a competitor for the powerful PKI-like designs in the previous literature, but rather another point on a tradeoff between cost and performance.

We are not aware of any previous work that meets the above-mentioned security requirements using only lightweight primitives and protocols. The alignment stage, which precedes the comparison stage of biometric matching, need not involve any cryptographic computations even when security is a concern (cf. [17], which implemented a secure version of [14]). It is carrying out the (Hamming or more general) distance-computation in a secure manner that involves the expensive cryptographic primitives (in [17], homomorphic encryption). It is therefore on this “bottleneck” of the distance comparison that we henceforth focus, except

that we do not restrict ourselves to Hamming distance and also consider other metrics (more on this below).

## 2 Security Definition for Biometric Authentication

### 2.1 Adversary Model

An adversary is defined by the resources that it has. We now list these resources, and of course an adversary may have any combination of these resources:

1. *Smartcard (SCU and SCC)*: An adversary may obtain an uncracked version of the client's smartcard (SCU) or a cracked version of the smartcard (SCC). An adversary with SCU does not see the values on the smartcard, but can probe with various fingerprints. An adversary with SCC is also able to obtain all information on the smartcard. We consider an adversary that cracks the smartcard and then gives it back to the user as outside of our attack model.
2. *Fingerprint (FP)*: An adversary may obtain someone's fingerprint, by dusting for the print or by some other more extreme measure.
3. *Eavesdrop (ESD, ECC, and ECU)*: An adversary can eavesdrop on various components of the system. These include: i) The server's database (ESD) which contains all information that the server stores about the client, ii) the communication channel (ECC) which has all information sent between the client and server, and iii) the comparison unit (ECU) which has all information from ESD, ECC, and the result of the comparison.
4. *Malicious (MCC)*: An adversary may not only be able to eavesdrop on the communication channel but could also change values. We consider adversaries that can change the comparison unit or the server's database as outside of our attack model.

### 2.2 Security Definitions

We look at the confidentiality, integrity, and availability of the system. The confidentiality requirements of the system are that an adversary should not be able to learn information about the fingerprint. The integrity of the system requires that an adversary cannot impersonate a client. The availability of the system requires that an adversary cannot make a client unable to login (i.e., "denial of service"). We now formally define the security requirements for the notions above.

#### **Confidentiality:**

We present three oracles that are considered secure in our paper, and we prove confidentiality by showing an adversary is equivalent to one of these oracles; in other words if given such an oracle you could emulate the adversary's information. We assume that the oracle has a copy of the ideal fingerprint  $\bar{f}$ .

1. Suppose the adversary has an oracle  $A : \{0, 1\}^{|\bar{f}|} \rightarrow \{0, 1\}$ , where  $A(f)$  is true iff  $\bar{f}$  and  $f$  are close. In other words, the adversary can try an arbitrary

- number of fingerprints and learn whether or not they are close to each other. We consider a protocol that allows such adversaries to be strongly secure.
2. Suppose the adversary has an oracle  $B : \emptyset \rightarrow \{0, 1\}^{\log |f|}$ , where  $B()$  returns the distance between several readings of a fingerprint (the actual fingerprints are unknown to the adversary). In other words, the adversary sees the distance between several readings of a fingerprint. We consider a protocol that allows such adversaries to be strongly secure.
  3. Suppose the adversary has an oracle  $C : \{0, 1\}^{|f|} \rightarrow \{0, 1\}^{\log |\bar{f}|}$ , where  $C(f)$  returns the distance between  $\bar{f}$  and  $f$ . In other words, the adversary can try many fingerprints and will learn the distance from the ideal. Clearly, this adversary is stronger than the above mentioned adversaries. A protocol with such an adversary has acceptable security only in cases where the attack is detectable by the client; we call this weakly secure.

**Integrity:**

To ensure integrity we show that there is a check in place (either by the server or by the client) that an adversary with the specific resources cannot pass without having to invert a one-way function or guess a fingerprint. Of course if the adversary can weakly guess the fingerprint, then we say that the adversary can weakly impersonate the client.

**Availability:**

The types of denial of service attacks that we consider are those where the adversary can prevent the parties from communicating or can make the parties have inconsistent information which would make them unable to successfully authenticate.

**2.3 Summary of Schemes' Security**

Before we define the security of our system, we discuss the security (in the terms outlined above) of an "ideal" implementation that uses a trusted oracle. Such a system would require that the client use his fingerprint along with the smartcard and that all communication with the oracle take place through a secure communication channel. The trusted oracle would authenticate the user if and only if both the fingerprint and the smartcard were present. Clearly, we cannot do better than such an implementation.

Table 1 is a summary of an adversary's power with various resources (in our protocol); there are three categories of security: Strong, Weak, and No. Where the first two are defined in the previous section, and "No" means that the system does not protect this resource against this type of adversary. Furthermore, we highlight the entries that are different from an "ideal" system. To avoid cluttering this exposition we do not enumerate all values in the table below, but rather for entries not in the table the adversary has capabilities equal to the maximum over all entries that it dominates.

Thus, in many ways, the smartcard is the lynchpin of the system. While it is desirable to have a protocol that requires both the biometric and the smartcard, having the smartcard be the lynchpin is preferable to having the biometric be

**Table 1.** Security of our Protocols

Resources	Confidentiality	Integrity	Availability
FP	No	Strong	Strong
SCC and ESD	<b>No</b>	<b>No</b>	No
SCU and FP	No	No	No
MCC and ESD	Strong	<b>No</b>	No
SCU and ESD and MCC	<b>No</b>	<b>No</b>	No
MCC	Strong	Strong	No
SCU	Strong	Strong	No
SCU and ECU	<b>Weak</b>	<b>Weak</b>	No

the lynchpin. The reason for this is that a biometric can be stolen without the theft being detected, however there is a physical trace when a smartcard is stolen (i.e., it is not there). The only exception to this is when the adversary has malicious control of the communication channel and can eavesdrop on the server’s database, and in this case it can impersonate the client (but cannot learn the fingerprint).

### 3 Some False Starts

In this section, we outline some preliminary protocols for biometric authentication that should be viewed as “warmups” for the better solutions given later in the paper. The purpose of giving preliminary protocols first is twofold: (i) to demonstrate the difficulty of this problem, and (ii) to provide insight into the protocol given later.

Initially, we give preliminary solutions for binary vectors and for the Hamming distance, however these preliminary solutions are extended to arbitrary vectors and other distance functions. The primary question that needs to be addressed is:

“How does the bank compute the Hamming distance between two binary vectors without learning information about the vectors themselves?”

We assume that the server stores some information about some binary vector  $f_0$  (the *reference vector*), and that the client sends the server some information about some other vector  $f_1$  (the recently measured *biometric vector*). Furthermore, the server authenticates the client if  $dist(f_0, f_1)$ , the Hamming distance between  $f_0$  and  $f_1$ , is below some threshold,  $\epsilon$ . In addition to our security goal of being able to tolerate a number of possible attacks, there are two requirements for such a protocol:

- *Correctness*: the server should correctly compute  $dist(f_0, f_1)$ .
- *Privacy*: the protocol should reveal nothing about  $f_0$  and  $f_1$  other than the Hamming distance between the two vectors.

We now give various example protocols that attempt to achieve these goals, but nevertheless fail at some point:

1. Suppose the server stores  $f_0$  and the client sends  $f_1$  in the clear or encrypted for the server. This amounts to the naive (but common) solution mentioned above in the introduction. Clearly, this protocol satisfies the correctness property, but it does not satisfy the privacy requirement. In our architecture, this is vulnerable to insider attacks at the server and it reveals actual biometric data to the server.
2. Suppose, instead of storing  $f_0$ , the server stores  $h(f_0||r)$ , the result of a cryptographic one-way hash of  $f_0$  and a random nonce,  $r$ . The client would then need to compute  $f_1||r$  and apply  $h$  to this string, sending the result,  $h(f_1||r)$ , to the server. This solution improves upon the previous protocol in that it protects the client's privacy. Indeed, the one-way property of the hash function,  $h$ , makes it computationally infeasible for the server to reconstruct  $f_0$  given only  $h(f_0||r)$ . Unfortunately, this solution does not preserve the correctness of biometric authentication, since cryptographic hashing does not preserve the distance between objects. This scheme will work only for the case when  $f_0 = f_1$ , which is unlikely given the noise that is inherent in biometric measurements.
3. Suppose, then, that the server instead stores  $f_0 \oplus r$  and the client sends  $f_1 \oplus r$ , for some random vector  $r$  known only to the client. This solution satisfies the correctness property for biometric authentication, because  $dist(f_0 \oplus r, f_1 \oplus r) = dist(f_0, f_1)$  for the Hamming distance metric. This solution might at first seem to satisfy the privacy requirement, because it hides the number of 0's and 1's in the vectors  $f_0$  and  $f_1$ . However, the server learns the positions where there is a difference between these vectors, which leaks information to the server with each authentication. This leakage is problematic, for after several authentication attempts the server will know statistics about the locations that differ frequently. Depending on the means of how feature vectors are extracted from the biometric, this leakage could reveal identifying characteristics of the client's biometric information. Thus, although it seems to be secure, this solution nonetheless violates the privacy constraint.
4. Suppose, therefore, that the scheme uses a more sophisticated obfuscating technique, requiring the server to store  $\Pi(f_0 \oplus r)$ , for some random vector  $r$  and some fixed random permutation,  $\Pi$ , known only to the client. The client can authenticate in this case by sending  $\Pi(f_1 \oplus r)$ . This solution satisfies the correctness property, because  $dist(\Pi(f_0 \oplus r), \Pi(f_1 \oplus r)) = dist(f_0, f_1)$ , for the Hamming distance metric. Moreover, by using a random permutation, the server does not learn the places in  $f_0$  and  $f_1$  where differences occur (just the places where the permuted vectors differ). Thus, for a single authentication round the server learns only the Hamming distance between  $f_0$  and  $f_1$ . Unfortunately, this scheme nevertheless still leaks information with each authentication, since the server learns the places in the permuted vectors

where they differ. Over time, because the same  $\Pi$  is used each time, this could allow the server to determine identifying information in the biometric.

This final scheme is clearly the most promising of the above false starts, in that it satisfies the correctness and privacy goals for a single authentication round. Our scheme for secure biometric authentication, in fact, is based on taking this final false start as a starting point. The main challenge in making this scheme secure even for an arbitrarily long sequence of authentications is that we need a secure way of getting the server and client to agree on future permutations and random nonces (without again violating the correctness and privacy constraints).

## 4 Our Schemes for Secure Biometric Authentication

In this section, we give our protocols for secure biometric authentication. We begin with a protocol for the case of Boolean vectors where the relevant distance between two such vectors is the Hamming distance. We later extend this to vectors of arbitrary numbers and distance metrics that depend on differences between the corresponding components (this is a broad class that contains the Euclidean distance  $L_2$ , as well as  $L_1$ ). We use  $H(\cdot)$  to denote a keyed hash, where the key is a secret known to the client and server but not to others. An additional challenge in using such a function is that we now must prevent someone who accidentally (or maliciously) learns the client information at the server’s end from using that information to impersonate the client to the server. Likewise, we must maintain the property that someone who learns the client’s information on the reader should not be able to use this information (and possibly previously eavesdropped sessions) to impersonate the client.

### 4.1 Boolean Biometric Vectors

The server (in the database and the comparison unit) and the client (in the smartcard) store a small collection of values, which are recomputed after each round. Also, there are  $q$  copies of this information at the server and on the card, where  $q$  is the number of fingerprint mismatches before a person must re-register with the server. In what follows,  $f_i$  and  $f_{i+1}$  are Boolean vectors derived from biometric readings at the client’s end,  $\Pi_i$  and  $\Pi_{i+1}$  denote random permutations generated by and known to the client but not the server, and  $r_i, r_{i+1}, s_i, s_{i+1}, s_{i+2}$  are random Boolean vectors generated by the client, some of which may end up being revealed to the server.

Before a round, the server and client store the following values:

- The server has:  $s_i \oplus \Pi_i(f_i \oplus r_i), H(s_i), H(s_i, H(s_{i+1}))$ .
- The client has:  $\Pi_i, r_i, s_i, s_{i+1}$ .

A round of authentication must not only convince the server that the client has a vector  $f_{i+1}$  that is “close” (in the Hamming distance sense) to  $f_i$ , but must also refresh the above information. A round consists of the following steps:

1. The client uses the smartcard to read a new biometric  $f_{i+1}$  and to generate random Boolean vectors  $r_{i+1}$  and  $s_{i+2}$  and a random permutation  $\Pi_{i+1}$ .
2. The smartcard connects to the terminal and sends to the server the following values:  $\Pi_i(f_{i+1} \oplus r_i)$ ,  $s_i$ , and “transaction information”  $T$  that consists of a nonce as well as some other information related to this particular access request (e.g., date and time, etc).
3. The server computes the hash of the just-received  $s_i$  and checks that it is equal to the previously-stored  $H(s_i)$ . If this check does not match it aborts the protocol. If it does match, then the server computes the XOR of  $s_i$  with the previously-stored  $s_i \oplus \Pi_i(f_i \oplus r_i)$  and obtains  $\Pi_i(f_i \oplus r_i)$ . Then the server computes the Hamming distance between the just-computed  $\Pi_i(f_i \oplus r_i)$  and the received  $\Pi_i(f_{i+1} \oplus r_i)$ .
  - If the outcome is a match, then the server sends  $H(T)$  to the client.
  - If it is not a match, then the server aborts but throws away this set of information in order to prevent replay attacks; if the server does not have any more authentication parts, then it locks the account and requires the client to re-register.
4. The smartcard checks that the value sent back from the server matches  $H(T)$  (recall that  $H$  is a keyed hash). If the message does not match, the smartcard sends an error to the server. Otherwise, the smartcard sends the server the following information:  $s_{i+1} \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$ ,  $H(s_{i+1}, H(s_{i+2}))$ , and  $H(s_{i+1})$ . It also wipes from its memory the reading of fingerprint  $f_{i+1}$  and of previous random values  $r_i$  and  $s_i$ , so it is left with  $\Pi_{i+1}$ ,  $r_{i+1}$ ,  $s_{i+1}$ ,  $s_{i+2}$ .
5. When the server receives this message it verifies that  $H(s_i, H(s_{i+1}))$  matches the previous value that it has for this quantity and then updates its stored values to:  $s_{i+1} \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$ ,  $H(s_{i+1}, H(s_{i+2}))$ , and  $H(s_{i+1})$ .

## 4.2 Arbitrary Biometric Vectors

Suppose the biometric vectors  $f_i$  and  $f_{i+1}$  now contain arbitrary (rather than binary) values, and the proximity decision is based on a distance function that depends on  $|f_i - f_{i+1}|$ .

Modify the description of the Boolean protocol as follows:

- Each of  $r_i, r_{i+1}$  is now a vector of arbitrary numerical values rather than Boolean values (but  $s_i, s_{i+1}, s_{i+2}$  are still Boolean).
- Every  $f_j \oplus x$  gets replaced in the protocol’s description by  $f_j + x$ , e.g.,  $f_i \oplus r_i$  becomes  $f_i + r_i$ . (The length of  $s_i$  must of course now be the same as the number of bits in the binary representation of  $f_i + r_i$ , but we refrain from belaboring this straightforward issue.)

The above requires communication  $O((\log \Sigma)n)$ , where  $\Sigma$  is the size of the alphabet and  $n$  is the number of items. This reveals slightly more than the distance, in that it reveals the component-wise differences. This information leakage is minimal especially since the values are permuted, but clearly a protocol

that does not leak such information is preferred. In the case where the function is  $\sum_{i=1}^n |f_i - f_{i+1}|$ , we could use a unary encoding for each value and reduce it to a Hamming distance computation, for which the protocols of the previous section can then be used. This does not reveal the component-wise differences, but it requires  $O(\Sigma n)$  communication.

## 5 Security of the Protocols

In this section, we prove that Table 1 in Section 2.3 is accurate. First, we define the information and abilities of the adversaries, and then prove the confidentiality, integrity and availability constraints.

### Resources

The following table summarizes the information of various adversaries. Generally, an adversary with multiple resources gets all of the information of each resource. There are cases where this is not the case, e.g., consider an adversary with SCU and ECC; the adversary could not see readings of the client's fingerprint, because the client no longer has the smartcard to attempt a login.

**Table 2.** Information of Various Adversaries

Adversary	Information
FP	f
SCU	Ability to probe small number of fingerprints
SCC	SCU and $r_i, s_i, \Pi_i, k$
ESD	$k$ and several sets of $H(s_i), H(s_i, H(s_{i+1})), s_i \oplus \Pi_i(f \oplus r_i)$
ECC	Several sets of $s_i, \Pi_i(f \oplus r_i), H(s_{i+1}), H(s_{i+2})$
ECU	ESD and ECC and distances of several readings
MCC	ECC and can change values

### Confidentiality

Before we prove the confidentiality requirements we need the following lemma:

**Lemma 1.** *The pair of values  $(\Pi(f \oplus r))$  and  $(\Pi(f' \oplus r))$  reveals nothing other than the distance between each pair of vectors.*

**Proof:** The proof of this claim will be in the full version of the paper.

**Theorem 1.** *The only cases where an adversary learns the fingerprint are in: i) FP, ii) SCC and ESD, iii) SCU and ESD and MCC, and iv) any superset of these cases. In the case of SCU and ECU the adversary weakly learns the fingerprint.*

**Proof:** First when the adversary has the fingerprint the case is clearly true. Suppose that the adversary has ECU and MCC, the adversary sees several pairs of  $\Pi(f \oplus r)$  and  $\Pi(f' \oplus r)$  and by Lemma 1, this only reveals a set of distances, which is equivalent to oracle B and thus is secure. Thus any attack must involve

an adversary with the smartcard in some form. Clearly, any adversary with the smartcard cannot eavesdrop on communication when the client is logging into the system.

Suppose that the adversary has SCC and MCC. The adversary has no information about the fingerprint in any of its information, since nothing is on the smartcard and a client cannot login without the smartcard, and thus the fingerprint is protected. However, if the adversary has SCC and ESD, they can trivially learn the fingerprint from knowing  $\Pi_i, r_i, s_i, s_i \oplus \Pi_i(f \oplus r_i)$ .

Any adversary with SCU can only probe various fingerprints, as no other information is given. Suppose that the adversary has SCU and ECU. In this case the adversary can probe various fingerprints and can learn the distance, which is equivalent to oracle  $C$  and thus is weakly secure. Consider an adversary with SCU and ESD. In this case they can probe using the SCU, but this is just oracle  $A$ . If the adversary has SCU and MCC, they can learn  $s, \Pi$ , and  $r$  values by stopping the traffic and trying various fingerprints, however they cannot use this to glean the fingerprint as the client cannot login once the smartcard is stolen. Finally, if the adversary has SCU and MCC and ESD, then they can learn the values and then learn the fingerprint.  $\square$

### Integrity and Availability

**Theorem 2.** *The only cases where an adversary can impersonate a client are in: i) SCU+FP, ii) SCC and ESD, iii) MCC and ESD, and iv) any superset of these cases. In the case of SCU and ECU the adversary weakly impersonate the client. The only cases where an adversary can attack the availability of the client are in: i) SCU, ii) MCC, and iii) any superset of these cases.*

**Proof:** The proof of this claim will be in the full version of the paper.

## 6 Storage-Computation Tradeoff

In this section, we introduce a protocol that allows  $q$  fingerprint mismatches before requiring the client to re-register with the server, with only  $O(1)$  storage, but that requires  $O(q)$  hashes to authenticate. This utilizes similar ideas as SKEY [19]; in what follows  $H^j(x)$  denotes the value of  $x$  hashed  $j$  times. We do not prove the security of this system due to space limitations. After the setup the following is the state of the system:

- Server has:  $\bigoplus_{j=0}^{q-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$ ,  $H^q(s_i)$ , and  $H(H^q(s_i), H^q(s_{i+1}))$ .
- Client has:  $\Pi_i, r_i, s_i$ , and  $s_{i+1}$ .

After  $t$  fingerprint mismatches the server has:  $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$ ,  $H^{q-t}(s_i)$ , and  $H(H^q(s_i), H^q(s_{i+1}))$ .

The authentication and information-updating round is as follows for the  $t$ th attempt to authenticate the client:

1. The client uses the smartcard to read a new biometric  $f_{i+1}$  and to generate random Boolean vectors  $r_{i+1}$  and  $s_{i+2}$  and a random permutation  $\Pi_{i+1}$ .

2. The smartcard connects to the terminal and sends to the server the following values:  $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_{i+1} \oplus r_i)$  and  $H^{q-t}(s_i)$ .
3. The server computes the hash of the just-received  $H^{q-t}(s_i)$  and checks that it is equal to the previously-stored  $H^{q-t+1}(s_i)$ . If this check does not match it aborts the protocol. If it does match, then the server computes the XOR of  $H^{q-t}(s_i)$  with the previously-stored  $\bigoplus_{j=0}^{q-t} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$  and obtains  $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$ . It then computes the Hamming distance between the just-computed  $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$  and the received  $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_{i+1} \oplus r_i)$ .
  - If the outcome is a match, then the server sends  $H(T)$  (recall that  $H$  is a keyed hash) to the client.
  - If it is not a match, then the server updates its values to the following:  $\bigoplus_{j=0}^{q-t-1} H^j(s_i) \oplus \Pi_i(f_i \oplus r_i)$ ,  $H^{q-t}(s_i)$ , and  $H(H^q(s_i), H^q(s_{i+1}))$ . If  $t = q$ , then the server locks the account and requires the client to re-register.
4. If the smartcard checks that the value sent back from the server matches  $H(T)$ , then the smartcard sends the server the following information:  $\bigoplus_{j=0}^{q-1} H^j(H^{q-t}(s_{i+1})) \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$ , as well as  $H^q(s_{i+1})$ , and also  $H(H^q(s_{i+1}), H^q(s_{i+2}))$ . If it does not match, then it sends an error to the server and aborts. In either case, it wipes from its memory the reading of fingerprint  $f_{i+1}$  and those previously stored values that are no longer relevant.
5. When the server receives this message it verifies that  $H(H^q(s_i), H^q(s_{i+1}))$  matches the previous value that it has for this quantity and then updates its stored values to:  $\bigoplus_{j=0}^{q-1} H^j(H^{q-t}(s_{i+1})) \oplus \Pi_{i+1}(f_{i+1} \oplus r_{i+1})$ ,  $H^q(s_{i+1})$ , and  $H(H^q(s_{i+1}), H^q(s_{i+2}))$ .

## 7 Conclusions and Future Work

In this paper, a lightweight scheme was introduced for biometric authentication that could be used by weak computational devices. Unlike other protocols for this problem, our solution does not require complex cryptographic primitives, but instead relies on cryptographic hashes. Our protocols are secure in that the client's fingerprint is protected, it is "hard" to impersonate a client to the comparison unit, and adversaries with malicious access to the communication channel cannot steal a client's identity (i.e., be able to impersonate the client to the comparison unit after the transaction). To be more precise, an adversary would need the smartcard and either the fingerprint or the server's database to impersonate the client. One problem with our protocol is that for every successful authentication the database must update its entry to a new value (to prevent replay attacks), and thus we present the following open problem: is it possible for the server to have a static database and have a secure authentication mechanism that requires only cryptographic hash functions?

## References

1. R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *International Workshop on Security Protocols*, pages 125–136, 1997.
2. R. J. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pages 1–11, 1996.
3. G. Bleumer. Biometric yet privacy protecting person authentication. In *Proceedings of 1998 Information Hiding Workshop (IHW 98)*, pages 101–112. Springer-Verlag, 1998.
4. G. Bleumer. Offline personal credentials. Technical Report TR 98.4.1, AT&T, 1998.
5. R. M. Bolle, J. H. Connell, and N. K. Ratha. Biometric perils and patches. *Pattern Recognition*, 35(12):2727–2738, 2002.
6. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Crypto '92, LNCS 740*, pages 89–105. Springer Verlag, 1993.
7. T. C. Clancy, N. Kiyavashr, and D. Lin. Secure smartcard-based fingerprint authentication. In *Proceedings of the 2003 ACM Workshop on Biometrics Methods and Applications*, pages 45–52, 2003.
8. G. Davida and Y. Frankel. Perfectly secure authorization and passive identification for an error tolerant biometric system. In *Proceedings of 7th Conference on Cryptography and Coding, LNCS 1746*, pages 104–113, 1999.
9. G. I. Davida, Y. Frankel, and B. J. Matt. On enabling secure applications through off-line biometric identification. In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, pages 148–157, May 1998.
10. G. I. Davida, Y. Frankel, and B. J. Matt. On the relation of error correction and cryptography to an off-line biometric based identification scheme. In *Proceedings of WCC99, Workshop on Coding and Cryptography*, 1999.
11. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004, LNCS 3027*, pages 523–540. Springer Verlag, 2004.
12. G. Hachez, F. Koeune, and J.-J. Quisquater. Biometrics, access control, smart cards: A not so simple combination. In *Proc. of the Fourth Working Conference on Smart Card Research and Advanced Applications (CARDIS 2000)*, pages 273–288. Kluwer Academic Publishers, September 2000.
13. R. Impagliazzo and S. M. More. Anonymous credentials with biometrically-enforced non-transferability. In *Proceedings of the Second ACM Workshop on Privacy in the Electronic Society (WPES '03)*, pages 60–71, October 2003.
14. A. Jain, L. Hong, and R. Bolle. On-line fingerprint verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):302–314, 1997.
15. A. Juels and M. Sudan. A fuzzy vault scheme. In *Proceedings of the 2002 IEEE International Symposium on Information Theory*, pages 408–413, 2002.
16. A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36. ACM Press, 1999.
17. F. Kerschbaum, M. J. Atallah, D. Mraihi, and J. R. Rice. Private fingerprint verification without local storage. In *International Conference on Biometric Authentication (ICBA)*, July 2004.
18. B. Schneier. Biometrics: Truths and fictions  
<http://www.schneier.com/crypto-gram-9808.html#biometrics>.
19. B. Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., 2nd edition, 1995.