A Biometric Identity Based Signature Scheme

Andrew Burnett, Fergus Byrne, Tom Dowling, and Adam Duffy
(Corresponding author: Andrew Burnett)

Computer Security and Cryptography Group, Computer Science Department
NUI Maynooth, Co. Kildare, Ireland
(Email: cryptogrp@cs.nuim.ie)

(Received Dec. 16, 2005; revised and accepted Jan. 7 & Jan. 27, 2006)

Abstract

We describe an identity based signature scheme that uses biometric information to construct the public key. Such a scheme would be beneficial in many repudiation situations for example a legal dispute over whether a contract had been signed or not by a user. A biometric reading provided by the alleged signer would be enough to verify the signature. We make use of fuzzy extractors to generate a key string from a variable biometric measurement. We use this biometric based key string and an elliptic curve point embedding technique to create the public key and corresponding private key. We then make use of a pairing based signature scheme to perform signing and verification with these keys. We discuss security issues of the system and suggest countermeasures to attack. Finally we describe a working biometric signature scheme developed in part by reusing components in an existing Java Identity Based Encryption implementation.

Keywords: Biometrics, cryptographic applications, identity based signatures, repudiation

1 Introduction

In this paper we present a biometric identity based signature scheme (BIO-IBS). Traditional public key cryptosystems use very long integers, typically 2048 bits, as public keys. These systems rely on digital certificates to connect an identity like a person or a machine to a public key. Identity based systems have the advantage that the public key is the identity, usually an arbitrary string like an email address. In our case we use a biometric measurement of an individual. A significant problem here is the fact that biometric identities tend to vary over time. Obviously this causes problems for key generation. We discuss how to counteract this problem below. One of the main uses of signature schemes is in the area of non-repudiation of documents. Our scheme is particularly useful in this area as biometric measurements such as fingerprints have been established for a long time as evidential tools [16].

Consider the following situation: A user signs a contract with their private key using the BIO-IBS signature scheme. A dispute develops about the signature on the contract. The user only needs to have their biometric taken by an arbitrator to determine the validity of the signature. As the biometric measurement is used as a public key here there is no need to worry about the biometric measurement being compromised [22].

The paper is organized as follows. In Section 2 we briefly outline the basics of elliptic curves over finite fields. We also discuss the important concept of bilinear maps over these curves that forms the core of identity based systems. In Section 3 we describe the process of turning biometric data into key strings. We include a discussion on the type of error correcting codes used. Section 4 will give an overview of the SOK-IBS identity based signature scheme using the key strings generated from the biometric data. It also describes the modifications needed to create the BIO-IBS system. An implementation note on how to practically embed an identity string onto an elliptic curve is presented for completeness. Section 5 outlines security issues with the system. Section 6 describes design issues involved in extending an existing Java Identity Based Encryption system to incorporate BIO-IBS. Finally, we will discuss conclusions and future work. We also provide appendices on the process of biometric extraction and on the working BIO-IBS prototype developed.

2 Elliptic Curve Background

We use the symbol $\oplus$ to denote bitwise exclusive or, XOR. We represent the finite field with p elements as $\mathbb{F}_p = \{0, 1, 2, 3, \ldots, p-2, p-1\}$. For the remainder of this paper we assume $p \equiv 3 \mod 4$. The extension field $\mathbb{F}_p^2$ can be defined as $\mathbb{F}_p^2 = \{a + bi\}$ where $a, b \in \mathbb{F}_p$ and $i = \sqrt{-1}$.

The basic units for elliptic curve arithmetic are points (x, y) on an elliptic curve, E, over a finite field, $\mathbb{F}_p$, denoted $E(\mathbb{F}_p)$, of the form

$$y^2 = x^3 + ax + b \text{ with } x, y, a, b \in \mathbb{F}_p.$$

We define abstract concepts of addition, $P + Q$, and

scalar multiplication by an integer, sQ, on the points of $E(\mathbb{F}_p)$. We also define a special point at infinity, ∞, which is not a solution to the equation given above. These operations combine to make $E(\mathbb{F}_p)$ a finite abelian group with ∞ behaving as the identity element. Details of how these concepts are implemented appear in [2, 14]. The elliptic curve discrete logarithm problem (ECDLP) states: Given two points in an elliptic curve group, P and Q, find a number k such that $kP = Q$. k is called the discrete logarithm of Q to the base P. Cryptographic schemes based on elliptic curves typically rely on the difficulty of solving the ECDLP for their security. The *order* of a point P is defined to be the smallest integer n such that $nP = \infty$. We let $E(\mathbb{F}_p)[q]$ be the subgroup of $E(\mathbb{F}_p)$ consisting of points of order q. This is also called the group of q-torsion points on $E(\mathbb{F}_p)$.

We let $\mu(q) = \{a \in \mathbb{F}_p^2 \mid a^q = 1\}$ be the q^{th} roots of unity. In the following we concentrate on the curve $E(\mathbb{F}_p) : y^2 = x^3 + x$.

Fundamental to pairing based cryptography is the concept of a bilinear mapping between two groups G_1 and G_2 denoted by $\hat{e} : G_1 \times G_1 \rightarrow G_2$ with the bilinearity property that $\hat{e}(xP, yQ) = \hat{e}(P, Q)^{xy}$ for all $P, Q \in G_1$ and for any integers x, y.

One such mapping is the Tate pairing τ_q' that outputs an element of the finite field $\mathbb{F}_p^2$.

$$\tau_q' : E(\mathbb{F}_p)[q] \times E(\mathbb{F}_p^2)/qE(\mathbb{F}_p^2) \rightarrow \mu(q)$$
$$\tau_q'(P, Q) = a + bi \in \mu(q).$$

Note that we can make computational savings by using a modified version of the Tate pairing. We define the map $\phi : E(\mathbb{F}_p) \rightarrow E(\mathbb{F}_p^2)$ as follows:

$$\phi(x, y) = (-x + 0i, 0 + yi).$$

For example, given $(3, 12) \in E(\mathbb{F}_{19})$, $\phi(3, 12) = (16 + 0i, 0 + 12i)$. We then define the modified Tate pairing

$$\tau_q : E(\mathbb{F}_p)[q] \times E(\mathbb{F}_p) \rightarrow \mu(q)$$

as follows:

$$\tau_q(P, Q) = \tau_q'(P, \phi(Q)).$$

This allows us to pick Q in $E(\mathbb{F}_p)$ and thus avoid the complexity of dealing with cosets in $E(\mathbb{F}_p^2)$. A more detailed account of the modified Tate pairing appears in [9]. The evaluation of the Tate pairing involves the ideas of formal divisors on elliptic curves and functions defined on these divisors. These abstract concepts can be computed numerically by using a computationally efficient algorithm developed by Miller [18]. A good account of how Miller's algorithm is used to evaluate the Tate pairing is given in [25]. The most important property of the Tate and modified Tate pairing is bilinearity. Further details on bilinear maps can be found in [25].

3 Generating Key Data from Biometrics

Using biometric data as a basis for cryptographic keys is problematic as biometric measurement is not perfectly reproducible. Recent work [8, 10, 11] demonstrates how such data can be used to generate strong keys for any kind of cryptographic application. They use the notion of a fuzzy extractor to describe the process of extracting a unique string *ID* from a biometric input b, in such a way that a certain amount of error is allowed for. If the input changes slightly to b' then the extracted *ID* will be the same. To enable the regeneration of *ID* from a slightly different biometric b' the fuzzy extractor process also outputs a publicly available reproduction parameter *PAR*. Dodis et al. [8] describe three metrics to measure the variation in the biometric reading: Hamming Distance, Set Difference and Edit Distance. They then detail the construction of fuzzy extractors using these metrics. Hamming Distance is defined to be the number of bit positions that differ between b and b' and is probably the most natural and straightforward metric to work with, although the other metrics may be more efficient for particular biometrics and applications.

3.1 Fuzzy Extraction

Obtaining a biometric reading is a variable process that introduces errors. Examples of such errors include variation in the physical biometric (e.g. a cut on a finger) or bad placement of the finger on the reading device. The reader and matching software can attempt to fix these errors using various techniques. These techniques such as feature extraction may also introduce variation [16]. We use Fuzzy Extractors as an extra level of error correction to try to improve the situation further. The fuzzy extractor construction using the Hamming Distance metric is based on previous work on a fuzzy commitment scheme in [10]. We now give a simplified outline of how such an extractor is constructed. Comprehensive accounts appear in [8, 10]. Further information on the other metrics can be found in [7, 8, 11].

First we give the formal definition of a fuzzy extractor. Let $\mathcal{M} = \{0, 1\}^v$ be a finite dimensional metric space consisting of biometric data points, with a distance function `dis`: $\mathcal{M} \times \mathcal{M} \rightarrow \mathbb{Z}^+$, which calculates the distance between two points based on the metric chosen. Let l be the number of bits of the extracted output string *ID* and t be the error threshold (i.e. for two points $b, b' \in \mathcal{M}$ to be classed as the same $\texttt{dis}(b, b') \leq t$). An $(\mathcal{M}, l, t)$-fuzzy extractor is constructed using two functions `Gen` and `Rep`. `Gen` is a probabilistic generation procedure, which on input $b \in \mathcal{M}$ outputs an "extracted" string $ID \in \{0, 1\}^l$ and a publicly available reproduction parameter, *PAR*. `Rep` is a deterministic reproduction procedure allowing recovery of *ID* from the corresponding *PAR* and any b' sufficiently

close to b. To clarify

$$\forall\, b, b' \in \mathcal{M} \text{ with } \mathtt{dis}(b, b') \leq t,$$
$$\text{if } \mathtt{Gen}(b) \rightarrow \langle ID, PAR \rangle,$$
$$\text{then } \mathtt{Rep}(b', PAR) \rightarrow ID.$$

We now outline the construction of a fuzzy extractor for the space $\mathcal{M}$ under the Hamming Distance metric. This construction is built using error correcting codes. We now define what we mean by an error correcting code. Let the set C be a subset of n-bit words (i.e. $C \subseteq \{0,1\}^n$), with $n > v$ and having at least 2^k elements for some positive integer k. Let $C_e : \mathcal{M} \rightarrow C$ be a one-to-one encoding function and also let $C_d : \{0,1\}^n \rightarrow C$ be a decoding function that has an error threshold of t (can correct up to t-bit errors). The decoding function C_d will take an arbitrary n-bit string and "correct" it to the nearest codeword in C. Using coding theory notation the combination of C, C_e, and C_d gives us an $[n, k, 2t+1]$ binary error correcting code [24]. We also define $H : \{0, 1\}^n \rightarrow \{0,1\}^l$ to be a one way hash function.

A note on the error correction process may be useful here. The most common use of error correcting codes is to reconstruct the original data from a possibly corrupted version of itself. The corruption is usually introduced during transmission. In our case we use error correcting codes to reconstruct the original string representing a biometric measurement from a possibly corrupted version of itself where the corruption is introduced by variation in the biometric and/or in the reader.

3.2 Choice of Error Correcting Codes

In this section we will give the reader an overview of the specific error correcting code we use in the implementation. We chose to use a BCH (Bose-Chaudhuri-Hocquenghem) code because it is an excellent general purpose binary code. The fact that the BCH class of codes have polynomials as codewords makes them more complex and powerful compared to simpler codes like Hamming codes, as they can be constructed to be multi-error correcting. They can be designed to correct errors up to about half the code's block length.

We now look at the actual process of constructing a BCH code. The variable t is defined to be the number of errors to be corrected and $n = 2^m - 1$ is the length of the code, where m, n are positive integers. The polynomial $f(x) \in \mathbb{Z}_2[x]$ is defined to be $x^n - 1$. The ring $R = \mathbb{Z}_2[x]/(f(x))$ consists of the polynomials in $\mathbb{Z}_2[x]$ of degree less than n. There exists a special generator polynomial $g(x) \in \mathbb{Z}_2[x]$ that divides $f(x)$.The code C is generated by $g(x)$ and consists of the multiples of $g(x)$ in $\mathbb{Z}_2[x]$ of degree less than n creating a vector space in R with dimension $k = n - \deg g(x)$. The polynomials in C form codewords in an $[n, k]$ linear code in R with 2^k codewords. A codeword $c(x) \in \mathbb{Z}_2[x]$ with n coefficients can be easily expressed as a unique vector in $\mathbb{Z}_2^n$ by listing its coefficients in order.

The generator polynomial $g(x)$ must be computed in a particular way in order to construct a valid BCH code. If $p(x)$ is a primitive polynomial of degree m in $\mathbb{Z}_2[x]$ then $\mathbb{Z}_2[x]/(p(x))$ is a field of order 2^m whose nonzero elements are generated by the field element x. We let $a_1, a_2, \ldots, a_s$ where $s < n$, be the roots of $f(x)$ with minimum polynomials $m_1(x), m_2(x), \ldots, m_s(x)$. We then let the roots $a_i = x^i$ for $i = 1, \ldots, s$. Using this method we can find $g(x)$ by forming the product of each unique $m_i(x)$, as a result $g(x)$ will divide $f(x)$ giving us a $[n, k, 2t+1]$ BCH code.

Encoding a block of data to a BCH codeword is straight forward. The binary data is first converted to a polynomial $d(x) \in \mathbb{Z}_2[x]$. The redundancy polynomial $w(x)$ is then calculated as the remainder after dividing $x^{n-k}d(x)$ by the generator $g(x)$. The codeword is then the coefficients of $w(x)$ with the coefficients of $d(x)$ appended to the end.

Due to the special way $g(x)$ is formed it is possible to correct errors in transmitted codewords. Suppose $c(x) \in C$ is transmitted and $r(x) \neq c(x) \in \mathbb{Z}_2[x]$ is received with degree less than n. Then $r(x) = c(x) + e(x)$ for some nonzero polynomial $e(x) \in \mathbb{Z}_2[x]$, called the error polynomial, with degree less then n. To correct $r(x)$ all we need to find is $e(x)$. Theorem 4.1 in [12] implies that $r(a^i) = e(a^i)$ for $i = 1, \ldots, 2t$, thus knowing $r(x)$ provides us with information about $e(x)$. The values of $r(a^i)$ are called the syndromes of $r(x)$. Consider $e(x) = x^{m_1} + x^{m_2} + \ldots + x^{m_p}$ for some integer error positions $m_1 < m_2 < \cdots < m_p$ with $p \leq t$ and $m_p < 2^n - 1$. We find these error positions by computing the first $2t$ syndromes of $r(x)$, i.e. $r(a), r(a^2), \ldots, r(a^{2t})$ and using the error locater polynomial $E(z) = (z - a^{m_1})(z - a^{m_2}) \ldots (z - a^{m_p})$. To find these error positions in our implementation we used Berlekamp's iterative algorithm, we refer the reader to [17] for a comprehensive account of BCH codes and Berlekamp's algorithm.

3.3 The Extraction Process

In this section we will give an overview of the implementation of the process of acquiring a unique string from a varying biometric. For the purposes of this paper this implementation will take place using a standard personal computer and a biometric reader connected via a USB port using BioAPI[1] compliant readers.

The first step is to obtain b from the biometric reader. We outline the approach in Appendix 7.

The next step is to use an error correcting code such as a BCH code to fuzzy extract some data from the biometric input. We now describe how such a code is used to build a fuzzy extractor by defining the `Gen` and `Rep` functions. The `Gen` function takes the biometric input b and outputs $ID = H(b)$ that is used for key generation and a publicly accessible reproducer $PAR = b \oplus C_e(ID)$ which is used in conjunction with another biometric input b' to recover ID. The `Rep` function takes b' and PAR as input and

[1]The BioAPI Consortium: `http://www.bioapi.org`

outputs $ID' = C_d(b' \oplus PAR) = C_d(b' \oplus b \oplus C_e(ID))$ which is equal to ID if and only if $\mathtt{dis}(b, b') \leq t$. We illustrate the process below in Figure 1.

The parameters produced in this phase, ID and PAR form the inputs into the key generation phase of Section 4.2. Note that PAR will be needed during verification in order to reconstruct the biometric key.

4 Biometric Identity Based Signature Scheme

Shamir first proposed the concept of identity-based cryptography in [23]. In this paper he also constructed an IBS scheme using the existing RSA algorithm [20]. After Boneh and Franklin used the bilinear pairing to create their Identity Based Encryption (IBE) scheme [3], many IBS schemes based on the bilinear pairing were created. One such scheme was the Sakai-Ohgishi-Kasahara Identity Based Signature (SOK-IBS) scheme [21]; a modified version of this scheme is given in Section 4.1. The SOK-IBS scheme forms the basis of the proposed BIO-IBS scheme described in Section 4.2. In Section 4.3, The functions to map a binary string to a point on an elliptic curve are briefly discussed.

4.1 Modified SOK-IBS

The modified SOK-IBS scheme [1], which can be regarded as an identity based extension of a randomized version of the Boneh-Lynn-Shacham (BLS) short signature scheme [4], was proven to have a "sub-optimal reduction from the Diffie-Hellman problem" in [15]. It consists of the following four parts.

4.1.1 Setup

Given a security parameter k, the Private Key Generator (PKG) selects groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order $q > 2^k$, a generator P of $\mathbb{G}_1$, a randomly chosen master key $s \in \mathbb{Z}_q^*$ and the associated public key $P_{pub} = sP$. It also selects cryptographic hash functions of the same domain and range $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$. A more detailed discussion on the workings of H_1 and H_2 are given in Section 4.3. The system's public parameters are

$$\text{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2).$$

4.1.2 Key Generation

Given a user's identity Φ, the PKG computes $Q_\Phi = H_1(\Phi) \in \mathbb{G}_1$ and the associated private key $d_\Phi = sQ_\Phi \in \mathbb{G}_1$ that is transmitted to the user.

4.1.3 Sign

In order to sign a message M,

1) Pick a random integer $r \in \mathbb{Z}_q$ and compute $U = rP \in \mathbb{G}_1$. Then $H = H_2(\Phi, M, U) \in \mathbb{G}_1$.
2) Compute $V = d_\Phi + rH \in \mathbb{G}_1$ where "+" indicates addition on the group $\mathbb{G}_1$.

The signature on M is the pair $\sigma = \langle U, V \rangle \in \mathbb{G}_1 \times \mathbb{G}_1$.

4.1.4 Verify

To verify a signature $\sigma = \langle U, V \rangle \in \mathbb{G}_1 \times \mathbb{G}_1$ on a message M, the verifier first obtains the signer's identity Φ and computes $Q_\Phi = H_1(\Phi) \in \mathbb{G}_1$. The verifier recalculates $H = H_2(\Phi, M, U) \in \mathbb{G}_1$. He then accepts the signature if $\hat{e}(P, V) = \hat{e}(P_{pub}, Q_\Phi)\hat{e}(U, H)$ and rejects it otherwise.

4.2 BIO-IBS

Our proposed scheme will incorporate fuzzy extractors into the SOK-IBS scheme. Like SOK-IBS, the BIO-IBS scheme consists of four parts.

4.2.1 Setup

Given a security parameter k, the Private Key Generator (PKG) selects groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order $q > 2^k$, a generator P of $\mathbb{G}_1$, a randomly chosen master key $s \in \mathbb{Z}_q^*$ and the associated public key $P_{pub} = sP$. It also selects cryptographic hash functions of the same domain and range $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$. The PKG picks $H_3 : b \rightarrow \{0,1\}^*$, an encoding function C_e and a decoding function C_d. It also selects a method of extracting the features of a biometric, F_e. The system's public parameters are

$$\text{params} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1, H_2, H_3, C_e, C_d, F_e).$$

4.2.2 Key Generation

On obtaining a user's biometric b using the feature extractor F_e, the identity string can be calculated as $ID = H_3(b)$. The PKG computes the public key $Q_{ID} = H_1(ID) \in \mathbb{G}_1$ and the associated private key $d_{ID} = sQ_{ID} \in \mathbb{G}_1$.

4.2.3 Sign

In order to sign a message M,

1) Pick a random integer $r \in \mathbb{Z}_q$ and compute $U = rP \in \mathbb{G}_1$. Then $H = H_2(ID, M, U) \in \mathbb{G}_1$.
2) Compute $V = d_{ID} + rH \in \mathbb{G}_1$.
3) The value $PAR = b \oplus C_e(ID)$ is included as part of the signature.

The signature on M is the triple $\sigma = \langle U, V, PAR \rangle$.

4.2.4 Verify

To verify a signature $\sigma = \langle U, V, PAR \rangle$ on a message M for an identity ID, the verifier performs the following steps.

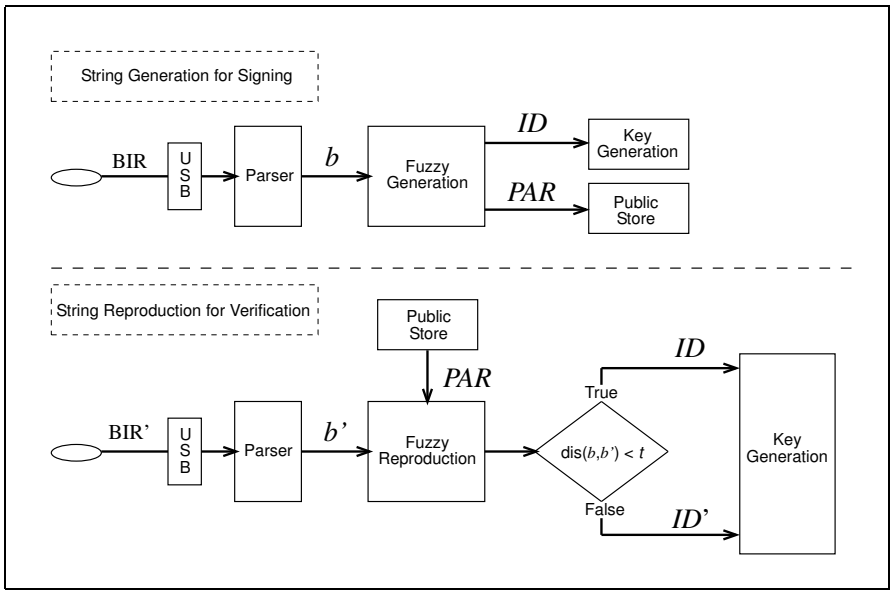

Figure 1: Biometric fuzzy extraction process

1) Obtain a biometric reading b' and calculate $ID' = \text{Rep}(b', PAR)$.[2]

2) Calculate $Q'_{ID} = H_1(ID') \in \mathbb{G}_1$ and $H = H_2(ID', M, U) \in \mathbb{G}_1$.

3) Signature is verified if $\hat{e}(P, V) = \hat{e}(P_{pub}, Q'_{ID})\hat{e}(U, H)$ and rejected otherwise.

4.3 Map To Point

For H_1 and H_2 mentioned above, a binary string $\{0,1\}^*$ has to be embedded onto a point Q on the elliptic curve $E(\mathbb{F}_p)$. This requires the use of a function $f : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ where $\mathbb{G}_1$ is a subgroup of the points on an elliptic curve. Rather than map directly onto $\mathbb{G}_1^*$ we use a standard hash function, H, to hash to a set $A \subseteq \{0,1\}^*$. Then we use a deterministic encoding function, G, to map A onto $\mathbb{G}_1^*$ so $f(ID) = G(H(ID))$. See [3] for more details.

Various algorithms exist to embed onto a point on the curve. We use the algorithm by Koblitz [14, Section 6.2], as we are using curves of the form $y^2 = x^3 + x$. Boneh and Franklin [3, Section 4.3] detail a simpler method for embedding points on curves of the form $y^2 = x^3 + 1$ which is suitable for Weil Pairing.

5 Security Issues

5.1 Malicious signing

It is relatively easy to obtain biometric data from an individual. For example fingerprints can be lifted from a glass. Other examples appear in [16]. Having this data does not allow an attacker to sign a document as the private key is the one used in the signature generation. So this attack is redundant. However it is possible to use this data to attempt to acquire a user's private key.

It is possible for an attacker to obtain your biometric measurement, b, and submit it to the PKG. The PKG will then calculate the corresponding private key and return it to the attacker. The attacker can then use this key to sign documents. This can be counteracted as in most identity based systems by imposing proper authentication procedures for users applying to the PKG to receive private keys. For example users could use a digital certificate here. The advantage over traditional public key systems is that a certificate is necessary only once to obtain a private key.

Note that a practical PKG could issue the private key to a user on a smart card at registration. If this card is pin protected the verification process would be a three-factor authenticated process.

5.2 Disavowal

In our situation recovery of the original biometric is not a problem as we use it to generate the public key. However tampering with β would be useful in an attempt by a legitimate signer to later disavow their signature. Recent work by Boyen et al. [5, 6] shows that the description of fuzzy extractors given in Section 3.1 and [8] is vulnerable to attack. The fuzzy extractor construction of Section 3.1 is secure despite there being a publicly available value β. However, it was found in [5, 6] that this construction is not secure if an active attacker performs a malicious modification of β . Depending on the particular construction of a fuzzy extractor such an attacker may be able to recover the original biometric data. The solution is the use of

[2] If the error $b' \oplus b$ between the two readings is less than the error threshold of the code, then $ID' = ID$ and $Q_{ID} = Q'_{ID}$.

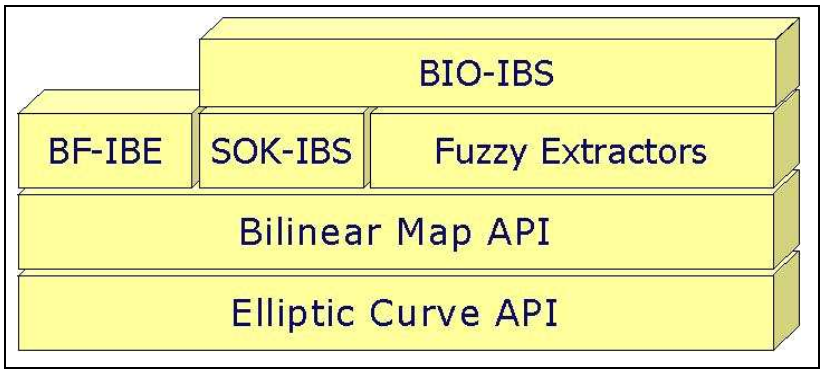

Figure 2: An existing IBE implementation with BIO-IBS extension

a so called robust fuzzy extractor which protects against modification of β. The user will be able to detect if β has been modified with a high probability. This solution is proven secure using the standard model. The proof and details of the construction of a robust fuzzy extractor can be found in [6].

It is also possible for a signer to attempt disavowal by physical attempts to alter their biometric. For example a signer could wear a thin film with another print on their finger or alter facial characteristics to try to fool facial scanners. As this is an undeniable signature scheme these threats can be counteracted by inspecting the biometric prior to taking the measurement.

5.3 Security of the Signature Scheme

A framework for analysing the security of Identity Based Signature schemes was given in [1]. This framework reduces the problem of proving a complex IBS scheme into that of proving a relatively simpler scheme. Consequently, a modified version of the SOK-IBS scheme [21] is then proven by Bellare et. al. to be secure against forgery by chosen message attack.

6 Implementation of BIO-IBS

In this section, we discuss the design and implementation issues involved in extending an existing IBE implementation to incorporate the BIO-IBS system. An overview of the IBE implementation and the BIO-IBS extension is given in Figure 2. Firstly we give a brief overview of the existing IBE implementation. We then discuss the issues related to implementing fuzzy extractors. This section concludes by covering the design of the BIO-IBS key generation and signature scheme.

6.1 An Existing Identity Based Encryption System

One of the core technologies required for our BIO-IBS is a bilinear mapping over an elliptic curve. Such a mapping, the Tate pairing, was previously developed as part of an implementation, [19] of the Boneh and Franklin (BF-IBE) system [3]. Details on an object-oriented (OO) solution for this system with a developer-friendly API appears in [9]. The design follows a pluggable architecture allowing for use of both alternative and enhanced implementations.

The flexible design accommodates both perfectly and not perfectly reproducible identities and, where appropriate, adheres to the Java Cryptographic Architecture (JCA) [13].

6.2 Fuzzy Extractors

The fuzzy extractor class performs two functions, the generation function `Gen` and the reproduction function `Rep`. Since the generation function returns two strings ID and PAR, those strings are encapsulated in the `Generator` class. The reproduction function result is encapsulated in the `Reproducer` class.

A binary error correcting code and a metric space make up the attributes of the fuzzy extractor. These attributes are represented by the interfaces `BinaryErrorCorrectingCode` and `MetricSpace` respectively, allowing for choice in the type of error correcting codes and metric spaces to be used. We extend the `MetricSpace` class with a `HammingDistance MetricSpace` that implements the Hamming Distance metric.

7 Conclusion

We have presented a biometric identity based signature scheme. We have utilised, extended, and implemented ideas in the areas of error corrected string construction from biometric data, key generation, and pairing based signature schemes to form the components of our system. We presented the application of such a scheme to repudiation situations. We discussed the advantage of using the biometric data in the public key and described the utility of using biometric evidence in disputes that may arise. We also discussed the main security issues around this system. Finally we described how such a biometric signature scheme was incorporated into an existing

IBE software package. The pluggable architecture provides for easy incorporation of different implementations of component algorithms. This will facilitate inclusion of future performance enhancements to existing algorithms or inclusion of new algorithms to the system.

Acknowledgements

Andrew Burnett and Adam Duffy acknowledge the support of the Irish Research Council for Science, Engineering and Technology.

References

[1] M. Bellare, C. Namprempre, and G. Neven, "Security proofs for identity-based identification and signature schemes," in *Advances in Cryptology - Eurocrypt'04*, pp. 268-286, 2004.

[2] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1st edition, 1999.

[3] D. Boneh and M. Franklin, "Identity based encryption from the weil pairing," *SIAM Journal of Computing*, vol. 32, no. 3, pp. 586-615, 2001.

[4] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proceedings from Advances in Cryptology - Asiacrypt*, pp. 514-524, 2001.

[5] X. Boyen, "Reusable cryptographic fuzzy extractors," in *ACM Conference on Computer and Communications Security - CCS 2004*, vol. 3494, pp. 82-91, New-York: ACM Press, 2004.

[6] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, "Secure remote authentication using biometric data," in *Advances in Cryptology - EUROCRYPT 2005*, LNCS 3494, pp. 147-163, Springer-Verlag, 2005.

[7] T. Clancy, N. Kiyavash, and D. Lin, "Secure smartcard-based fingerprint authentication," in *Proceedings of the 2003 ACM SIGMM workshop on Biometrics methods and applications*, pp. 45-52, 2003.

[8] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Proceedings from Advances in Cryptology - EuroCrypt*, pp. 523-540, 2004.

[9] A. Duffy and T. Dowling, "An object oriented approach to an identity based encryption cryptosystem," in *The 8th IASTED International Conference on Software Engineering and Applications*, 2004.

[10] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *Proceedings of the 6th ACM conference on Computer and Communications Security*, pp. 28-36, 1999.

[11] A. Juels and M. Wattenberg, "A fuzzy vault scheme," in *Proceedings of the IEEE International Symposium on Information Theory*, 2002.

[12] R. E. Klima, N. Sigmona, and E. Stitzinger, *Applications of Abstract Algebra with MAPLE*, CRC Press, 1st edition, 2000.

[13] J. Knudsen, *Java Cryptography*, O'Reilly, 1st edition, 1998.

[14] N. Koblitz, *A Course in Number Theory and Cryptography*. Springer, 2nd edition, 1994.

[15] B. Libert and J. Quisquater, *The Exact Security of an Identity Based Signature and its Applications*, Cryptology ePrint Archive, Report 2004/102, 2004. `http://eprint.iacr.org/`.

[16] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer, 2003.

[17] R. McEliece, *The Theory of Information and Coding*, Cambridge University Press, student edition, 2004.

[18] V. S. Miller, "The weil pairing, and its efficient calculation," *Journal of Cryptology*, vol. 17, pp. 235-261, 2004.

[19] L. Owens, A. Duffy, and T. Dowling, "An identity based encryption system," in *Proceedings of the 3rd International Conference on Principles and Practice of Programming in Java*, pp. 154-159, 2004.

[20] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, pp. 120-126, 1978.

[21] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *Proceedings of the Symposium on Cryptography and Information Security*, 2000.

[22] B. Schneier, "Inside risks: The uses and abuses of biometrics," *Communications of the ACM*, vol. 42, pp. 136, 1999.

[23] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO'84*, pp. 47-53, 1984.

[24] W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory*, Prentice Hall, 1st edition, 2001.

[25] L. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman and Hall, CRC, 1st edition, 2003.

Appendix A: Acquiring Biometric Data

We outline how to obtain the biometric data b from the biometric reader. An overview of this process can be seen in Figure 3. The process uses the BioAPI framework to acquire the biometric data from the reader. The BioAPI interfaces with the reader through the Biometric Service Provider (BSP) interface of the reader. The BSP interface acquires the raw biometric sample and constructs a Biometric Identification Record (BIR) from this data. This BIR can then be processed by the BSP to enhance the quality of the capture and perform feature extraction on the sample. The BIR returned from a raw sample is large and will contain significant differences from any other sample acquired from the same person depending on properties of the capture. An example of this can be

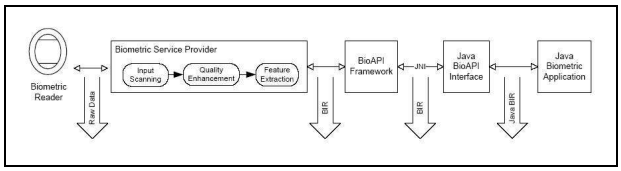

Figure 3: Biometric data extraction process

found in fingerprint biometrics, where the position and orientation of the finger results in significant differences between samples. To get around this problem distinguishing features of the sample are identified using a feature extraction process to form a template. Examples of feature extraction techniques appear in [16]. It is this template which will make up the biometric data b used in key generation. Using the template introduces a problem concerning the compatibility of feature extraction algorithms used by different BSP developers, the algorithm to be used must be then explicitly defined by the system and must not change throughout the process to avoid compatibility problems.

Once the template has been created, we must transfer the data from the native BioAPI interface into the Java fuzzy extractor implementation for key generation. Our approach to this was to create a Java Native Interface (JNI) to the native BioAPI framework. This is important for our system as it allows for system initiated capture, process or template creation operations whenever needed. Our system can use the Java BioAPI Interface to initiate the capture of a biometric sample. The interface then forwards this call to the native BioAPI which performs the capture operation and returns the capture in the form of a BIR to our system. This is then passed onto the next stage for string generation.

Appendix B: Demonstration Application

A Java based implementation of the BIO-IBS scheme exists. In this section we outline the working prototype. The implementation of the scheme conforms to the Java Cryptography Architecture (JCA) API. The JCA API allows developers to use cryptographic components without detailed knowledge of the underlying mathematics reducing the time and effort required by a developer to integrate these components into their own solutions. The BIO-IBS JCA Provider and source code can be downloaded at `http://crypto.cs.nuim.ie/`.

A demonstration application that illustrates the usability of the BIO-IBS scheme can also be downloaded at the aforementioned URL. Due to the lack of an available biometric input device, the demonstration application simulates the capture of biometric data.

Setup

Figure 4 shows a screenshot of the initialisation of the BIO-IBS implementation. The screen consists of two text areas, the left one showing the text to be signed and the right one showing the parameters of the BIO-IBS. The far right shows three biometrics, Bob's biometric at key generation and signing (Bob), Bob's biometric at verification (Bob+e) where e denotes a variation in his biometric reading, such that the variation is within the error correcting capabilities of the BCH code, and Alice's biometric. Bob's biometric should verify his signature on the document whereas Alice's biometric should not verify the signature of the document.

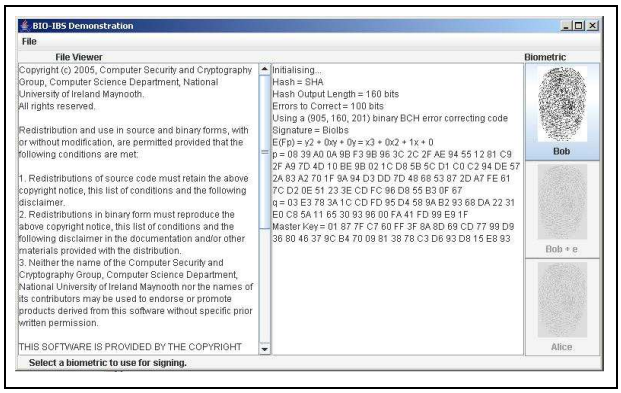

Figure 4: Initialisation of the BIO-IBS. The document to be signed is shown on the left hand side of the screenshot. The right hand side shows the parameters used in the BIO-IBS implementation.

Note that the BCH parameters $[n, k, 2t + 1] = (905, 160, 201)$. The elliptic curve used is $y^2 = x^3 + x$. The master key s is also shown at the end of the right text area. It is shown here for demonstrative purposes only.

Key Generation

Bob uses the biometric reader to sign the document. Before the document is signed, his public and private keys are generated as in Figure 5. As in standard Identity Based cryptosystems, his public key is based on his identity. In the case of BIO-IBS, his public key is derived from his biometric. Note the first part of his biometric value

b = D4 44 5C B3 71 D8 47 A1 20 5B 5D AD 37 06 82 E0.

Please note the underlined segments for later. The remainder of the biometric reading is omitted for brevity. Note that his identity

ID = 15 2A 1E 68 B1 90 27 5C 5C 9C 58 AB 4F C8 0E AE 26 43 CE 38.

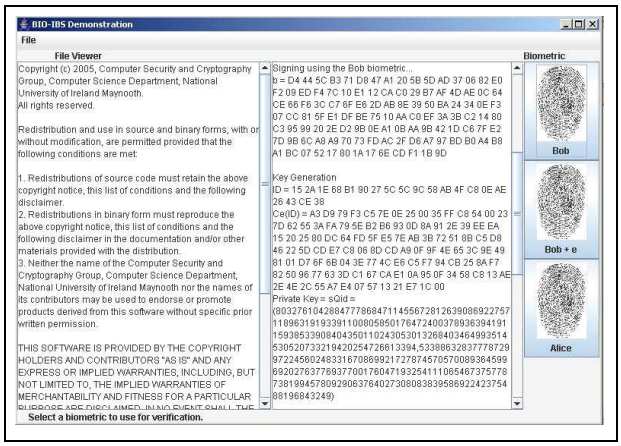

Figure 5: Key generation using BIO-IBS. The public and private keys are generated the first time Bob signs a document (right hand side).

His public key Q_{ID} is then derived from this identity value as shown in Section 4.2.2. His corresponding private key is the affine coordinate $sQ_{ID} = (x, y)$ as shown in the bottom of the right text area.

Signing and Verification

Figure 6 shows the values that represent the signature $\sigma =< U, V, PAR >$ that are calculated by the BIO-IBS implementation. These values are digitally appended to the document to sign it.

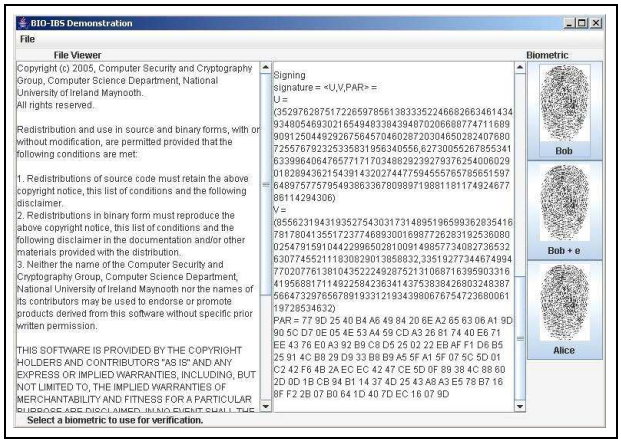

Figure 6: Signing using BIO-IBS. The values for the triple $< U, V, PAR >$ are shown in the right text area.

Bob attempts to verify his signature at a later date (Bob + e) as in Figure 7. His biometric is obtained again. For a variety of reasons there may be errors in his biometric. The screenshot shows that the first part of his biometric is now

b = D4 44 5C FB 71 D8 47 A1 20 5B 58 AD 37 06 82 E0.

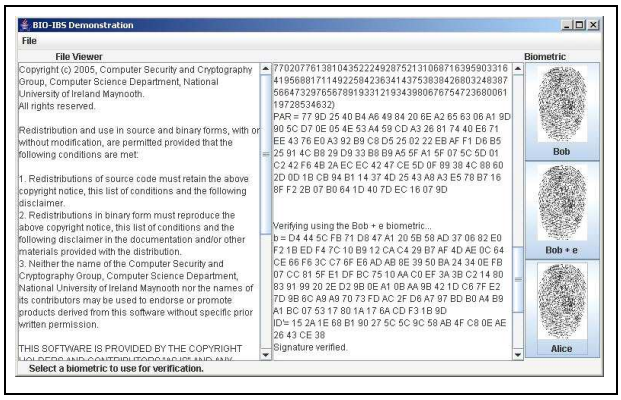

Figure 7: Verification using BIO-IBS. As shown in the bottom of the right text area, Bob's signature is successfully verified.

Note that the underlined segments of this biometric reading are different from those underlined segments of the reading taken at the key generation stage. These are the errors that are corrected by the fuzzy extractor to generate the original identity of

ID' = 15 2A 1E 68 B1 90 27 5C 5C 9C 58 AB 4F C8 0E AE 26 43 CE 38.

The signature is then verified by performing the relevant Tate pairing $\tau_q(P, V) = \tau_q(P_{pub}, Q'_{ID})\tau_q(U, H)$ where P, P_{pub} and H are system parameters, U and V are taken from the signature and Q'_{ID} is the recreated public key based on the biometric reading.

Verification Failure

Figure 8 illustrates when Alice attempts to verify a document signed by Bob. The error correcting code does not create the same identity ID as Bob's. The resulting Tate pairing fails resulting in the signature not being verified.

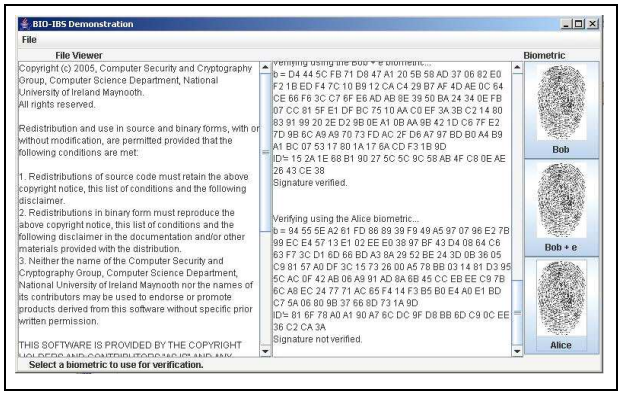

Figure 8: Verification using BIO-IBS. As shown in the bottom of the right text area, Alice's signature is not verified.

Andrew Burnett received his B.Sc. degree in Computer Science from National University of Ireland, Maynooth in 2002. He is currently a Ph.D. candidate at the same institute. His research interests include cryptography, network security and software testing.

Tom Dowling is a lecturer and researcher with the computer security and cryptography group at the National University of Ireland Maynooth. He received a degree in Applied Physics and Mathematics from Dublin Institute of Technology, Ireland in 1991. His M.Sc. and Ph.D. are in pure Mathematics from the National University of Ireland. His research interests include cryptography, network security, smart cards, and numerical computing.

Adam Duffy received his B.Sc. degree in Computer Science from the National University of Ireland, Maynooth in 2001. He is currently pursuing his Ph.D. degree. His research interests include applied cryptography and software engineering.