

Number 640



**UNIVERSITY OF
CAMBRIDGE**

Computer Laboratory

Combining cryptography with biometrics effectively

Feng Hao, Ross Anderson, John Daugman

July 2005

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
<http://www.cl.cam.ac.uk/>

© 2005 Feng Hao, Ross Anderson, John Daugman

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

<http://www.cl.cam.ac.uk/TechReports/>

ISSN 1476-2986

Combining cryptography with biometrics effectively

Feng Hao, Ross Anderson, John Daugman

Computer Laboratory, Cambridge University, UK
{Feng.Hao, Ross.Anderson, John.Daugman}@cl.cam.ac.uk

Abstract

We propose the first practical and secure way to integrate the iris biometric into cryptographic applications. A repeatable binary string, which we call a biometric key, is generated reliably from genuine iris codes. A well-known difficulty has been how to cope with the 10 to 20% of error bits within an iris code and derive an error-free key. To solve this problem, we carefully studied the error patterns within iris codes, and devised a two-layer error correction technique that combines Hadamard and Reed-Solomon codes. The key is generated from a subject's iris image with the aid of auxiliary error-correction data, which do not reveal the key, and can be saved in a tamper-resistant token such as a smart card. The reproduction of the key depends on two factors: the iris biometric and the token. The attacker has to procure both of them to compromise the key. We evaluated our technique using iris samples from 70 different eyes, with 10 samples from each eye. We found that an error-free key can be reproduced reliably from genuine iris codes with a 99.5% success rate. We can generate up to 140 bits of biometric key, more than enough for 128-bit AES. The extraction of a repeatable binary string from biometrics opens new possible applications, where a strong binding is required between a person and cryptographic operations. For example, it is possible to identify individuals without maintaining a central database of biometric templates, to which privacy objections might be raised.

1 Introduction

A number of researchers have studied the interaction between biometrics and cryptography, two potentially complementary security technologies. Biometrics is about measuring unique personal features, such as a subject's voice, fingerprint, or iris. It has the potential to identify individuals with a high degree of assurance, thus providing a foundation for trust. Cryptography, on the other hand, concerns itself with the projection of trust: with taking trust from where it exists to where it is needed.

A strong combination of biometrics and cryptography might, for example, have the potential to link a user with a digital signature she created with a high level of assurance. For example, it will become harder to use a stolen token to generate a signature, or for a user to falsely repudiate a signature by claiming that the token was stolen when it was not. Previous attempts in this direction include a signature-verification pen and associated

signal processor made available with the IBM Transaction Security System in 1989 [4]. One problem with this approach is its complete reliance on hardware tamper-resistance: if the token is broken, both the template and the key are lost. In many cases, attackers have been able to break tokens, whether by hardware attacks exploiting chip-testing technology, or (as with the IBM design) by API attacks on the token's software [1]. We therefore set out to find a better way of combining biometrics, cryptography and tamper-resistance.

The main obstacle to algorithmic combination is that biometric data are noisy; only an approximate match can be expected to a stored template. Cryptography, on the other hand, requires that keys be exactly right, or protocols will fail. For that reason, previous product offerings have been based on specific hardware devices. It would be better to have a more general, protocol-level approach, combining cryptography and biometrics. Yet another consideration is privacy. Many users may be reluctant to have biometric data stored on central databases; and there may be less resistance to biometric technology if users can be credibly assured that their templates are not stored centrally (or, perhaps, at all).

Other researchers have tried to map biometric data into a unique and repeatable binary string [7, 10, 11, 8, 9]. Subsequently, the binary string would be mapped to an encryption key by referring to a look-up table [7, 8, 9], or direct hashing [10, 13]. The potential of this approach is that storage of a biometric template would not be needed. So far, however, these attempts have suffered from several drawbacks, which we will now explain. In the paper, we will use the term *biometric key*, proposed in [2], to refer to the repeatable string derived from a user biometric.

The hardest problem with biometrics is the unreliability of individual bits in the template. Biometric measurements, being made of attributes of the human body, are noisy by nature, while cryptography demands correctness in keys. There have been a number of attempts to bridge the gap between the fuzziness of biometrics and the exactitude of cryptography, by deriving biometric keys from key stroke patterns [11], the human voice [8], handwritten signatures [10], fingerprints [7, 5], and facial characteristics [9]. However, so far, these attempts have suffered from an excessive False Rejection Rate (FRR) – usually over 20%, which is unacceptable for practical applications [6].

Second, many proposals have failed to consider security engineering aspects, of which the most severe are the irrevocability of biometrics and their low level of secrecy [6]. Biometric features are inherent in individuals, so they cannot be changed easily. A related problem is key diversity: an individual may wish separate keys for their bank account and for access to their workplace computer, so that they can revoke one without affecting the other.

Third, biometric data are not very secret. People leave (poor-quality) fingerprints everywhere, and iris images may be captured by a hidden camera. Generally speaking, the more a biometric is used, the less secret it will be [1]. It would be imprudent to rely on a biometric alone, especially if that biometric became used on a global scale (for example, in the biometric identity cards proposed in some countries). One might expect Mafia-owned businesses to collect biometric data in large quantities if there was any potential exploit path.

Fourth, social acceptance is crucially important to the success of biometric technology [6]. The fear of potential misuse of biometric data may make the public reluctant to use systems that depend on it, and this could be especially the case if there is a large

central database of biometric data which focuses privacy worries and acts as a target of privacy activists. There may be a fear that personal health information will leak out via biometric data, and there may even be religious objections [1].

Finally, we specifically studied the problem of deriving a biometric key from iris codes, as they are at present the most reliable biometric and have the greatest power to distinguish individual persons. There is one previous paper – by Davida, Frankel, Matt and Peralta – proposing to derive a key from iris code using error-correction codes [16]. But no concrete implementation work was reported, and we found that majority coding does not work with real iris data as errors are strongly correlated. We discuss this in detail below.

We therefore set out to design a system in which we do not need to store a biometric template, but only a string of error-correction data from which the biometric cannot be derived, and from which the key cannot be derived either unless the biometric is present. We present a two-factor scheme, relying on the biometric and a token, and also show how it can be easily extended to a three-factor scheme with a password as well. In each case we argue that the protection is the best achievable given the limitations of the components: all factors are needed to compromise the key. In addition, the key can be easily updated or revoked. Finally, we aim to provide a system with a false rejection rate good enough for real use.

2 Past work

We now provide a more detailed survey of recent research on extracting biometric keys [7, 8, 9, 10, 11, 12]. Monroe, Reiter, Li and Wetzel were among the first: their system [11] is based on key-stroke dynamics. A short binary string is derived from the user’s typing patterns and then combined with her password to form a hardened password. Each key-stroke feature is discretized as a single bit, which allows some error tolerance for feature variation. The short string is formed by concatenating the bits. In a follow-up paper, Monroe, Reiter and Wetzel proposed a more reliable implementation based on voice biometrics, but with the same discretization methodology [8]. Their paper reports an improvement in performance: the entropy of the biometric key is increased from 12 bits to 46 bits, while the false rejection rate falls from 48.4% to 20% [8].

Hao and Chan made use of handwritten signatures in [10]. They defined forty-three signature features extracted from dynamic information like velocity, pressure, altitude and azimuth. Feature coding was used to quantize each feature into bits, which were concatenated to form a binary string. This achieved on average 40-bit key entropy with a 28% false rejection rate; the false acceptance rate was about 1.2% [10].

Fingerprints are among the more reliable biometrics, and there is a long history of their use in criminal cases [1]. Soutar, Roberge, Stoianov, Gilroy and Vijaya Kumar reported a biometric-key system based on fingerprints in [12] and were the first to commercialize this technology into a product – *Bioscrypt* (see www.bioscrypt.com). They extract phase information from the fingerprint image using a Fourier transform and apply majority coding to reduce the feature variation. Instead of generating a key directly from biometrics, they introduce a method of *biometric locking*: a pre-defined random key is locked with a biometric sample by forming a phase-phase product. This product can be unlocked by another genuine biometric sample. Biometric locking appears a promising idea, because

the biometric key can be randomly defined. However, performance data are not reported.

Clancy, Kiyavash and Lin proposed a similar application based on fingerprints in [7] and used a technique called a *fuzzy vault*, which had been first introduced by Juels and Sudan [15]. In Clancy's work, the fingerprint minutiae locations are recorded as real points which form a *locking set*. A secret key can be derived from this through polynomial reconstruction. In addition, chaff points are added to the locking set to obscure the key. If a new biometric sample has a substantial overlap with the locking set, the secret key can be recovered by a Reed-Solomon code. This work is reported to derive a 69-bit biometric key but unfortunately with 30% false rejection rate.

Goh and Ngo combined some of the above techniques to build a system based on face biometrics [9]. They adopted the *biometric locking* approach used by Soutar et al. Eigen-projections are extracted from the face image as features, each of which is then mixed with a random string and quantized into a single bit. A binary key is formed by concatenating these bits, and majority-coding is added as suggested by Davida et al. Error correction involves polynomial thresholding which further reduces feature variance. Goh and Ngo report extracting 80-bit keys with a 0.93% false rejection rate. This is beginning to approach the parameters needed for a practical system. However, the experiments reported are based on images taken from a continuous video source with minor variations, rather than a face database. So doubts remain about the evaluation of this work.

In summary, a range of biometrics have been used in previous practical work. With the exception of Goh and Ngo's paper, the false rejection rates are over 20%, which is way beyond the level acceptable for practical use. In addition, the key lengths are usually too short.

There is also some theoretical work on key extraction from noisy data. The *fuzzy extractor* is a recently proposed primitive to extract strong keys from noisy data such as biometrics [21]. In this proposal, Dodis, Reyzin and Smith apply an error-correction code to the input, followed by a hash function, and prove that the information leakage from the input data into the output of the hash function is negligible.

This sort of approach can be useful if the noisy data can be kept secret. However, biometric applications lie between the extremes of secret data and fully public data. People leave behind fingerprints, and their irises can be photographed surreptitiously; a biometric sample stolen in this way will reveal most of its entropy to the attacker.

A related issue is issuing multiple keys for different applications. The fuzzy extractor scheme was modified by Boyen [22], in that a fixed permutation is applied to the iris-code bits before hashing. The compromise of one key derived from an individual's biometric does not compromise any other key derived from the same biometric using a different permutation. But this revised design still assumes that biometric data remain secret, and it fails completely whenever the original biometric is stolen.

The third theory paper is by Juels and Wattenberg. Their *fuzzy commitment scheme* starts out with a random key, adds redundancy, and XOR's this with the iris code [14]. So the key is completely independent of the biometric data.

Our scheme is somewhat similar to theirs but with a number of important differences. First, we have developed a concrete coding scheme that works well with real iris data. None of the papers so far, whether practical or theoretical, have solved this critical engineering problem. Second, we add an auxiliary secret – a password – and an interaction with a token such as a tamper-resistant smartcard. We designed our scheme to give the

best security available given the limitations of these authentication factors – biometrics that might be compromised, passwords that might be guessed, and tokens that might be reverse-engineered.

3 Algorithms

In this section, we present the detailed design of our coding scheme. The design was driven by the error characteristics of iris codes, which are 256-byte strings of phase information derived from an infrared image of an iris by demodulating it with complex-valued 2D-Gabor wavelets [3]. The errors, seen as the differences between different observations of the same iris, are of two types. First, there is a background of random errors, due to CCD camera pixel noise, iris distortion and image-capture effects that cannot be effectively corrected by the preparatory signal processing. Second, there are burst errors, due largely to undetected eyelashes and specular reflections, whether from the cornea or from spectacles. Efforts are made by the standard Daugman algorithms to identify these; along with the string representing the iris code, the software returns a mask string indicating those bits that are considered suspect. However, the identification of eyelashes and reflections is not perfect; faint reflections and out-of-focus eyelashes in particular lead to burst errors.

Majority coding was suggested in some past work to remove errors [12, 16]. We found it does not work at all well with iris data, because multiple scanning does not improve the bit error rate very much. A faint reflection or an out-of-focus eyelash can easily give similar errors on successive scans. We found that with a corpus of images of 70 users, without using masking, an average bit error rate of 13.69% for single-scan iris-code acquisition was reduced to 10.68% after 3 scans, and 9.36% after 5 scans. To deal with such persistent errors, we use a concatenated-coding scheme in which the background-noise errors are first corrected using a Hadamard code, and the burst errors are then corrected using a Reed-Solomon code.

3.1 Basic scheme

We will first describe a basic two-factor scheme without a password. The key depends on a combination of a biometric and a token, which stores error-correction information. We assume it is difficult for the attacker to procure both factors, and we will initially assume if the attacker obtains the token, he will have the full knowledge of the data stored on it. The initial design goal is thus to ensure that the compromise of a single factor will not reveal the key. In the next section, we will show how to extend the scheme to three factors by adding a user password, and we will also consider two levels of attacker: a common attacker who can merely use a token if he steals it, and a highly-skilled attacker who can extract all the secrets from a stolen token.

Figure 1 shows an overall picture of our design. To bridge the gap between the fuzziness of iris biometric and the exactitude of cryptography, we use a two-layer error correction method. The outer layer uses a Hadamard code to correct random errors at the binary level, while the inner layer uses a Reed-Solomon code to correct errors at the block level, i.e. burst errors.

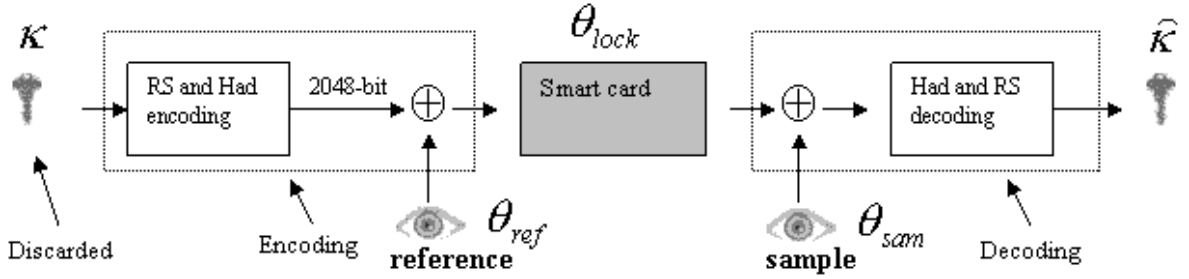


Figure 1: A two-factor scheme for biometric key generation

We first generate the biometric key κ as a string of random bits. It is then encoded with our concatenated code to get what we call a *pseudo-iris code* θ_{ps} . This looks like an iris code because it has the same length as the real iris code, namely 2048 bits. It will be “locked” by XORing it with the user’s reference iris code θ_{ref} , obtained on enrolment:

$$\theta_{lock} = \theta_{ps} \oplus \theta_{ref} \quad (1)$$

The θ_{lock} data will be saved in the smartcard or other physical token \mathcal{T} , together with a hash value of the key, $H(\kappa)$. Subsequently the key κ must be securely erased. The encoding process can be formalised as

$$\langle \kappa, \theta_{ref} \rangle \Rightarrow \mathcal{T} : \{ \theta_{lock}, H(\kappa) \} \quad (2)$$

During the decoding phase, the user presents his iris sample θ_{sam} to “unlock” the key. After XORing with the θ_{lock} data on the smart card, it is then decoded with Hadamard and RS codes in turn to output a biometric key $\hat{\kappa}$. If the hash of the $\hat{\kappa}$ matches the stored hash, i.e. $H(\hat{\kappa}) = H(\kappa)$, the derived key is correct. Otherwise, the key will be deemed false and rejected. The decoding process can be formalised as

$$\langle \theta_{sam}, \mathcal{T} \rangle \Rightarrow \hat{\kappa} \quad (3)$$

In the following sections, we will explain the specific Hadamard and Reed-Solomon codes we use in detail, and show how they can be integrated to achieve our goal. Their choice is based on a detailed study of iris-code error patterns. Iris codes from the same eye usually disagree in 10–20% of the bits [3]. On the other hand, the disagreement of inter-personal iris codes, or the codes for different eyes from the same person, is usually 40–60%. The coding must be able to correct the differences between error bits of iris codes for the same eye, but unable to correct the differences between different eyes. We chose a Hadamard code that can correct about 25% of the error bits in a block [18], which approximately separates same-eye and different-eye error rates. We then fine-tune the scheme with a Reed-Solomon code that can correct for six block errors out of 32.

3.2 Hadamard codes

A Hadamard code is generated by a Hadamard matrix, a square orthogonal matrix with elements 1 and -1 . Orthogonality means that the inner product of any two distinct rows or columns is always 0. The size of a Hadamard matrix must be $\{1, 2, 4m\}$ for natural

numbers m . There are several ways to generate Hadamard matrices; we have chosen the Sylvester method, which recursively defines normalized matrices whose size is a power of 2, $n = 2^k$ [18].

The simplest Hadamard matrix, whose order $k = 1$, is

$$H_1 = \begin{bmatrix} + & + \\ + & - \end{bmatrix} \quad (4)$$

where we use “+” to denote “1” and “-” to denote “-1”. With Sylvester’s method, we obtain further Hadamard matrices recursively by:

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix} \quad (5)$$

To construct a Hadamard code, we obtain a Hadamard matrix H and then cascade H and $-H$ as follows:

$$H_c = \begin{pmatrix} H \\ -H \end{pmatrix} \quad (6)$$

We get the list of codewords by replacing each -1 by 0 in the rows of H_c . A Hadamard matrix of size $n = 2^k$ has $2n$ codewords. The code has minimum distance 2^{k-1} and hence corrects up to $2^{k-2} - 1$ errors.

The encoding process is to encode an input value i into a codeword w . The matrix of H_c has $2n$ rows, each of which is a codeword. The value i can be seen as a row index, which ranges within $[0, 2n - 1]$. The binary representation of i comprises of $\log 2n$ bits. It can be shown that with the Sylvester method, the encoding is efficient, because $\log 2n$ equals an integer $(k + 1)$. The output codeword is the row selected by the index i , which has a size of n bits. To summarize, the function encodes an input block of $(k + 1)$ bits into an output block of $n = 2^k$ bits.

In the decoding process, let the received codeword be w . In w , replace every 0 with -1 to get v . Then calculate

$$vH_c^T = (a_0, a_1, \dots, a_r, \dots, a_{n-1}) \quad (7)$$

From the resultant output, find the position r where the value a_r is maximum. This position will be unique if at most $2^{k-2} - 1$ errors have occurred. If the decoding is correct, we should have the decoded value equal to the input value, i.e., $r = i$. More details about Hadamard error correction can be found in [18].

3.3 Reed-Solomon code

As explained in Section 3.2, the Hadamard code encodes each block of $k + 1$ bits input into one of 2^k bits. We will see below that a suitable choice of k is 6; so it can correct up to 15 errors in each block of 64 bits. This is sufficient to deal with the background errors, but is inadequate in the face of a burst error caused by an eyelash or specular reflection that is not recognised by the preprocessing software.

The quantity of wrongly-decoded blocks is very small, but if it is greater than zero then the decoded key will be wrong and the cryptography will fail. Hence we need another layer of error correction to deal with block errors. The Reed-Solomon code, whose details

can be found in [19], is a suitable choice. We will now explain how Reed-Solomon coding complements Hadamard coding, and justify our choice of parameters.

3.4 Concatenated encoding and decoding

Recall that we use Reed-Solomon coding, then Hadamard coding, to encode a random key κ as shown in Figure 1. The Reed-Solomon code is denoted as $\mathcal{RS}(n_s, k_s, t_s)$, where k_s represents the number of blocks before encoding and n_s represents the number of blocks after encoding. The t_s is the number of error blocks that can be corrected. By the Berlekamp-Massey algorithm [19], we get $n_s - k_s = 2t_s$.

The size of each block for $\mathcal{RS}(n_s, k_s, t_s)$ at both input and output is m . After this code, each m -bit block will be further encoded with the Hadamard code, $\mathcal{HC}(k)$, where k is the order of the matrix. For the two codes to operate on the same blocks, we need $m = k + 1$.

After Hadamard encoding, we obtain a pseudo-iris-code θ_{ps} , where $\|\theta_{\text{ps}}\| = 2048$. We XOR this with a reference iris code θ_{ref} to get a locked code θ_{lock} , which is then saved in the token.

$$\theta_{\text{lock}} = \theta_{\text{ps}} \oplus \theta_{\text{ref}} \quad (8)$$

We call it a locked code, because by itself it cannot be used to deduce either the iris code or the biometric key. Note that correlations exist among iris bits, which reduces the randomness of θ_{ref} [3]. In practice, however, this has a limited impact on security, as an attacker will not in general know which bits are correlated without knowing the subject's actual iris code. We will analyze this further in Section 4.3.

The decoding process involves XORing the locked iris code θ_{lock} with a presented sample θ_{sam} .

$$\begin{aligned} \theta'_{\text{ps}} &= \theta_{\text{lock}} \oplus \theta_{\text{sam}} \\ &= \theta_{\text{ps}} \oplus e \end{aligned} \quad (9)$$

where e is the error vector between two iris codes. The error correction is applied and recovers a trial value of the biometric key $\hat{\kappa}$. If the error e is within its correction capability, $\hat{\kappa} = \kappa$, which we can verify by comparing the hash values. Otherwise, the key will be rejected. We will show in Section 4.2, the error e is correctable for most genuine iris codes, but uncorrectable for different iris codes.

The bit-length of the key κ is given by the following equation:

$$\|\kappa\| = (k + 1) \times \left(\frac{2048}{2^k} - 2t_s \right) \quad (10)$$

In our implementation, we correct for 6 block errors and up to 25% bit errors in the other blocks. This means that for the Reed-Solomon code $t_s = 6$ and for the Hadamard code $k = 6$. Thus the Hadamard code outputs $2048/2^6 = 32$ blocks of 64 bits, and the Reed-Solomon code outputs 20 blocks. Thus the length of the key κ is 140 bits.

4 Results

In this section, we report an evaluation of our implementation against a database of iris codes. We then proceed to a security analysis, discuss how the scheme can be extended from two factors to three by the addition of a user password, and compare our results against the prior art.

4.1 Iris database

The iris database we used consists of 700 iris samples from 70 different eyes, with 10 samples from each eye. The images were acquired in a laboratory setting using the same camera at a fixed measurement distance. A 256-byte iris code, together with a 256-byte mask, is computed from each iris image using the algorithm reported in [3]. The hamming distance between two iris codes is given there as:

$$HD = \frac{\|(codeA \oplus codeB) \cap maskA \cap maskB\|}{\|maskA \cap maskB\|} \quad (11)$$

The mask filters out bits thought to be unreliable because of eyelashes, reflections, obscure boundary detections, etc. This reduces iris code bit error rates dramatically. We have kept things simple so far by not incorporating masks: they would introduce complexity as at the time of encoding we only know the mask function for the reference sample, not for the image that will be taken at the decoding stage. We intend to incorporate the masks into the error correction scheme later, but for now we use the raw iris codes only. We compute the hamming distance between two iris codes without masks as:

$$HD = \frac{\|codeA \oplus codeB\|}{2048} \quad (12)$$

We chose iris samples from the same eyes to compute the intra-eye hamming distances, and chose samples from different eyes to compute the inter-eye hamming distances. We carried out 241,300 comparisons between different eyes and 3,150 comparisons for the same eyes. The results are shown in Figure 2. Without masks, the mean intra-eye Hamming distance increases from 3.37% to 12.7%, while the mean inter-eye Hamming distance remains relatively unaffected. This makes our work more challenging, as we have to handle more error bits as a result of not using the mask functions.

We also need to deal with iris orientation. This varies due to head tilt, camera angles, torsional eye rotation, etc [3]. In the normal use of the iris recognition algorithms, orientation is readily normalized by cyclically scrolling the iris code by multiples of octet-bits. In our off-line comparisons, we chose the first iris sample from each user as a reference, shifted other observed iris codes seven times by octet-bits and attempted to recover the key each time.

4.2 Key length and error rates

The order of the Hadamard matrix sets a trade-off between error tolerance and key length: from Equation 10, a larger value of k will result in a smaller key length. On the other hand, a larger k means a larger block size which will tolerate more errors. We found that

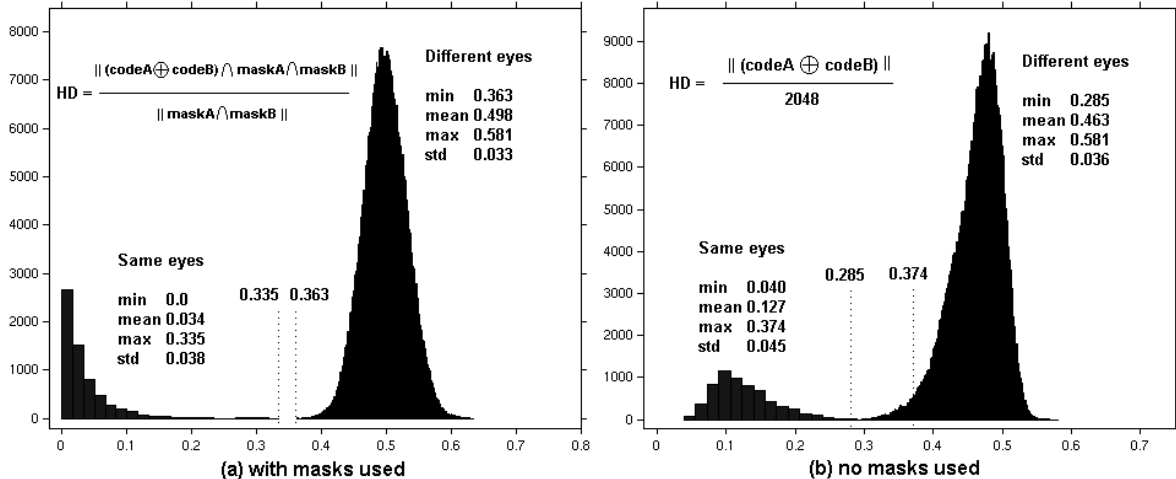


Figure 2: Hamming distance between iris codes

$k = 6$ is a suitable value by experiment. Table 1 shows the performance of error correction for $k = 6$.

As shown in Table 1, the $t_s = 6$ can be a suitable operating point. It generates a biometric key of 140 bits. The corresponding False Rejection Rate is only 0.47% – only 3 among our 630 (70×9) authentic samples were falsely rejected. These three false rejections occurred because of relatively high bit-error rates, above 27% in each case. Iris codes with bit-error rates less than 27% are handled by our coding mechanism quite effectively. Table 2 compares our design with the prior art discussed in Section 2. Our system achieves vastly better performance. The key length is 140 bits, much longer than the 69 bits obtained from fingerprints in [7]. The false rejection rate (0.47%) is much smaller than the 20% common for previous systems. In fact, experience suggests it is about as good as can be achieved from biometric systems used by members of the public; the poor samples are a fact of life in biometric systems and have to be dealt with by other mechanisms, such as retries.

4.3 Security analysis

Our basic design depends on two factors: a biometric and a physical token. If only one factor is compromised, the biometric key remains secure. If the biometric becomes known, this does not help the attacker, because the key is randomly generated.

We make the key completely independent of the iris biometric, as this cannot be kept very secret. However, it is still costly to steal an iris code. A near-infrared camera is needed and it is difficult to capture a person’s iris image close-up without being noticed; most likely, iris code thefts will be conducted using subverted equipment in apparently genuine verification settings. In such a threat model, the attacker would get a password too, if one were in use; so we must rely completely on the token being tamper-resistant.

Let us now consider the contrary case – where the token is stolen, while the iris code remains unknown. We assume that all the internal data in the token are revealed, including the locked iris code $\theta_{\text{lock}} = \theta_{\text{ps}} \oplus \theta_{\text{ref}}$. This is the XOR of a key with redundancy,

RS corrected blocks t_s	key length $\ \kappa\ $	FRR %	FAR %
0	224	12.22	0
1	210	6.50	0
2	196	3.65	0
3	182	2.06	0
4	168	1.26	0
5	154	0.79	0
6	140	0.47	0
7	126	0.15	0
8	112	0.15	0.02
9	98	0	0.04
10	84	0	0.08
11	70	0	0.08
12	56	0	0.12
13	42	0	0.22

Table 1: Performance when $k = 6$

Biometrics	Author	Features	Error handling	Bit-len	FRR	FAR	Ref
Key stroke	Monrose (1999)	Durations, latencies	Discretization	12	48.4%	–	[11]
Voice	Monrose (2001)	Ceptral coefficients	Discretization	46	20%	–	[8]
Signature	Hao (2002)	43 dynamics	Feature coding	40	28%	1.2%	[10]
Fingerprint	Soutar (2002)	Phase after Fourier Transform	Majority code	–	–	–	[12]
Fingerprint	Charles (2003)	Minutiae points	Reed-Solomon code	69	30.0%	–	[7]
Face	Goh (2003)	Face eigen-projections	Discretization and polynomial thresholding	–	–	–	[9]
Iris	– (2005)	Iris code	concatenated coding	140	0.47%	0%	–

Table 2: Summary of biometric key implementations

and a biometric. Correlations exist in every iris: these are mainly caused by the radial structure of furrows, but some further amplitude and phase correlations are introduced by the 2D Gabor wavelet demodulation used to generate an iris code. The critical question is whether these correlations can be used, together with the correlations introduced by the error-correction process, to unlock the key. However, experiments on large corpora of iris codes show that a 2048-bit iris code has 249 degrees of freedom, and that there is little systematic correlation among irises [3].

To try to set a rough lower bound on the difficulty facing an attacker who has obtained the locked code and attempts to reconstruct the key, consider the worst case and assume the attacker has a perfect knowledge of the correlations within the subject’s iris code. Then the uncertainty of the iris code is only 249 bits. Our coding scheme allows up to 27% of the bits to be wrong, so the attacker is trying to find a 249-bit string within 67 bits Hamming distance of the key. Let $z = 249$, and $w = 67$. By the sphere-packing bound [19]:

$$\begin{aligned}
 \mathcal{BF} &\geq \frac{2^z}{\sum_{i=0}^w \binom{z}{i}} \\
 &\approx \frac{2^z}{\binom{z}{w}} \\
 &= 2^{44}
 \end{aligned} \tag{13}$$

So such a search will require at least 2^{44} computations. This may seem an alarmingly small number to the crypto purist, now accustomed to thinking of 56 bits as inadequate. Several things need to be said. First, iris codes currently give – by a large margin – the most secure biometric available. If they are not good enough for an application, then no biometric is. Second, the figure of 2^{44} is a very conservative theoretical bound: if the attacker has no or little knowledge about how the target person’s iris bits are correlated, the effort would be significantly larger, and with our current state of knowledge we really do not know how to correlate someone’s iris bits unless we know their iris code anyway. Third, each of the 2^{44} computations is moderately complex, involving not just coding but also the computation of a hash of the biometric key. If 2^{64} security is sought, one can run the hash function a million times. Finally, security can be significantly strengthened by a third factor – a password – as we will now explain.

4.4 Three-factor scheme

The practical threat to the basic two-factor scheme is that someone obtains the target’s iris image using a hidden camera, then steals the token and derives the key. A two-factor biometric-key scheme by its nature cannot prevent such attacks. When iris codes are used in typical subject-identification applications, there are further options, such as cameras that distinguish between living and fake eyes. One such – the LG-3000 IrisAccess camera – uses a set of 16 “liveness detection” countermeasures and has been certified by the Australian Ministry of Defence. Another possibility is to insist on attended operation. Interesting as these issues may be – and there may well be an arms race between defenders and attackers – liveness detection is of limited help if we wish to assume that the attacker will use his own camera and understand the iris-scanning process.

For applications where the threat model demands it, a password may be incorporated to give a three-factor scheme. There are various ways to do this. Ideally we want to prevent any short-cuts; an attacker trying to search for a biometric key given a guessable password should have to expend an effort equal to the product of the key search effort and the password-guessing effort. One simple way is to use passwords to encrypt the locked iris code. A more interesting option is to permute the Hadamard matrix: row/column permutations turn one Hadamard matrix into another. Thus the matrix of size 64 that we used to construct our code can give rise to $64! \times 64!$, or 2^{592} , different matrices through permutation. Permuting the Hadamard matrix also makes the encoded data θ_{ps} appear random (see Equation 8), which would minimize the entropy leakage of the key and raise the lower-bound brute-force effort attacking on the key.

An important security-engineering aspect is to prevent the industrialisation of attacks (as has for example occurred with Trojan attachments to autoteller machines that read a magnetic-strip card as it is entered into the equipment, and also record PIN entry using a pinhole camera). Once any token-based authentication scheme comes into wide use, individual attacks on it can be expected: users will be simply tricked into authenticating transactions they should not have. However, industrial attacks should be prevented. Our scheme will force an attacker who wishes to misuse the keys of a large number of users to arrange to confiscate their tokens, to obtain high-quality photographs of their irises, and to solicit their passwords too if passwords are used. This is a much tougher challenge. It is also highly significant that one user can be issued with a number of different biometric keys for different applications. The use of a simple biometric database for (say) both banking and national-ID purposes might entail that an attack on the bank yielded an attack on national ID, and vice versa. With our design, this no longer has to be the case.

Finally, revocation is critical to good security engineering. Many of the earlier biometric-key schemes are incapable of it, as the key is derived directly from the biometric data, and are thus not usable in their existing form. Our scheme shows how to do revocation in a system based on biometrics.

4.5 Privacy and identity

The acquisition of a repeatable string from iris biometric opens up new opportunities for privacy. One current debate concerns the possible privacy abuses of biometric databases collected to support applications such as ID cards. This prospect has started to raise a number of concerns, ranging from the possibility that biometric data might be correlated with health and thus leak health information (which in the case of iris codes appears limited to gross conditions such as cataracts), to religious concerns.

Our work shows that high-quality identification of persons is possible using biometric means but without a central database of templates. The subject would present at an enrolment station with foundational identifying materials such as a passport, and have an iris scanned. The biometric data need not be retained by the issuing authority. The enrolment station could use the generated biometric key to protect a Kerberos key shared with an authentication service, or to protect a private digital-signature key whose public verification key is linked to their distinguished name by an X.509 certificate. This is relatively well-understood technology, and lies outside the scope of the discussion here.

5 Conclusion

In this paper, we tackled the most difficult problem for merging cryptography and biometrics: how to generate a repeatable string from a biometric in such a way that it can be revoked. Previous attempts have almost all had quite unacceptable false-reject rates. Most of them also have problems with revocation, have produced too-short keys, and have not been well-tested. We have shown how to generate keys robustly from iris biometric measurements, using associated error-correction data that can be changed to yield different keys. Our scheme produces long enough keys; it can produce different keys for different applications, so that an attack on one does not give an attack on all; it supports revocation; its security case is founded on extensive research in the application area, as well as a statistical lower-bound argument; and we have shown that its false-reject rate is under half a percent. This makes it feasible, we believe, for many practical uses.

Acknowledgment

We thank David Wheeler for very helpful discussions on Hadamard error correction codes.

References

- [1] R.J. Anderson, *Security Engineering : A Guide to Building Dependable Distributed Systems*, New York, Wiley 2001.
- [2] J. Daugman, "Biometric decision landscapes," Technical Report UCAM-CL-TR-482, Computer Laboratory, University of Cambridge, 2000.
- [3] J. Daugman, "The importance of being random: statistical principles of iris recognition," *Pattern Recognition*, Vol. 36, No. 2, pp. 279–291, 2003.
- [4] D.G. Abraham, G.M. Dolan, G.P. Double, J.V. Stevens, "Transaction Security System," *IBM Systems Journal*, Vol. 30, No.2 pp. 206–229, 1991.
- [5] Y. Seto, "Development of personal authentication systems using fingerprint with smart cards and digital signature technologies," the Seventh International Conference on Control, Automation, Robotics and Vision, Dec 2002.
- [6] U. Uludag, S. Pankanti, S. Prabhakar and A. K. Jain, "Biometric cryptosystems: issues and challenges," *Proceedings of the IEEE*, Vol. 92, No. 6, pp. 948–960, 2004.
- [7] T.C. Clancy, N. Kiyavash and D.J. Lin, "Secure smart card-based fingerprint authentication," *Proceedings of the 2003 ACM SIGMM Workshop on Biometrics Methods and Application*, WBMA 2003.
- [8] F. Monrose, M.K. Reiter, Q. Li and S. Wetzel, "Cryptographic key generation from voice," *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.
- [9] A. Goh, D.C.L. Ngo, "Computation of cryptographic keys from face biometrics," *International Federation for Information Processing 2003*, Springer-Verlag, LNCS 2828, pp. 1–13, 2003.

- [10] F. Hao, C.W. Chan, “Private key generation from on-line handwritten signatures,” *Information Management & Computer Security*, Issue 10, No. 2, pp. 159–164, 2002.
- [11] F. Monrose, M.K. Reiter and R. Wetzel, “Password hardening based on keystroke dynamics,” Proceedings of sixth ACM Conference on Computer and Communications Security, CCCS 1999.
- [12] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy and B.V.K. Vijaya Kumar, “Biometric Encryption,” *ICSA Guide to Cryptography*, McGraw-Hill, 1999, also available at http://www.bioscrypt.com/assets/Biometric_Encryption.pdf
- [13] K.J. Pawan and M.Y. Siyal, “Novel biometric digital signature for Internet based applications,” *Information Management and Computer Security*, Vol. 9, No. 5, pp. 205–212, 2001.
- [14] A. Juels and M. Wattenberg, “A fuzzy commitment scheme,” Proceeding of the 6th ACM Conference on Computer and Communication Security, CCCS, 1999.
- [15] A. Juels and M. Sudan, “A fuzzy vault scheme,” Proceedings of IEEE International Symposium on Information Theory, 2002.
- [16] D. I. Davida, Y. Frankel, B.J. Matt and R. Peralta, “On the relation of error correction and cryptography to an off line biometrics based identification scheme,” Workshop on Coding and Cryptography, 1999.
- [17] David Wheeler, “Protocols using keys from faulty data,” Security Protocols Workshop, Cambridge, 2001.
- [18] S.S. Aghaian, *Hadamard Matrix and Their Applications*, LNM, Springer Verlag, 1985.
- [19] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-correcting Codes*, North Holland, 1991.
- [20] R.J. McEliece, *The Theory of Information and Coding*, Cambridge University Press, 2002.
- [21] Y. Dodis, L. Reyzin, A. Smith, “Fuzzy extractors: how to generate strong keys from biometrics and other noisy data,” Eurocrypt 2004, Lecture Notes in Computer Science 3027, pages 523–540.
- [22] X. Boyen, “Reusable cryptographic fuzzy extractors,” CCS 2004, pp. 82–91, ACM Press.