Technical Report

UCAM-CL-TR-715
ISSN 1476-2986

Number 715

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

On using fuzzy data in security mechanisms

Feng Hao

April 2008

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted April 2007 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Queens' College.

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

Abstract

Under the microscope, every physical object has unique features. It is impossible to clone an object, reproducing exactly the same physical traits. This unclonability principle has been applied in many security applications. For example, the science of biometrics is about measuring unique personal features. It can authenticate individuals with a high level of assurance. Similarly, a paper document can be identified by measuring its unique physical properties, such as randomly-interleaving fiber structure.

Unfortunately, when physical measurements are involved, errors arise inevitably and the obtained data are fuzzy by nature. This causes two main problems: 1) fuzzy data cannot be used as a cryptographic key, as cryptography demands the key be precise; 2) fuzzy data cannot be sorted easily, which prevents efficient information retrieval. In addition, biometric measurements create a strong binding between a person and his unique features, which may conflict with personal privacy. In this dissertation, we study these problems in detail and propose solutions.

First, we propose a scheme to derive error-free keys from fuzzy data, such as iris codes. There are two types of errors within iris codes: background-noise errors and burst errors. Accordingly, we devise a two-layer error correction technique, which first corrects the background-noise errors using a Hadamard code, then the burst errors using a Reed-Solomon code. Based on a database of 700 iris images, we demonstrate that an error-free key of 140 bits can be reliably reproduced from genuine iris codes with a 99.5% success rate. In addition, despite the irrevocability of the underlying biometric data, the keys produced using our technique can be easily revoked or updated.

Second, we address the search problem for a large fuzzy database that stores iris codes or data with a similar structure. Currently, the algorithm used in all public deployments of iris recognition is to search exhaustively through a database of iris codes, looking for a match that is close enough. We propose a much more efficient search algorithm: Beacon Guided Search (BGS). BGS works by indexing iris codes, adopting a "multiple colliding segments principle" along with an early termination strategy to reduce the search range dramatically. We evaluate this algorithm using 632,500 real-world iris codes, showing a substantial speed-up over exhaustive search with a negligible loss of precision. In addition, we demonstrate that our empirical findings match theoretical analysis.

Finally, we study the anonymous-veto problem, which is more commonly known as the Dining Cryptographers problem: how to perform a secure multiparty computation of the boolean-OR function, while preserving the privacy of each input bit. The solution to this problem has general applications in security going way beyond biometrics. Even though there have been several solutions presented over the past 20 years, we propose a new solution called: Anonymous Veto Network (AV-net). Compared with past work, the AV-net protocol provides the strongest protection of each delegate's privacy against

collusion; it requires only two rounds of broadcast, fewer than any other solutions; the computational load and bandwidth usage are the lowest among the available techniques; and our protocol does not require any private channels or third parties. Overall, it seems unlikely that, with the same underlying technology, there can be any other solutions significantly more efficient than ours.

Acknowledgments

My supervisors, Prof. Ross Anderson and Dr. John Daugman, guided me through this research project. I sincerely appreciate their teaching and encouragement along the way. It was also fortunate for me to be in the Security Group, which provides arable grounds for learning. My heartfelt "thanks" goes to all group members for stimulating discussions, countless tea breaks, and having fun together.

During this research project, it was my pleasure to work with several people, who also contributed to this thesis. In Chapter 3, Ross Anderson contributed insights on applying the devised technique to various application areas, including national-ID cards. John Daugman provided iris databases; he also explained various aspects of iris recognition, including iris-code correlation and practical implementation of exhaustive search. In Chapter 5, Piotr Zieliński improved my initial protocol design by reducing the computation load and bandwidth usage per participant by half.

Finally, all of my work wouldn't be possible without the support from my family. I thank my parents for always believing in me and supporting me. And thanks to my wife, Lihong, for her patience, encouragement and being the first reader of all my papers. This dissertation is dedicated to her.

Contents

Chapter 1

Introduction

This dissertation studies how to use fuzzy data – obtained from physical measurements – to fulfil security requirements. It starts with a survey of different types of fuzzy data. Then, it illustrates the problems arising from the data being fuzzy. To solve these problems, it introduces a range of new techniques, employing error correction codes, cryptography and probability theory. Finally, it suggests further research.

1.1 Motivations

Cryptography is built upon trust: with taking trust from where it exists to where it is needed [1]. Commonly, trust is transferred using number theory. For example, the factorization and discrete logarithm problems are assumed – and widely believed – to be intractable, thus providing a foundation of trust. Many public key schemes, such as the RSA and ElGamal algorithms [16, 17], are based on these assumptions. It is the subject of a large research program to construct encryption and signature schemes that are provably reducible to common number-theoretic problems [18].

However, the approach based on number theory is shadowed by the fact that those assumptions remain unproven [16]. Hence, efficient attacks against public key cryptosystems might still exist; they just have yet to be found. In addition, the likely presence of a quantum computer – which is able to perform factorization in polynomial time [19, 20] – poses a threat to many cryptographic algorithms. If factoring turns out to be easy, then many public-key systems will fail.

There are other but insecure trust bases. Some banking systems trust their internal users (i.e., bank employees), and are thus particularly vulnerable to insider attacks [2, 22]; proprietary encryption methods relying on keeping the algorithms secret were repeatedly broken [1]; a recently proposed "totally secure" communication system [25] assumes technological perfection that temperature can be controlled precisely and wire resistance can be removed completely, but technology can never be perfect [11].

With the increasing reliance on number theory, trust diversity becomes particularly important and urgent.

One alternative is based on *physical unclonability*. For example, the science of 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. On the other hand, passwords and tokens are transferrable between different people, and hence

cannot ensure authentication down to the real person.

Physical uniqueness is not confined to the human body – in fact, every physical object is measurably unique and cannot be cloned identically. Some researchers proposed to measure the microscopic imperfection of the paper surface by using a camera [33] or laser [26]. Thus, the physical properties of a paper document can be used as its unclonable identifier [33, 26]. This is much more secure than the conventional bar-code-based approach to product identification, since bar codes can be easily counterfeited (e.g., by making photocopies).

Pappu et al. applied the unclonability principle to build a physical one-way function – an epoxy token that displays different speckle patterns when measured in different ways [28, 29]. The security of this one-way function relies on the physical unclonability of the token rather than number theory. The unclonability principle is also seen in quantum cryptography, which works on the impossibility of cloning quantum information [28] (also see [21]).

Unfortunately, when physical measurements are involved, errors are inevitable, and the obtained data are fuzzy. Iris codes, for example, usually have a bit error rate between 10 to 20% [4]. It is unlikely that multiple measurements of the same eye give exactly the same results. Similarly, the physical one-way functions and quantum communication are error-prone too [28, 21].

The unavoidable measurement errors make it difficult to integrate the obtained results into cryptographic operations. In some applications, it is desirable to use the measurement of a physical object as a cryptographic key [48], but cryptography does not accommodate keys that contain errors [53]. Matching two fuzzy strings depends on their similarity rather than exact equality [60]. But two similar strings will become completely different after applying one-way hash functions [1]. This implies that while in Unix systems, one can protect the privacy of passwords by only storing the hashes of passwords, the same measure does not work with fuzzy data.

Furthermore, it is difficult to sort data containing random errors. Many applications, on the other hand, require data be stored in a structured way so that information retrieval is fast. However, as we will explain in Chapter 4, efficient sorting algorithms for high-dimensional fuzzy data – such as iris codes – are lacking in past work. The algorithm used in all current public deployments of iris recognition is to search exhaustively through a database of iris codes, looking for a match that is close enough. But such an exhaustive search approach is not efficient and cannot well scale up to a large database (see Chapter 4).

Finally, privacy issues should be addressed. The physical unclonability principle aims to create a strong binding between objects and their measurements. However, in some cases, such binding itself may conflict with personal privacy. For example, people may not want to have their biometric features measured or stored at all.

1.2 Contributions

In this dissertation, we will examine the above problems in more detail, and have the following contributions.

1. Propose a two-layer error correction scheme that can derive a 140-bit error-free key

from genuine iris codes with a 99.5% success rate. With this technique, high-quality identification of persons can be performed using biometric means but without a central database of templates.

2. Propose a fast search algorithm for a large iris-code database. This technique is evaluated using 632,500 real-world iris codes, showing a substantial improvement in search speed over exhaustive search with a negligible loss of precision.

3. Propose a protocol that allows participants to anonymously veto a decision without relying on any trusted third parties or private channels. Essentially, this technique presents a new solution to the Dining Cryptographers problem [73], achieving the best efficiency among all available solutions.

1.3 Thesis outline

Chapter 2 reviews a range of applications that involve physical measurements and produce fuzzy data. Additionally, we show a new paper-fingerprinting method, which uses only an ordinary camera to capture the unique translucency patterns of a paper document. This method highlights the potential of using general-purpose cameras to capture an object's unique features at low cost.

Chapter 3 studies how to effectively combine biometrics with cryptography. We propose an error correction technique to derive error-free keys from biometric data. 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 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. Based on a database of 700 iris images, we show that our technique can produce an error-free key of 140 bits from genuine iris codes with a 99.5% success rate.

Chapter 4 studies the search problem for a large fuzzy database that stores iris codes or data with a similar structure. Currently, all public deployments of iris recognition adopt an exhaustive search approach [7], looking for a match among an enrolled database. Our new technique, Beacon Guided Search (BGS), tackles this problem by dispersing a multitude of "beacons" in the search space. Despite random bit errors, iris codes from the same eye are more likely to collide with the same beacons than those from different eyes. By counting the number of collisions, BGS shrinks the search range dramatically with a negligible loss of precision. We evaluate this technique using 632,500 iris codes enrolled in the United Arab Emirates (UAE) border control system, showing a substantial improvement in search speed with a negligible loss of accuracy. In addition, we demonstrate that the empirical results match theoretical predictions.

Chapter 5 studies the veto problem in a biometrically-enabled threshold control scheme. This problem requires a secure multiparty computation on the boolean-OR function, while preserving the privacy of each input bit. We propose an efficient solution: Anonymous Veto Network (AV-net). The AV-net construction is provably secure under the Decision Diffie-Hellman assumption, and is better than past work in the following ways. It provides the strongest protection of each input's privacy against collusion; it requires only two rounds of broadcast, fewer than any other solutions; the computational load and

bandwidth usage are the least among the available techniques; and our protocol does not require any private channels or third parties.

Chapter 6 concludes this dissertation and suggests future research.

Chapter 2

Fuzzy data

This chapter reviews a range of applications that involve physical measurements and produce fuzzy data, such as iris codes, physical one-way functions, paper fingerprints, and audio fingerprints. We summarize the common features, as well as common problems, of these fuzzy data. In addition, we demonstrate how to use only an ordinary camera to fingerprint a piece of paper for later auto-recognition. This demonstration shows that an object's unique physical traits can be captured at low cost.

2.1 Survey

In this section, we will review several types of fuzzy data, obtained from measuring human irises, epoxy tokens, paper documents, and audio clips (Table 2.1). All these data are represented in the form of binary strings. In the survey, we exclude fuzzy data that consist of (floating-point) scalars and require non-linear alignments, e.g., fingerprints [1]. This is because the techniques presented in Chapter 3 and 4 will not be directly applicable to such data.

First, we study the iris biometric [5]. Human irises contain random texture patterns, which can be captured by a camera at close distances up to 1 m. The photographing process is assisted by near infrared (NIR) illumination, which appears unintrusive to human eyes and helps reveal rich and complex stromal features of irises.

The recorded iris patterns need to be encoded for efficient comparison and storage [5]. Daugman's algorithm is the standard technique to represent the random iris patterns with a 256-byte binary code, called the iris code[1]. Statistical analysis reveals that an iris code using Daugman's algorithm contains about 249 degrees of freedom. The iris measurements are error-prone; iris codes derived from multiple measurements of the same eye typically have about 10 to 20% of bits that differ (also see [4, 6, 7]).

The physical uniqueness is not confined to the human body – in fact, every physical object is measurably unique. Based on the impossibility of cloning physical objects, Pappu et al. built a physical one-way function – an epoxy token that works like a physical random oracle [28]. The token contains tiny glass spheres that are 500 to 800 μm in diameter. When illuminated by a HeNe laser, the token displays a 2D optical speckle-pattern at the

[1]It is written as "IrisCode" in Daugman's papers [4, 6, 7]. However, throughout this thesis, we will use the term "iris code" – which is also used in [36, 1] – to refer to a binary code derived from an iris image.

Data	Source	Bytes	Errors	Entropy	Equipments
Iris code [5]	Human iris	256	10–20%	249-bit	NIR and monoch. camera
Token fingerprint [28]	Epoxy token	300	25%	233-bit	Laser and camera
Paper fingerprint [33]	Paper	50–300	10%	16-bit	Customized camera
Paper fingerprint [26]	Paper	200–500	$< 30\%$	—	Laser and photodetectors
Audio fingerprint [31]	Audio clips	1024	15%	1141-bit	Microphone recorder

Table 2.1: Summary of fuzzy applications

background, which is then recorded by a camera. Gabor wavelets are applied to encode the optical pattern into a 300-byte fingerprint. Aiming the laser at different spots, and with different angles, produces different optical patterns, giving the derived fingerprints 233 degrees of freedom. The measurement contains unavoidable errors; with the same input parameters, the token produces fingerprints that are similar with an average Hamming distance of 25%.

Métois et al. proposed to identify a piece of paper by analyzing its non-uniform paper surface [33]. A few dots are printed on the paper to mark the area that should be analyzed. The imaging device consists of a customized video camera, housed with the embedded lighting apparatus. Assisted by the reflected light, the camera takes an image of the paper surface. Instead of using the standard 2D image processing techniques as in [5, 28], the authors simply average the raw pixel values over the imaged area, and then quantize the analogue values into a binary string of 50–300 bytes. The information density is reported to be 0.64 bits/mm^2, which gives 20 bits of information over an area of 25 mm^2.

A different paper-fingerprinting technique is proposed in [26], in which Buchanan et al. applied a focused laser beam to scan across the paper surface, and used four photodectors to record the reflected light intensity from four angles. The coordinates of the scanning path are determined with reference to the paper cutting edges. The obtained analogue signal is transformed into a 200 to 500-byte fingerprint through quantization. Repeated scans over the same area give similar fingerprints with less than 0.3 Hamming distances between each other [27].

Finally, audio clips can also be fingerprinted. This technology allows automatic recognition of a piece of played music [31, 32]. Miller et al. used a scheme that encodes each 5-second interval of an audio clip into a 1024-byte fingerprint by applying a Fourier transform [31]. The fingerprint consists of 256 sub-fingerprints, with 4 bytes each. Consecutive sub-fingerprints overlap substantially in the time frame, and consequently, a 8194-bit fingerprint has only 1141-bit entropy. A similar audio fingerprinting system is proposed in [32].

2.2 Common features

The above fuzzy data are all represented as binary strings, containing random bit errors. The value on each bit position represents more a statistical likelihood than the "ground truth" real value. Hence, matching two strings is not based on the exact equality, but on a fuzzy condition: whether their Hamming distance is close enough. The lack of "true values" also implies that one cannot reduce the bit error rates much by taking successive measurements followed by a majority voting (see Chapter 3 for more details).

Furthermore, the obtained fuzzy data are high-dimensional. Iris codes are dispersed in a 2048-D Hamming space, and other fuzzy data have dimensionality of about the same

order (see Table 2.1). However, fuzzy search algorithms proposed in the past generally only work in a low dimensional space, such as 2 to 3-D, and suffer from "curse of dimensionality" when the data dimensionality increases [66]. More details about this search problem can be found in Chapter 4.

In addition, fuzzy data contain redundancies. Due to the strong correlation between bits, a 2048-bit iris code has only 249-bit entropy [5]. In analogy to tossing coins, which is also called the *Bernoulli trial* [5], the one-bit information is obtained by fairly tossing a coin. Repeating this trial will generate a string of bits. The issue of correlation arises when some tosses are not independent, but merely repeat the earlier results. Evidently, an iris code usually has a run length of 8 consecutive '1's or '0's, which roughly explains why it has only 249-bit entropy: $2048/8 = 256$ (see [5]). However, since the correlated bits may be sporadically dispersed, it is actually very difficult to describe the correlations precisely. (Otherwise, one could trivially compress an iris code by 8 times.)

Finally, the acquisition of fuzzy data requires a secure measuring process. Instead of cloning a human finger, an attacker may make a gummy-finger with the engraved patterns to cheat the fingerprint scanner [37, 38]. This kind of attack can be easily defeated under attended operations where the finger is checked physically before being measured. If such checking is not available, the security must depend on the effectiveness of the liveness-detection functions embedded with biometric scanners. Interesting and important as these issues are, liveness detection is outside the discussion scope of this dissertation. We refer readers to [39] for more details.

2.3 A new paper-fingerprinting method

In this section, we will give a quick preview on early work-in-progress: to design a new paper-fingerprinting method that, unlike the reviewed techniques [33, 26] in Table 2.1, requires no customized hardware. This on-going work is not yet fully developed as a presentable contribution (it was not included under Section 1.2 "Contributions"). Here, a brief preview only serves to highlight a possibly low-cost way to capture the uniqueness of this physical world.

Our idea is based on the following observation: when held in front of a table lamp, or towards the sun, a sheet of blank paper reveals complex translucency patterns as shown in Figure 2.1. This phenomenon is caused by the random interleaving of fibers during the paper manufacturing process. To fingerprint a paper document, we propose to analyze the random translucency patterns rather than the non-uniform paper surface as in [33, 26]. To capture the translucency patterns, we will use an ordinary camera to take a close-up, as shown below.

In the photographing process, a sheet of blank paper was put on an overhead projector, with the lamp inside the projector illuminating the back of the paper. The illuminated patterns were then photographed by a 4-mega-pixel camera (Nikon Coolpix 4500 [30]) at a close distance (about 2 cm). Here, the overhead projector was chosen because it was easily available in our laboratory. Alternative lighting sources may include: natural light during bright daytime, flashlights, desk lamps, or office strip lights.

Registration marks are needed to define the imaging area. In addition, they also serve to align position/rotation-shifts between different images. Instead of using dots [33], or cutting edges [26], we chose to print an L-like mark: a short horizontal line plus a slightly

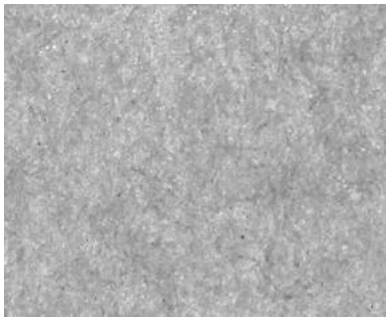


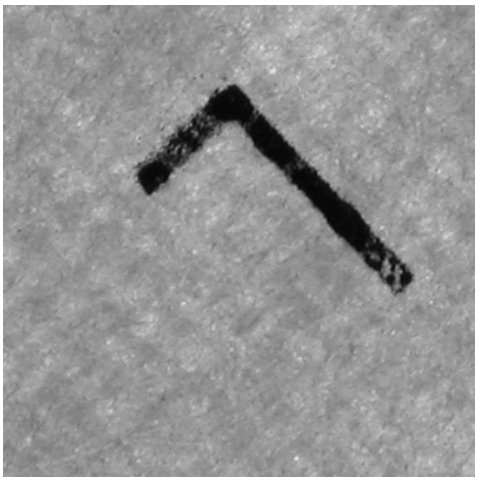
Figure 2.1: Translucency patterns over 1 mm^2 of a sheet of paper

Figure 2.2: The L-mark alongside a 5-pence coin

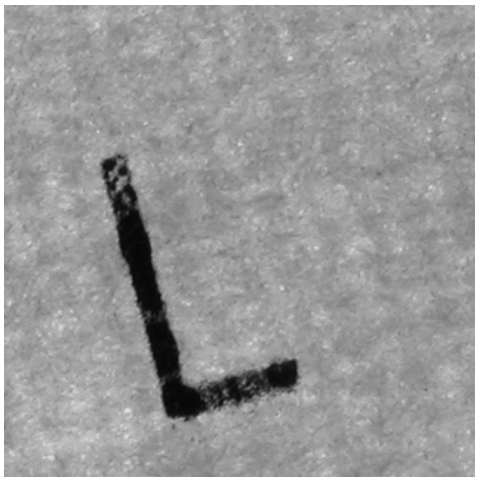
longer vertical line (Figure 2.2). This mark appears tiny on paper, and is sufficiently simple and effective for our purpose. Figure 2.3 displays the "L" marks printed on two sheets of paper.

The L-mark can be detected easily and robustly. Note that the printed "L" looks fuzzy at the microscopic view, which is caused by the imperfection of laser-printing (it may be of independent research interest to capture the unique features of each printed "L" itself). The captured image is processed as follows. First, the L-mark is isolated from the rest of the image (Figure 2.4). Subsequently, points on the inner edges of the two arms (one short, and the other long) are sampled accordingly. On the long arm, a best-fitting line – which has the least Root Mean Square (RMS) distance to the sampled points on the long arm – is drawn. Similarly, another fitting line is plotted on the short arm. Finally, the intersection of the two lines becomes the new origin of the coordinate system, and the image is rotated accordingly by forming the upright "L" pattern (i.e., putting the long arm vertical). Figure 2.5 shows the images after the rotations, with the imaging areas highlighted. With this L-mark, it is also possible to detect whether the paper side is wrongly flipped.

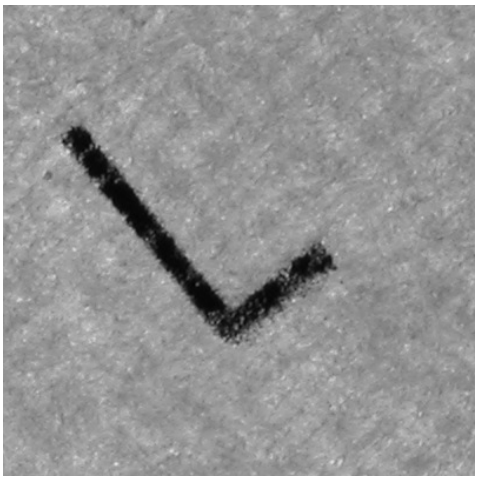
After alignment, photographs of the same paper look very similar, showing reproducible translucency patterns (see Figure 2.5). On the other hand, the displayed patterns appear completely different between different pieces of paper. This makes it possible to encode the random translucency patterns into a compact *paper code*, and there are standard 2D image processing techniques to do that [4, 6]. Note that the captured paper image could contain a much higher information density than an iris image (refer to Figure 2.1). This is because the translucency patterns can be photographed at an arbitrarily



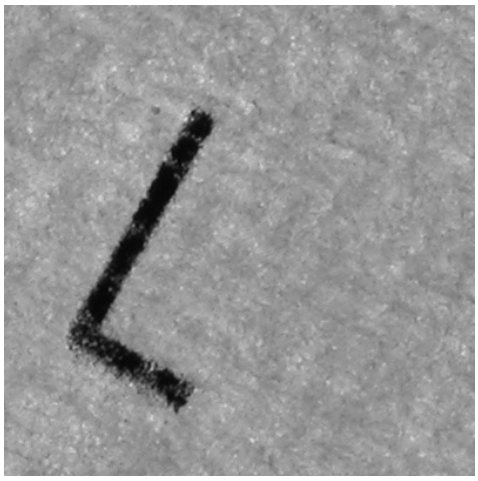
(a) Paper A (sample 1)



(b) Paper A (sample 2)



(c) Paper B (sample 1)



(d) Paper B (sample 2)

Figure 2.3: Photographs of paper A and B before alignment

close distance without being invasive. But in iris recognition, human psychology must be considered; the camera cannot be placed too close to the eyes.

We introduce the above paper-fingerprinting method mainly to highlight the potential of using general-purpose cameras to capture the uniqueness in this physical world. Today, even an ordinary camera has a micro-mode function, which allows a high-resolution close-up of an object. The captured image presents a microscopic view of the object, revealing unique and unclonable physical features. With the continually increasing mega-pixel resolution and dropping price, household cameras will open up a wide range of new security applications (some contemporary developments include biometrics based on analyzing the skin texture [34, 35]).

To sum up, with biometrics enjoying rising popularity, physical one-way functions

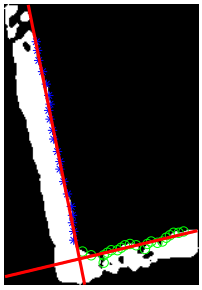
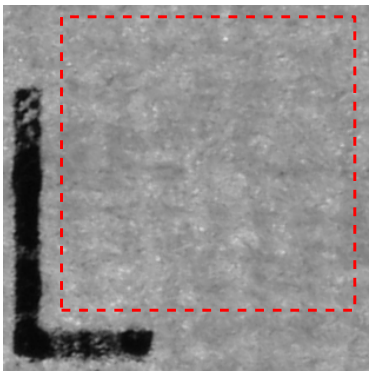
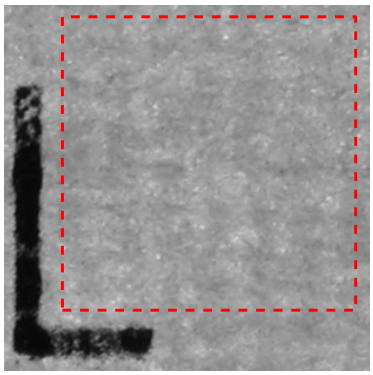


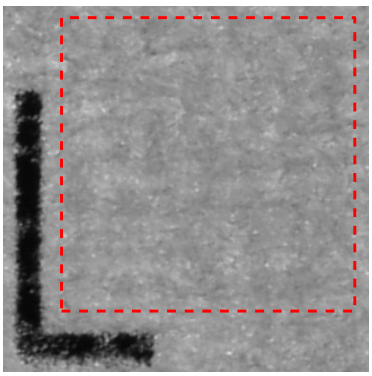
Figure 2.4: Detection of the registration mark



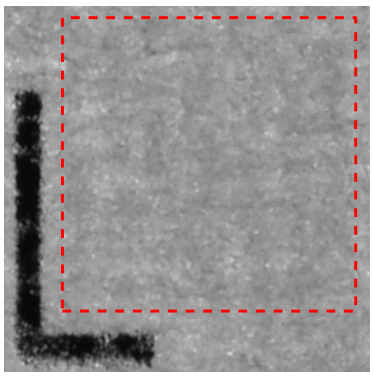
(a) Paper A (sample 1)



(b) Paper A (sample 2)



(c) Paper B (sample 1)



(d) Paper B (sample 2)

Figure 2.5: Photographs of paper A and B after alignment

showing promises to practise cryptography without using number theory, and ordinary cameras opening many possibilities of low-cost measurements, fuzzy data will become more common in future security applications.

In this dissertation, we focus on studying how to effectively use fuzzy data in security applications. For that, we will use the iris biometric as a concrete example. The iris biometric is chosen here, mainly because there is a large amount of ready iris data available in our research project. Details about the iris data sets will be explained in Chapter 3 and 4.

2.4 Conclusion

In this chapter, we reviewed different types of fuzzy data, including iris codes, token fingerprints, paper fingerprints and audio fingerprints. Their common features were summarized, and problems highlighted. In addition, we demonstrated how to fingerprint a piece of paper at low cost by photographing its unique translucency patterns. In the next chapter, we will study how to integrate fuzzy data into cryptographic applications.

Chapter 3

Combining crypto with biometrics effectively

You normally change the problem, if you can't solve it.
— **David Wheeler**

This chapter[1] studies how to integrate fuzzy data, such as iris codes, 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 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.

3.1 Introduction

A number of researchers have studied the interaction between biometrics and cryptography, two potentially complementary security technologies. The science of 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 [1].

[1]The content of this chapter has been published in [12]

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 [40]. 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.

Other researchers have tried to map biometric data into a unique and repeatable binary string [43, 14, 46, 44, 45]. Subsequently, the binary string would be mapped to an encryption key by referring to a look-up table [43, 44, 45], or direct hashing [14, 48]. 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 this dissertation, we will use the term *biometric key*, proposed in [3], 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 many attempts to bridge the gap between the fuzziness of biometrics and the exactitude of cryptography, by deriving biometric keys from key stroke patterns [46], the human voice [44], handwritten signatures [14], fingerprints [43, 41], and facial characteristics [45]. However, so far, these attempts have suffered from an excessive False Rejection Rate (FRR) – usually over 20%, which is unacceptable for practical applications [42].

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 [42]. Biometric features are inherent in individuals, so they cannot be changed easily. A related problem is key diversity: a user may wish separate keys for her bank account and for access to her workplace computer, so that she 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/deployed 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 [42]. 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 codes using error-correction codes [51]. 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.

3.2 Past work

We now provide a more detailed survey of recent research on extracting biometric keys [43, 44, 45, 14, 46, 47]. Monrose, Reiter, Li and Wetzel were among the first: their system [46] 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, Monrose, Reiter and Wetzel proposed a more reliable implementation based on voice biometrics, but with the same discretization methodology [44]. 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% [44].

Hao and Chan made use of handwritten signatures in [14]. 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% [14].

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 Kumar reported a biometric-key system based on fingerprints in [47] and were the first to commercialize this technology into a product – *Bioscrypt* (see `www.bioscrypt.com`). They extract an array of phase values 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 (i.e., the dot product of the extracted phrase array and a random-value array). 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 [43] and used a technique called a *fuzzy vault*, which had been first introduced by Juels and Sudan [50]. 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 [45]. 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 [51, 52]. 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 [57]. 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 et al. [58, 59], 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 [49]. 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 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 stolen and reverse-engineered.

3.3 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 [5]. 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 algorithm 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 [47, 51]. 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 taking the majority bit of 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.

3.3.1 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 that 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.

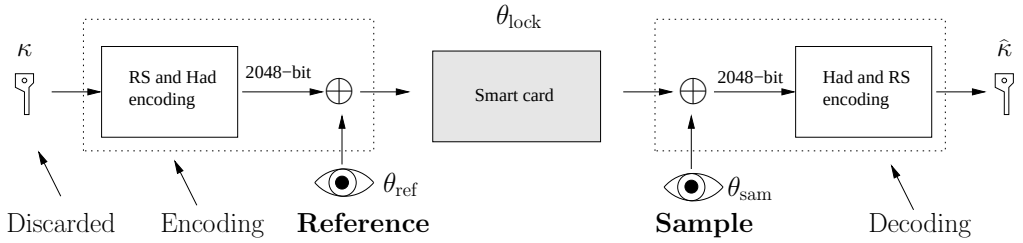


Figure 3.1: A two-factor scheme for biometric key generation

Figure 3.1 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.

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:

$$\theta_{\text{lock}} = \theta_{\text{ps}} \oplus \theta_{\text{ref}}. \tag{3.1}$$

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 written as:

$$(\kappa,\, \theta_{\text{ref}}) \;\mapsto\; \mathcal{T} = (\theta_{\text{lock}},\, H(\kappa))\,. \tag{3.2}$$

During the decoding phase, the user presents her 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 $\widehat{\kappa}$. If the hash of the $\widehat{\kappa}$ matches the stored hash, i.e., $H(\widehat{\kappa}) = H(\kappa)$, the derived key is correct. Otherwise, the key will be deemed false and rejected. The decoding process can be written as

$$(\theta_{\text{sam}},\, \mathcal{T}) \;\mapsto\; \widehat{\kappa}. \tag{3.3}$$

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 [5]. 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 [54], 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.

3.3.2 Hadamard codes

A Hadamard code is generated by an $n \times n$ 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 n of a Hadamard matrix must be 1, 2, or $4m$ for natural numbers m. There are several ways to generate Hadamard matrices; we chose the Sylvester method, which recursively defines normalized matrices whose size is a power of 2, $n = 2^k$ [54].

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

$$H_1 = \begin{bmatrix} + & + \\ + & - \end{bmatrix}, \tag{3.4}$$

where we use "+" to denote "1" and "−" to denote "−1". With the Sylvester method, we obtain further Hadamard matrices recursively by:

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}. \tag{3.5}$$

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

$$H = \begin{pmatrix} H_k \\ -H_k \end{pmatrix}. \tag{3.6}$$

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

The encoding process is to encode an input value i into a codeword w. The matrix of H 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 $\log 2n$ 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

$$v \times H^T = (a_0, a_1, \ldots, a_r, \ldots, a_{2n-1}). \tag{3.7}$$

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., $r = i$.

To sum up, a Hadamard code of size $n = 2^k$ has $2n$ codewords. The code has minimum distance 2^{k-1} and hence corrects up to $2^{k-2} - 1$ bit errors. An input value i is encoded into a codeword w – essentially a row of a Hadamard matrix – which has a size of n bits. So the code maps an input block of $(k+1)$ bits into an output block of $n = 2^k$ bits. More details about Hadamard error correction can be found in [54].

3.3.3 Reed-Solomon code

As explained in Section 3.3.2, 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. To disperse errors, we randomly permute the 2048 iris-code bits before applying the Hadamard code, but some errors are still clustered in certain blocks. Hence we need another layer of error correction to deal with block errors. The Reed-Solomon code, whose details can be found in [55], is a suitable choice. We will now explain how Reed-Solomon coding complements Hadamard coding, and justify our choice of parameters.

3.3.4 Concatenated encoding and decoding

Recall that we use Reed-Solomon coding, then Hadamard coding, to encode a random key κ as shown in Figure 3.1. The Reed-Solomon code is denoted as $\mathcal{RS}(n_\mathrm{s},\, k_\mathrm{s},\, t_\mathrm{s})$, 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 [55], we get $n_\mathrm{s} - k_\mathrm{s} = 2t_\mathrm{s}$.

The size of each block for $\mathcal{RS}(n_\mathrm{s},\, k_\mathrm{s},\, t_\mathrm{s})$ 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 $\|\theta_\mathrm{ps}\| = 2048$. We XOR this with a reference iris code θ_ref to get a locked code θ_lock, which is then saved in the token.

$$\theta_\mathrm{lock} = \theta_\mathrm{ps} \oplus \theta_\mathrm{ref}. \tag{3.8}$$

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 [5]. 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 3.4.3.

The decoding process involves XORing the locked iris code θ_lock with a presented sample θ_sam.

$$\begin{aligned} \theta'_\mathrm{ps} &= \theta_\mathrm{lock} \oplus \theta_\mathrm{sam} \\ &= \theta_\mathrm{ps} \oplus e, \end{aligned} \tag{3.9}$$

where $e = \theta_\mathrm{ref} \oplus \theta_\mathrm{sam}$ is the error vector between two iris codes. The error correction is applied and recovers a trial value of the biometric key $\widehat{\kappa}$. If the error e is within its correction capability, $\widehat{\kappa} = \kappa$, which we can verify by comparing the hash values. Otherwise,

the key will be rejected. We will show in Section 3.4.2, 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:

$$\|\kappa\| = (k+1) \times \left(\frac{2048}{2^k} - 2t_s \right). \tag{3.10}$$

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 $2048/2^k = 32$ blocks of 64 bits, and the Reed-Solomon code outputs 20 blocks. Thus the length of the key κ is 140 bits.

3.4 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.

3.4.1 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 [5]. The matching of two iris codes is decided based on their normalized Hamming distance HD_{norm} [8]:

$$HD_{\text{norm}} = 0.5 - (0.5 - HD_{\text{raw}})\sqrt{\frac{n}{911}}, \tag{3.11}$$

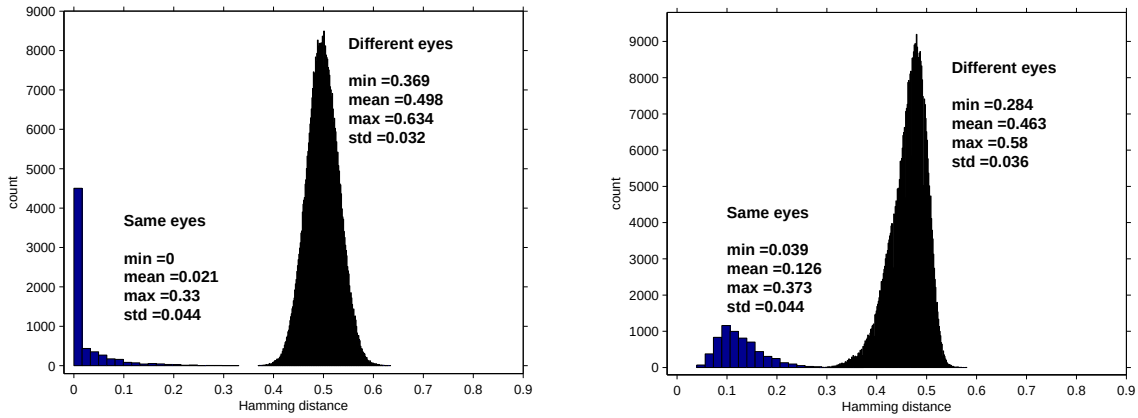
where $n = \|\text{maskA} \cap \text{maskB}\|$, and

$$HD_{\text{raw}} = \frac{\|(\text{codeA} \oplus \text{codeB}) \cap \text{maskA} \cap \text{maskB}\|}{\|\text{maskA} \cap \text{maskB}\|}. \tag{3.12}$$

The mask filters out bits thought to be unreliable because of eyelashes, reflections, obscure boundary detections, etc. The parameter 911 in Equation 3.11 is the typical number of bits mutually available between different iris codes (see [8]). 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 in future research, but for now we use the raw iris codes only. We compute the Hamming distance between two iris codes without masks as:

$$\widehat{HD}_{\text{raw}} = \frac{\|\text{codeA} \oplus \text{codeB}\|}{2048}. \tag{3.13}$$

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



(a) With masks – HD_{norm}

(b) Without masks – $\widehat{HD}_{\text{raw}}$

Figure 3.2: Hamming distance between iris codes

carried out 241,300 comparisons between different eyes and 3,150 comparisons for the same eyes. The results are shown in Figure 3.2. Without masks, the mean intra-eye Hamming distance increases from 2.1% to 12.6%, 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. [5]. In the normal use of the iris recognition algorithm, orientation is readily normalized by cyclically scrolling the iris code by multiples of bytes. 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 bytes and attempted to recover the key each time.

3.4.2 Key length and error rates

The order of the Hadamard matrix sets a trade-off between error tolerance and key length: from Equation 3.10, 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 $k = 6$ is a suitable value by experiment. Table 3.1 shows the performance of error correction for $k = 6$.

As shown in Table 3.1, the $t_{\text{s}} = 6$ 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 3.2 compares our design with the prior art discussed in Section 3.2. Our system achieves vastly better performance. The key length is 140 bits, much longer than the 69 bits obtained from fingerprints in [43]. 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;

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 3.1: Performance when $k = 6$

Biometrics	**Author**	**Features**	**Error handling**	**Key bits**	**FRR**	**FAR**
Keystroke [46]	Monrose (1999)	Durations, latencies	Quantization	12	48.4%	–
Voice [44]	Monrose (2001)	Ceptral coefficients	Quantization	46	20%	–
Signature [14]	Hao (2002)	Forty three dynamics	Feature coding	40	28%	1.2%
Fingerprint [47]	Soutar (2002)	Phase Info	Majority code	–	–	–
Fingerprint [43]	Clancy (2003)	Minutiae points	Reed-Solomon code	69	30%	–
Face [45]	Goh (2003)	Face eigen projections	Quantization, thresholding	–	–	–
Iris	–	**Iris codes**	**Concatenated coding**	**140**	**0.47%**	**0%**

Table 3.2: Summary of biometric key experiments

the poor samples are a fact of life in biometric systems and have to be dealt with by other mechanisms, such as retries.

3.4.3 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 the later 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, and a biometric. Correlations exist in every iris: these are partially 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 [5].

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 [55]: $\mathcal{BF} \geq \frac{2^z}{\sum_{i=0}^{w} \binom{z}{i}} = 2^{44}$.

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 other existing 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.

3.4.4 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. Another possibility is to insist on attended operation. However, these solutions are of limited help if we 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. Another option might be 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 $64! \times 64! \approx 2^{592}$, different matrices through permutation. Permuting the Hadamard matrix also makes the encoded data θ_{ps} appear random (see Equation 3.8), 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 occurred with Trojan attachments to automated teller 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, industrialised 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.

3.4.5 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 [1], 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 how to use biometric means to perform high-quality identification of persons 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.

3.5 Conclusion

In this chapter, 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; it preserves privacy of biometric data as a central database of templates is not needed; and we have shown, in one realistic experiment, that its false-reject rate was under half a percent. With this technique, high-quality identification of persons can be performed using biometric means but without a central database of templates.

Chapter 4

A fast search algorithm for a large fuzzy database

Good research is done with a shovel, not with tweezers.
— **Roger Needham**

This chapter[1] studies the problem of searching a large fuzzy database that stores iris codes or data with a similar structure. The fuzzy nature of iris codes and their high dimensionality render many modern search algorithms – which mainly rely on sorting or hashing – inadequate. The algorithm that is used in all current public deployments of iris recognition is based on a brute force exhaustive search through a database of iris codes, looking for a match that is close enough [7]. Our new technique, Beacon Guided Search (BGS), tackles this problem by dispersing a multitude of "beacons" in the search space. Despite random bit errors, iris codes from the same eye are more likely to collide with the same beacons than those from different eyes. By counting the number of collisions, BGS shrinks the search range dramatically with a negligible loss of precision. We evaluate this technique using 632,500 iris codes enrolled in the United Arab Emirates (UAE) border control system, showing a substantial improvement in search speed with a negligible loss of accuracy. In addition, we demonstrate that the empirical results match theoretical predictions.

4.1 Introduction

In the previous chapter, we introduced a technique that performs biometric recognition without requiring a central database of biometric templates. One condition of that technique, however, is that it requires cooperative users who must supply tokens and passwords (if any); otherwise recognition will fail. In some cases, it is desirable to have an automatic recognition process with little assistance from the user – for example, by using iris recognition.

Iris recognition is a relatively new biometric technology. It takes a high-resolution infrared image of a person's eye, isolates the iris, demodulates the pattern of iris texture into a binary iris code, and compares it exhaustively against an enrolled database for a match [5]. Because of its high accuracy, this technology has been deployed at many

[1]The content of this chapter has been accepted for publication in [9]

airports as a replacement for passports, and in particular it is deployed at all 27 air, land, and sea-ports of entry into the United Arab Emirates (UAE) as a border control security system to prevent expellees from re-entering the country [7].

To deploy a large-scale biometric recognition system, the first concern is the likelihood of false matches, which increases with the number of records enrolled in the database. Iris patterns contain a high degree of randomness, which provides the biological basis for their uniqueness. Daugman's algorithm [5] is the standard technique for encoding that randomness into a 256-byte iris code; it also produces a 256-byte mask, which excludes those iris-code bits affected by eyelids, eyelashes, specular reflections, and other noise. Statistical analysis reveals that with careful selection of matching thresholds, the cumulative false match rate for iris recognition remains negligible even over large populations [8, 5]. To date, there have been about one million iris codes enrolled in the UAE central database, and the UAE Ministry of Interior reports that the system has yet to make a false match [8].

The success of the UAE application since 2001 has encouraged even larger deployments. One such will be seen in the UK, where the Government is to introduce biometrically-enabled ID cards in 2009 [68]. Under this scheme, biometric data including the iris codes of the 45 million citizens older than 16 will be stored in a central database. A similar program exists in India. The Andhra Pradesh State government has been enrolling iris codes for 80 million local people since July 2005 under a ration-card scheme and, within the first year, about 26 million people had been enrolled [69]. With the advent of large biometric databases, information retrieval and database management will become increasingly challenging problems.

Comparing two iris codes is simple; it mainly involves counting the bits that differ between two binary vectors. Iris images may be tilted to various degrees; this problem is handled by repeating the comparisons of the iris codes over a range of relative rotations [5]. Since the comparison requires no expensive non-linear warping operations as in [15, 70], searching iris-code databases can be quite fast. The exhaustive search method can compare about a million iris codes per second on a "3.2 GHz CPU" [7]. However, continual database expansion will slow down the search speed linearly, and the default solution is to use several search engines in parallel [7].

Far more severe problems arise in applications requiring that all records in a national-sized database be compared with all others in order to detect fraudulent multiple identities, which is one of the purposes of the UK ID card scheme ("One person, one identity" [68]). The number of cross-comparisons then scales with the square of the national population. Although this process need only be done over the time-course of ID card issuance, its demands are still daunting. The 45 million UK enrollees generate about 2×10^{15} iris pair comparisons. At 1 million iris code comparisons per second per 3.2 GHz CPU, this would require 2 billion CPU-seconds, which is 63 CPU-years.

In the UK biometric ID plan with a database of 90 million iris codes, the memory management and maintenance requirements for an exhaustive search approach are also daunting. A single exhaustive search would require loading $90 \times 10^6 \times 512$ bytes = 46 GB into memory, and every update would require manipulating a file of this size. There is thus a strong motivation to try to develop some kind of indexing based approach instead, in which each iris code (despite its fuzziness) could be treated almost as an address that points directly to the identity of the person who generated it.

This search problem is more general. There are a wide range of closely related applications, for instance, searching music fingerprints [31, 32], paper fingerprints [26], and optical fingerprints [28]. All these applications produce iris-code-like fuzzy data: high-dimensional binary vectors with the comparison based on the Hamming distance. In practice, there could be billions of music fingerprints or paper fingerprints in a database [26, 31, 32]. Using exhaustive search would require thousands of parallel machines, which is clearly infeasible under cost constraints.

We therefore set out to solve this problem at the algorithmic level, without relying on customized hardware or parallelism. Though our work focuses on searching iris codes, the devised technique is generally applicable to other fuzzy search domains.

4.2 Past work

The problem we investigate is: given a 2048-bit vector with random errors, how to quickly find its nearest neighbor in the 2048-D Hamming space. A more general problem, in any metric space, is called Nearest Neighbor Search [60].

Nearest Neighbor Search (NNS) is defined as follows: given a set P of points in a high-dimensional space, construct a data structure which, given any query point q, finds the point p closest to q under a defined distance metric [60]. The NNS problem has been extensively studied for the past two decades. The results, however, are far from satisfactory, especially in high dimensional spaces [64, 65, 66]. We will now review the past work in this line of research.

Most NNS techniques are based on the "partitioning principle" [60]. The idea is intuitive: dividing the search space into regions, so that once a query is given, only some regions are searched. Generally, the first step is to choose pivots – the reference points that divide the space. For instance, in the "ball partitioning" method, the ball center is a pivot. The ball cuts the search space into halves: inside the ball and outside. Such spheric cuts are performed recursively, leading to a balanced binary tree with pivots placed at nodes and data at leaves. An alternative method is the "generalized hyperplane partitioning", which separates the data set into two based on their relative distances to two pivot points.

The "partitioning principle" spawns many tree-like data structures. Specific techniques differ in how trees are constructed and traversed, as well as trade-offs. For example, m-tree, vp-tree, fq-tree, mvp-tree, mwvp-tree are based on the "ball partitioning", while Bisector tree, gh-tree, gna-tree, kd-tree, pcp-tree are derived from the "generalized hyperplane partitioning". Selecting the right pivots is important, and not trivial. Due to the complexity of the problem, most techniques choose pivots at random [61]. Some techniques, like gna-tree and mvp-tree, employ pre-computation to reduce the distance computations, but require a large amount of memory. The state-of-the-art of the partitioning-based approach is summarized in [60].

Unfortunately, all of the above tree-like data structures succumb to the "curse of dimensionality" [64, 65, 66]. While they work reasonably well in a 2 or 3 dimensional space, as the dimensionality of data increases, the query time and data storage exhibits an exponential increase, thereby doing no better than the brute-force linear search [66]. The reason is that the intuition of dividing a space into regions no longer holds in the high dimensional case. In the 2048-D Hamming space, for instance, the distances between different-eye iris codes are sharply centered at 0.5 (see Section 4.3.3); an iris code is very

nearly equidistant from all the others. Hence, it is very difficult, if not impossible, to divide this space into regions. Furthermore, iris codes form no clusters in this space, so various clustering-based NNS techniques [60] do not apply here either.

Indyk and Motwani proposed a different approach for NNS: Locality Sensitive Hashing (LSH) [64], with the follow-on work in [65, 66, 67]. The core part in LSH is the definition of a hash function family $\mathcal{G}$, from which l functions, $g_1, g_2, \ldots, g_l$, are chosen uniformly at random. During preprocessing, every point $p \in P$ is stored in each of the l buckets, identified by $g_i(p)$. To process a query q, LSH searches all buckets $g_1(q), g_2(q), \ldots, g_l(q)$ exhaustively for the nearest match. With some probability, the nearest match exists in one of the buckets. To avoid repeated checking on the same points, the searched points are marked in memory and, thus, ignored for the following occurrences in other buckets [66, 67].

The biggest limitation facing LSH is that it often needs to search a significant percentage of the database [65]. The suggested measure is to define a threshold, and interrupt the search when there are too many points in buckets [64, 65, 66]. We will explain this problem in further detail in Section 4.4. In addition, Indyk and Motwani developed the LSH algorithm based on main memory [64], and thus ignored the delay of data retrieval. However, most large-scale applications store data in disk-based databases.

Haitsma and Kalker proposed a similar technique and applied it to search music fingerprints [32]. They divide a 1024-byte music fingerprint into 256 sub-fingerprints of 32 bits each. They show that, for a "mildly degraded" music signal, at least one of the sub-fingerprints is error-free. Thus, the error-free one can serve as a pointer to look for possible matches, by following a 32-bit look-up table. However, it is acknowledged in the paper that "mild degradation" may be too strong an assumption in practice. Furthermore, the 32-bit look-up table requires at least 4 GB memory, which is more than many computers can offer.

To sum up, LSH and Haitsma-Kalker's algorithm are built on essentially the same principle, which we call the "single collision principle". This principle assumes that, if two records are similar, they have a relatively high chance of having at least one "colliding" identical segment, obtained from either hashing [64, 65, 66] or direct sampling [32]. However, both algorithms ignore multiple occurrences of a collision. We improve their work by introducing the "multiple colliding segments principle", and demonstrate that counting the number of collisions is crucial for achieving optimal performance. Applying this new principle, we are able to resolve the limitations explained above (also see [67, 66, 65, 32]).

4.3 Algorithms

4.3.1 Experiment setup

The UAE database contains $N = 632500$ non-duplicate iris records. Each record includes an iris code and a mask [5]. It is assigned a unique 32-bit ID, and stored as a 512-byte binary *blob* in a MySQL database [71]. The evaluation is performed on a 3-GHz Intel PC with 3-GB RAM, running FreeBSD 5.5. A Java client program retrieves the data in real time, using the Connector/J JDBC driver.

Input: Query iris code q
Output: Match iris code p or 'No match'

1: **for** ID $i = 1$ to N **do**
2: $p \leftarrow$ IrisTable $[i]$
3: **for** $k = -3, -2, \ldots, 3$ **do**
4: Obtain q_k by rotating q by k bytes (pre-computed)
5: **if** PreliminaryCheck(p, q_k) is OK **then**
6: **if** $HD_{\text{norm}}(p, q_k) <$ MatchThreshold **then**
7: **return** p
8: **return** 'No match'

Algorithm 1: Exhaustive Search

4.3.2 Exhaustive search

With several optimizations, Exhaustive Search (ES) is implemented as in Algorithm 1. First, the system retrieves the stored records into memory. We find that the fastest method is loading all data (about 320 MB) at once with one SQL query, instead of fetching one by one, which requires 632,500 SQL queries. Next, the query iris code is compared exhaustively with all retrieved records. The comparison between two iris samples is based on their smallest Hamming distance obtained from seven relative rotations. Computing the Hamming distance mainly involves counting the bits that differ between two binary vectors. A look-up table is used to speed up the counting. Furthermore, the system performs a preliminary check before fully comparing two vectors. It selects every fourth byte from two iris codes correspondingly, and counts the bits that differ. Only if less than a third of these bits disagree will ES proceed to a full comparison. The matching decision is based on the normalized Hamming distance between two iris samples (see Equation 3.11 on Page 27).

In the experiment, the matching threshold is set at 0.22, which is suggested in [8] for the UK population. Figure 4.1 shows the distribution of the normalized Hamming distances for the UAE database based on $\frac{N \times (N-1)}{2} = 200$ billion cross comparisons.

To find an existent match, ES searches on average half of the database. With Java code, the average delay of computing these Hamming distances for each given query is 2.675 s, which is slower than the C-based program that can compare one million iris codes per second [7]. This is because Java executes slower than C (which is a trade-off for Java's portability) [71]. However, this difference is irrelevant since we aim to compare the two algorithms based on the same platform.

In addition, loading data from the database incurs a significant delay too. In the experiment, it takes 6.508 seconds to load 632,500 iris records into memory. Therefore, if the search returns no match, the total delay is: $6.508 + 2.675 \times 2 = 11.858$ s. At first glance, it appears trivial to reduce the delay by loading all data once and holding it in memory. However, this would create problems for memory management: it is complex and expensive to maintain consistency between the data in memory and the constantly updated records in database. Hence, for an efficient search algorithm, *both* the search and loading times need to be substantially reduced.

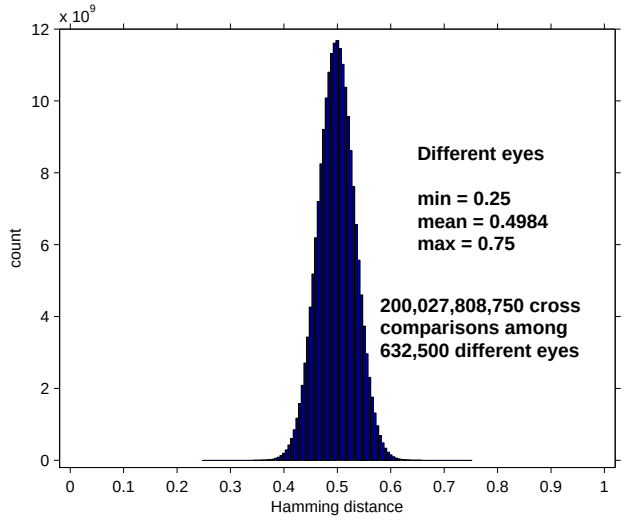


Figure 4.1: Histogram of the UAE database (632,500 records)

Input: A table of N iris codes: IrisTable $[1 \ldots N]$
Output: BeaconGuidingStructure: BeaSpace_1, BeaSpace_2, ..., BeaSpace_n ($n = 128$)

1: **for** ID $i = 1$ to N **do**
2: $p \leftarrow$ IrisTable $[i]$
3: **for** $j = 1$ to n **do**
4: Compute the beacon index b_j from p
5: Insert i into BeaSpace_j $[b_j]$

Algorithm 2: Beacon Guided Search – Preprocessing

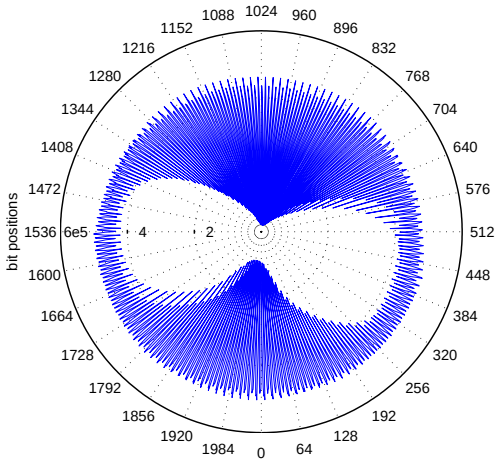
4.3.3 Beacon Guided Search

We now describe a new search algorithm called Beacon Guided Search (BGS). Here, a beacon is a collection of *iris-code IDs* that have the same defined feature. It differs from a bucket [64] in that it is more than a storage unit, but more importantly, a guiding landmark. Technically, it works as follows.

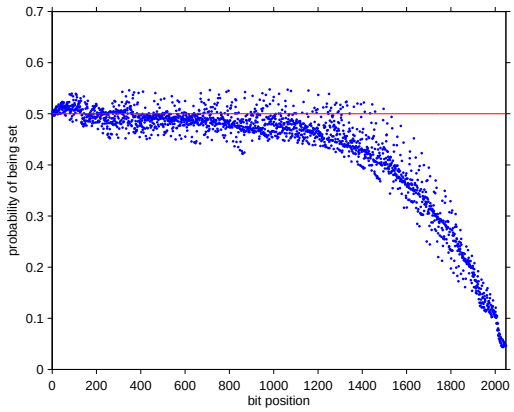
Preprocessing

During preprocessing, all iris codes in the database are indexed to create a Beacon Guiding Structure (Algorithm 2). This process creates $n = 128$ *beacon spaces*, with 2^m ($m = 10$) beacons in each space. An m-bit *beacon index* b_i uniquely identifies every beacon in the ith space. Indexing an iris code involves saving its unique 32-bit ID on one beacon per beacon space. The extra storage required is $128 \times 4 \times N = 512 \times N$ bytes, the same size as the original database.

To compute b_i ($1 \leq i \leq 128$), we first need to study the iris code more closely. The 2048 bits in the iris code vary in quality, with the unreliable ones filtered out by the mask. The set bits in the mask indicate those positions, on which the iris-code bits are effectively included in the comparison. We count the set bits of 632,500 masks across the 2048 bit



(a) count of mask set bits



(b) probability of iris-code set bits (sorted)

Figure 4.2: 632,500 masks and iris codes across 2048 bit positions

positions and plot the result in Figure 4.2 (a). Figure 4.2 (b) plots the probability of having a set iris-code bit across the positions, which are sorted based on the descending counts of the mask set bits. It shows that unmasked iris-code bits generally have an equal probability of being '0' or '1'.

The "butterfly" pattern in Figure 4.2 (a) is caused by several factors. The outer circumference of the graph comprises the Least Significant Bits (LSBs) of the 256 bytes that represent the area near the pupil, where few bits are ever masked. The inner circumference, on the other hand, represents the area far away from the pupil, where the eyelids have a significant effect. The iris-code bits are derived from the iris image, starting from the 6 o'clock position and sweeping the iris area counter-clockwise in one round, as shown in Figure 4.4 (the same for mask bits). The 3 and 9 o'clock regions of the iris are usually unobstructed by either eyelid, which leads to relatively high counts near the 384 and 1536 bit positions, with the "fine-teeth" oscillations due to the varying wavelet sizes used in image processing. The outer regions near the 12 and 6 o'clock of the iris are often occluded by the upper and lower eyelids respectively. The effect of gravity makes both

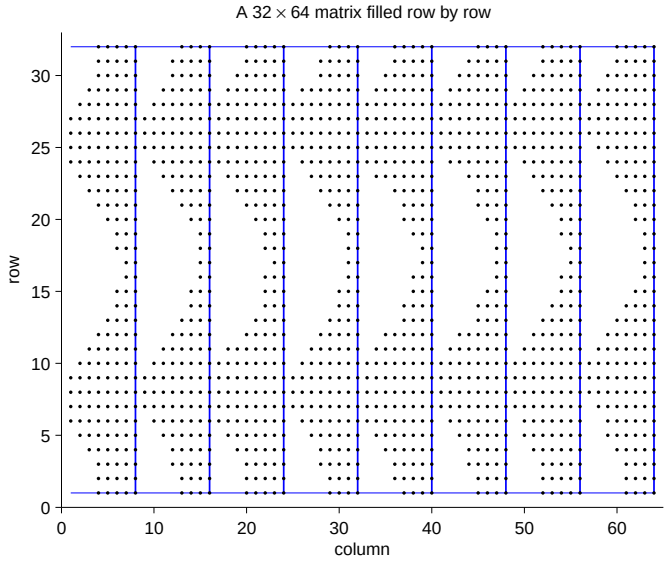


Figure 4.3: bit positions with the 1408 highest counts of mask set bits

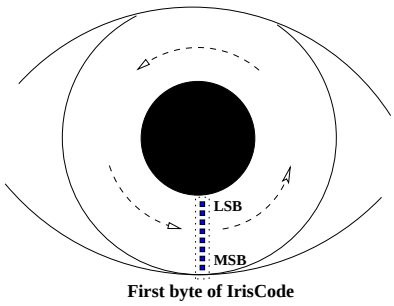


Figure 4.4: Generation of iris code from eye

eyelids sag down; the upper eyelid is more likely than the lower one to obstruct the iris area, causing the upper/lower asymmetry on the graph.

In Figure 4.3, we highlight 1408 (256×5.5) bit positions with the highest counts of mask set bits, filling a 32×64 matrix row by row. This figure shows that the unmasked iris-code bits tend to occupy the least significant bits in a byte.

Based on the above observations, we compute the beacon index b_i from an iris code in two steps. First, we permute the iris code by interleaving the bytes, and then rotating the bit columns (see Figure 4.5). This well separates the neighborhood bits which are strongly correlated [5]. Second, we divide the permuted vector into 128 blocks of 2 bytes each. In each block, b_i is obtained by concatenating the five least significant bits of both bytes; the selected bits correspond to the iris region near the pupil, where the obstruction by eyelashes/eyelids is unlikely, and the signal to noise ratio is generally high.

The beacons derived above are cyclic. Intuitively, the 8 bit-columns in Figure 4.5 correspond to 8 concentric rings overlaying the iris region; rotating each ring by a different angle separates the bits in a byte, meanwhile still keeping the rings cyclic. Since we only extract the last 5 bits of a byte, it is sufficient to rotate just the last 4 columns; in our

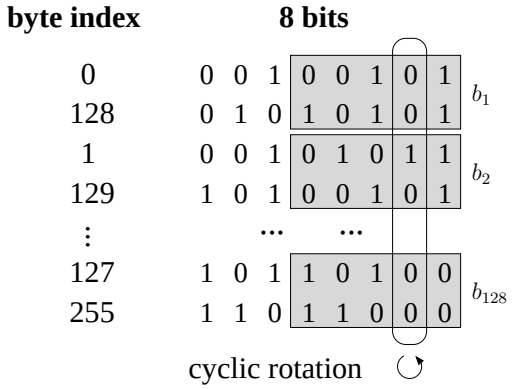


Figure 4.5: Permutation by rotating bit columns

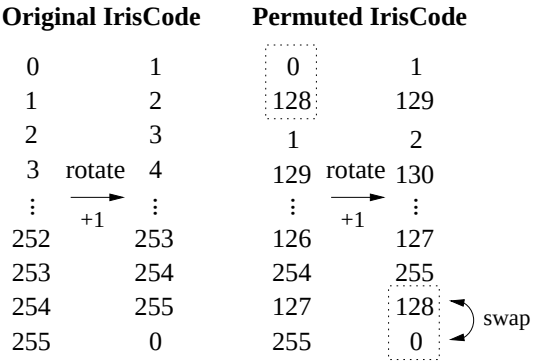


Figure 4.6: Cyclic iris code after permutation

implementation, we rotated the 4 columns downwardly by 25, 50, 75, 100 rows respectively so that the two adjacent bits become at least 25 bytes apart. The reason for interleaving the bytes is to keep the beacon's and the iris code's rotations synchronous – rotating the iris code by one byte corresponds to shifting b_i to the next (or previous) beacon space (see Figure 4.6). When b_i is shifted out of the first space and thus into the last space (and vice versa), the first and second halves of the bits in b_i need to be swapped to ensure the correct beacon value. Such a calibration is done in memory with a negligible delay.

After indexing all iris codes, BGS generates a Beacon Guiding Structure, with $128 \times 2^m = 0.13$ million beacons in total. Because of the permutation, the distribution of the beacon storage density is approximately uniform, with an average of $N/2^m = 617$ IDs stored on each beacon for the UAE database. Preprocessing is fast; it took less than 5 minutes to index the entire UAE database.

Searching

Searching an iris code involves traversing the beacon spaces (Algorithm 3). In each space, BGS selects one beacon and retrieves the IDs stored in that beacon. The array, counter_k, records the accumulated occurrences of the retrieved IDs, with the subscript k referring to a particular rotation. If a specific ID has been encountered c times (e.g., $c = 3$), BGS loads the full 512-byte iris data using the ID as the primary key. The subsequent

Input: Query iris code q
Output: Match iris code p or 'No match'
Preprocessed: BeaSpace_1, BeaSpace_2, ..., BeaSpace_n ($n = 128$)

```
1: Compute the beacon indices b_1, b_2, ..., b_n from q.
2: for i = 1 to n do
3:   for k = −3, −2, ..., 3 do
4:     IDs ← BeaSpace_i [b_{i+k}]
5:     for j ∈ IDs do
6:       Increment counter_k [j] by 1;
7:       if counter_k [j] = c then
8:         p ← IrisTable [j]
9:         if HD_norm(p, q_k) < MatchThreshold then
10:          Return p
11: Return 'No match'
```

Algorithm 3: Beacon Guided Search – Searching

No	Components	Compl	Operation	Cost
1	Compute beacons	$O(1)$	memory	< 1 ms
2	Retrieve beacon IDs	$O(n)$	disk	lightweight
3	Count collisions	$O(n)$	memory	very fast
4	Load iris codes	$O(n)$	disk	heavyweight
5	Calculate HD	$O(n)$	memory	fast

Table 4.1: Cost components in Beacon Guided Search

comparison is based on the same matching condition as in ES (see Algorithm 1). Note that in Algorithm 3, the computation of the space index after shifting is based on the modular operation, since the beacon spaces are cyclic.

There are five cost components in BGS, as summarized in Table 4.1. The first one is to compute which beacons a given query belongs to. This operation incurs a negligible delay (< 1 ms). The second one is to retrieve IDs stored in the beacons. This I/O operation is fast, since the IDs are lightweight, and can be fetched rapidly in the form of *blobs* (see [71]). The third operation is to count collisions. This simply involves read/write accesses to the memory array. The next is to load the full 512-byte iris data. This is an expensive I/O operation. The final one is to fully compare the retrieved iris codes with the query. Delays in operations 2–5 increase linearly with more records enrolled into the database, giving the algorithm a linear complexity overall. Our strategy is to make the best use of the very fast operations 2 and 3, while avoiding 4 and 5. In the following section, we will explain how this goal is achieved.

4.4 Results

In this section, we evaluate the BGS performance both analytically and empirically.

4.4.1 Theory

The Binomial Probability Distribution function [5] is:

$$f(x,\, n,\, p) = \frac{n!}{x!(n-x)!}\, p^x (1-p)^{n-x},$$

where x is the number of successful trials, n is the total number of independent trials, and p is the success probability. The Binomial Cumulative Distribution function is defined as:

$$F(x,\, n,\, p) = \sum_{i=0}^{x} f(i,\, n,\, p).$$

For simplicity, we assume the beacon bits are equiprobable and uncorrelated, and will address the effect of correlation later. The probability of finding a match, thus terminating the search, in the ith space depends on two conditions: 1) there had been $c-1$ collisions with the matching iris code before the ith space; and 2) there is one more collision in the ith space. It is:

$$P_{\text{term}} = f\left(c-1,\, i-1,\, (1-p_{\text{b}})^m\right) \times (1-p_{\text{b}})^m, \tag{4.1}$$

where p_{b} is the bit error rate of the same-eye query after the 7-rotation adjustment (i.e., the smallest result among 7 rotations); it typically ranges from 10 to 20%.

The miss rate is the probability that the query iris code (after the 7-rotation adjustment) and the supposed match collide with less than c beacons. It is:

$$P_{\text{miss}} = F(c-1,\, 128,\, (1-p_{\text{b}})^m). \tag{4.2}$$

In the ith space, the probability for two different iris codes to have less than c beacon collisions for all 7 rotations is: $F^7(c-1,\, i,\, 0.5^m)$. Here, we define the *search size* as the number of iris records retrieved for comparison. When the search is terminated in the ith space, the search size can be estimated by:

$$S_i = N \times (1 - F^7(c-1,\, i,\, 0.5^m)). \tag{4.3}$$

The value S_{128} represents the search size when a match is not found.

4.4.2 Experiment

The search size in BGS is only a fraction of the whole database. This is mainly due to the "multiple colliding segments principle" and the early termination strategy, whose effects are described by S_{128}/N and S_i/S_{128} $(i \leq 128)$ respectively.

First, we study the performance when a search returns no match, hence has no early termination. We conduct the experiment using two types of queries: a string of random bits and an iris code with no match in the database. Figure 4.7 summarizes the search delays.

When $c = 1$, the implementation would be similar to [64, 65, 66, 32], which are based on the "single collision principle". However, a significant percentage of the database needs to be searched. Based on random queries, a theoretical estimate of the search size is $S_{128} = 368947$, and the experiment shows 384648. This problem might be less evident

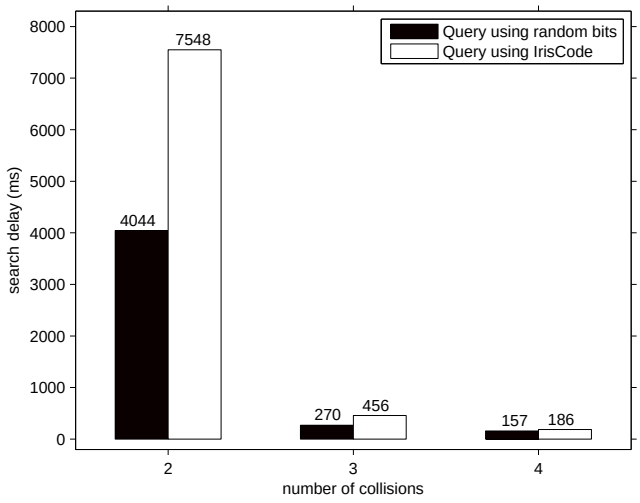


Figure 4.7: Delays in BGS when no matches are found

	c = 2		c = 3		c = 4	
Query type	theory	experi	theory	experi	theory	experi
Random	30959	30819	1283	1280	38	38
Iris code	–	43966	–	3333	–	227

Table 4.2: Search sizes in BGS when no matches are found

if all data are held in memory, but could prove severe for disk-based applications. It is particularly slow to retrieve sporadically dispersed data from disk; we observe that, even for fetching 10% of the records, the resultant delay would be longer than that using ES.

Table 4.2 reports the search sizes using BGS, together with the theoretical estimates: S_{128} (see Equation 4.3). The different results for the two query types are due to data correlation, for which a 2048-bit iris code has only 249 degrees of freedom [5]. Applying permutation helps select uncorrelated bits into b_i, but cannot remove the correlation between b_i ($1 \leq i \leq 128$). As a result, iris codes tend to cling to the same beacons in different spaces. For a query of random bits, the beacon selection is random too, which cancels the effect of correlation. On the other hand, if the query is an iris code, the beacons derived are correlated; thus, more IDs fulfill the c-collision requirement, bulging the search size.

However, the bulge in the search size has a limited effect on search performance. When $c = 3$, it causes the search size to increase from 1280 to 3333, a factor of 2.6. Even after bulging, the size occupies a small fraction of the database. As a result, the delay increases from 270 to 456 ms, a factor of only 1.6.

On the other hand, correlation has almost no effect on the search accuracy. To study the variation of intra-eye iris codes, we collected multiple iris samples from each of seventy eyes (more details about the data collection can be found in Section 4.4.3). From each eye, one sample is added into the database, and the rest are used as queries. Figure 4.8 plots the probability of finding matches versus the bit error rates of the queries, as well

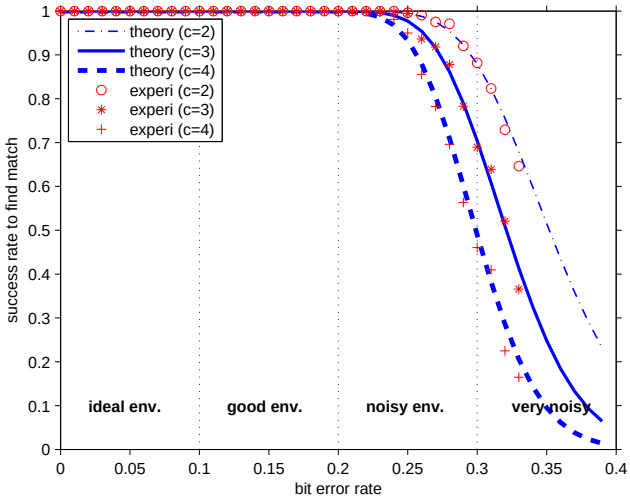


Figure 4.8: Success rates of finding matches for different c values ($m = 10$)

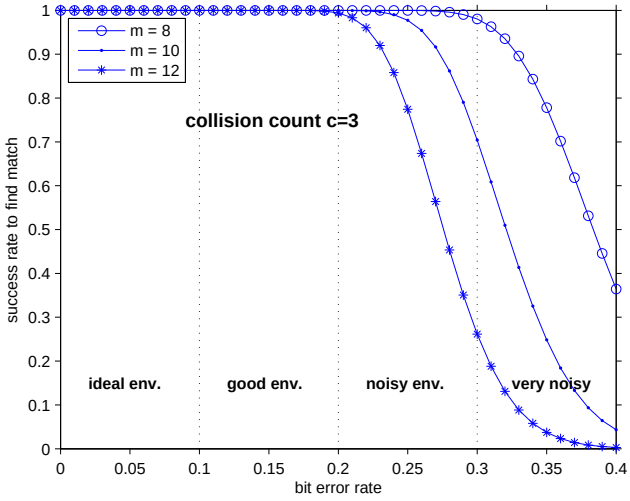


Figure 4.9: Error tolerance for different m values

as the theoretical results: $1 - P_{\text{miss}}$ (see Equation 4.2). Theoretical and empirical data are consistent. Overall, the value $c = 3$ strikes a suitable trade-off between speed and accuracy, as we will explain in further detail in Section 4.4.3.

The number of beacons in one beacon space, 2^m, presents another trade-off between speed and accuracy. From Equation 4.2, a smaller m leads to better error tolerance, as shown in Figure 4.9. On the other hand, a faster speed is achieved by choosing a bigger m, since fewer iris records need to be retrieved. Without considering early termination, the fractions S_{128}/N for $m = 8, 10, 12$ are 9.5%, 0.2% and 0.0034% respectively (see Equation 4.3). Defining a suitable m value may well depend on particular application requirements. In the experiment, we chose $m = 10$.

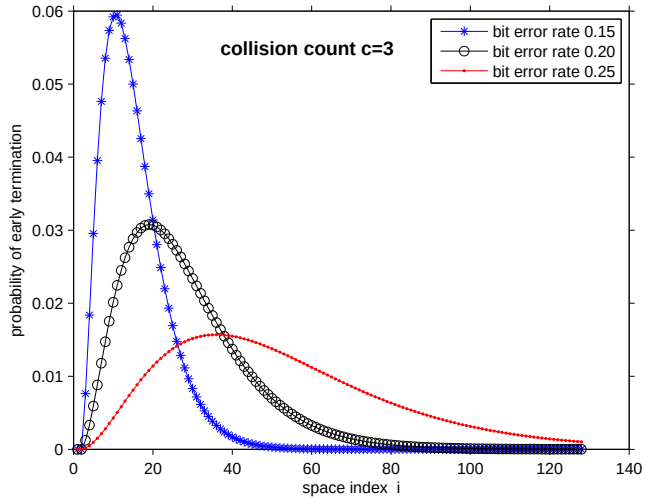


Figure 4.10: Probability of early termination

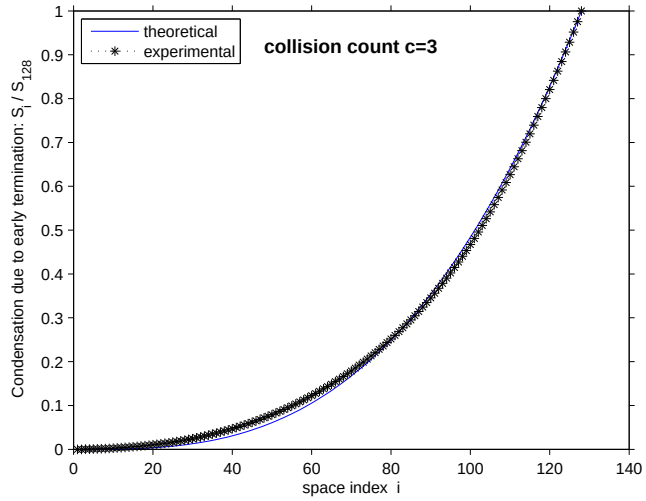
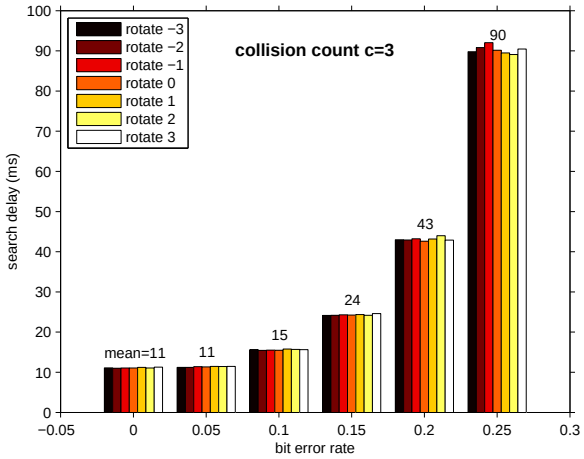


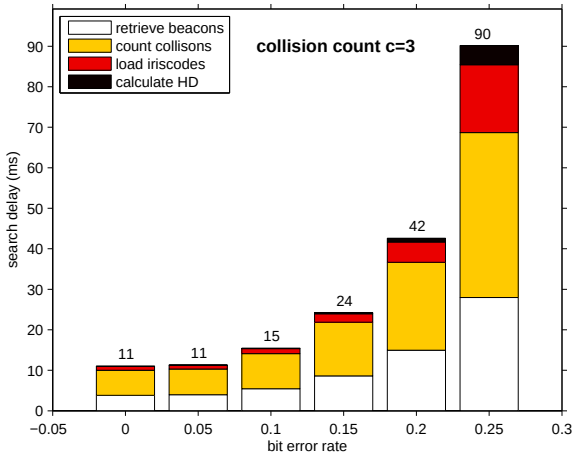
Figure 4.11: Condensation due to early termination

Besides the "multiple colliding segments principle", the early termination strategy also contributes to the speed-up. This strategy was commonly used in past work [60]. The fewer errors in a query, the more likely the match is found at an early stage. Figure 4.10 plots the probability of having early termination for varying bit error rates, based on Equation 4.1. For a query with 15% bit errors, it is statistically guaranteed that the match is found by traversing no more than 50 beacon spaces. Figure 4.11 plots the theoretical value of S_i/S_{128} $(1 \leq i \leq 128)$ as well as the experimental results. The two curves fit closely, which shows that the condensation due to early termination is not affected by the data correlation.

By design, our algorithm caters for cyclic rotations of a query. The common approach in past work is trial-and-error: shifting the query a few times, then searching the shifted



(a) Delays for 7 rotations



(b) Breakdown of delays for rotation=0

Figure 4.12: Delays in BGS with seven cyclic rotations

queries [31]. The problem, though, is that a slight shift of the query would remove most benefits of early termination, since failed trials will incur relatively long delays. We tackle this problem by incorporating the rotations into the algorithmic design, so that the search performance is rotation-invariant, as shown in Figure 4.12 (a). Figure 4.12 (b) shows a cost breakdown which indicates that the delay is dominated by the cost components 2 and 3 (see Table 4.1), with 4 and 5 diminishing to be insignificant.

4.4.3 Comparison

To make the evaluation more realistic, we enroll more iris samples under two different conditions: favorable and noisy. The histograms of the two additional datasets are shown in Figure 4.13.

In the first experiment, 10 samples were collected from 70 eyes each, using the same camera and at a fixed measurement distance. We index the first sample from each eye,

Algorithm	FAR	FRR	Search size	Avg delay (ms)	max (ms)
ES	0	0.32%	317262	9,192	11,860
BGS (c=2)	0	0.48%	1087	133	6,488
BGS (c=3)	**0**	**0.64%**	**41**	**30**	**318**
BGS (c=4)	0	0.96%	2	33	177

Table 4.3: Comparison between ES and BGS

and use the remaining nine as queries. Since the enrollment setting is consistent, the mean intra-eye Hamming distance is small: only 2% (see Figure 4.13 (a)). Currently, the BGS algorithm only indexes the iris codes, which have a higher bit error rate without including masks. We find the bit error rate of the beacons derived from the same eyes ranging from 4.56% up to 30.99%, with the mean of 14.94%. This error range can be well tolerated by BGS, as shown in the following.

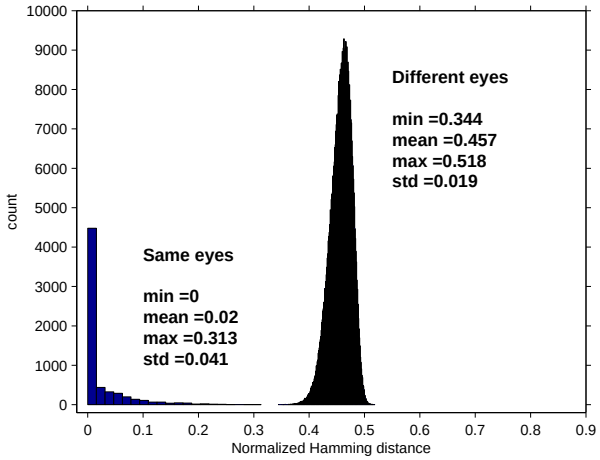
Table 4.3 summarizes the experiment results for both ES and BGS. When $c = 3$, BGS reports an average delay of 30 ms, which is over 300 times faster than ES. This is achieved as BGS checks only $41/N = 0.006\%$ of the database, while ES has to compare with half of the records on average. The degradation of the false rejection rate – from 0.32% to 0.64% – is negligible. When $c = 4$, only 2 records are checked with a 99.04% success rate in finding the match. However, the gain in condensation is offset by the delayed early termination. As a result, the average search delay is 33 ms, slightly longer than that of $c = 3$. Note that in this evaluation, we took the data loading time into account. If we wish to hold all data in memory, we could achieve far more impressive performance by modifying BGS accordingly, but that will be less useful in practice.

In fact, BGS can be made much faster if we remove the 7-rotation requirement. In that case, the experiment shows that it takes merely 4 ms to find a match, provided that the query has less than 30% bit errors and no rotations. However, a slight rotation of the query would cause the search to return no match and incur an upper-bound delay of 28 ms. This may not be an issue if rotations are rare. However, in a noisy environment, as demonstrated below, rotational shifts are fairly common.

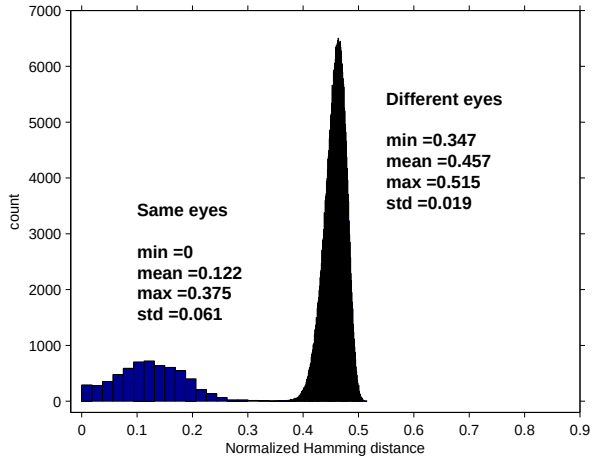
In reality, some users may wish not to be recognized, as opposed to the cooperative users discussed in Chapter 3. Since iris recognition requires little assistance from users, the non-cooperation usually cannot change the recognition outcome. Nonetheless, for those non-cooperative users, the enrolled samples are likely to contain more errors.

Hence, in the second experiment, we simulate a noisy environment. The experiment setup includes two platforms using different cameras and measurement distances. On either platform, 10 samples were collected from 61 eyes each. We index samples obtained on one platform, and use those acquired on the other platform as queries. The iris samples become fuzzier now: the mean intra-eye Hamming distance increases to 12.2% (see Figure 4.13 (b)).

Under this noisy condition, ES reports a 1.32% false rejection rate. The bit error rate of the beacons derived from the same eyes ranges from 12.81% up to 36.17%, with the mean of 25.58%. In addition, more than half (61%) of the queries are rotated, which makes non-cyclic BGS unsuitable. The over 30% bit errors are more than BGS can readily accommodate, but the rapid speed of BGS provides an option to query multiple scans



(a) Using one type of camera



(b) Mixing two types of cameras

Figure 4.13: Histograms of two additional datasets after seven rotations

of the same eye. The false rejection rate on the first attempt is 11.32%, and is reduced to 3.28%, 0.55% on the second and third attempts respectively. While allowing three attempts, BGS reports an average delay of 128 ms, with the maximum of 883 ms. It is still significantly faster than ES; besides, it is more accurate because of the three attempts.

Regardless of the enrollment condition, BGS uses much less RAM than ES. The primary memory usage in BGS is an array that counts beacon collisions. If $c = 3$, only 2 bits are needed to record the collision count. To make it general, we use a byte, leading to $7 \times N$ bytes in total. In addition, BGS needs to store the retrieved IDs only temporarily; the same space could be re-used by subsequent retrievals (which is done automatically in Java through garbage collection). Hence, the total required memory in BGS is $\frac{N}{2^m} \times 4 + 7 \times N \approx 7 \times N$ bytes. In comparison, a time-efficient implementation of ES requires the maximum use of the available memory; it uses $512 \times N$ bytes memory in our experiment (see Section 4.3.2).

4.5 Conclusion

In this chapter, we proposed Beacon Guided Search (BGS) for searching a large iris-code database efficiently. This algorithm works by indexing, adopting a "multiple colliding segments principle" and an early termination strategy to reduce the search range dramatically. We evaluated it using 632,500 iris codes enrolled in the UAE border-control database since 2001. Two additional datasets are also included to study the variation of iris samples obtained from the same eyes under different conditions. The experiment shows that BGS is substantially faster than the currently used exhaustive search, with a negligible loss of precision; it requires much less memory; it does not depend on caching data in memory, hence obliterates the need of complex memory management; the preprocessing is simple and fast; it accommodates up to 30% bit errors in the query, as well as up to 7 cyclic rotations; the extra storage space is small and readily affordable; it supports dynamic maintenance, enabling easy indexing of new records; the high speed of BGS allows multiple acquisitions from the same eye, thus reducing the false rejection rate due to poor capture; finally, we have shown that the empirical findings match theoretical analysis.

Chapter 5

A 2-round anonymous veto protocol

Any apparently contradictory set of requirements can be met using right mathematical approach.
— **Ronald L. Rivest** [72]

This chapter[1] studies the veto problem in a multiparty setting. Essentially, this veto problem requires a secure multiparty computation on the boolean-OR function, while preserving the privacy of each input bit. It was coined as the Dining Cryptographers problem in 1988, and has attracted several solutions over the past twenty years.

We propose a new solution: Anonymous Veto Network (AV-net). The AV-net construction is provably secure under the Decision Diffie-Hellman and random oracle assumptions, and is better than past work in the following ways. It provides the strongest protection of each input's privacy against collusion; it requires only two rounds of broadcast, fewer than any other solutions; the computational load and bandwidth usage are the least among the available techniques; and the efficiency of our protocol is achieved without relying on any private channels or third parties.

5.1 Introduction

In this chapter, we study a well-known problem in cryptography: the Dining Cryptographers problem [73]. The solution to this problem has general applications in security going way beyond biometrics. Our work was initially motivated by the following imaginary scenario.

In a galaxy far far away ...

> *Several planets form an alliance, and put together their nuclear warheads under unified control. For the highest level of security, the launch button is subject to a biometrically-enabled threshold control: each delegate's biometrics act as one key, and access is only granted when all keys are correctly supplied.*
>
> *The problem arises when one delegate wishes to say "no". In fact, if he simply refuses the biometric measurement, he effectively vetoes the launch process.*

[1] The content of this chapter has been published in [13].

But, he worries that such an overt refusal might jeopardize relations with some other member states. How can he veto the launch process without revealing his identity?

In the above scenario, there are no private channels available among delegates. The only way to communicate between each other is through public announcement; and during announcement, every word uttered or message sent can be traced back to its sender. There is no external help either, as trusted third parties do not exist.

In essence, this problem requires a secure computation of the boolean-OR function, while preserving the privacy of each input bit. It was coined by Chaum as the Dining Cryptographers problem [73]; however, the "unconditional secrecy channels" assumed in [73] are no longer readily available in our case, which makes the delegate's task harder.

There have been a number of solutions in past work, ranging from circuit evaluation [92, 75] and Dining Cryptographers Network (DC-net) [73] proposed nearly twenty years ago, to several anonymous veto protocols [76, 77, 74] published in recent years. However, these techniques all have various limitations, as we discuss now.

The DC-net protocol has long been considered a classic privacy-preserving technique [79]. This protocol has two stages. First, n participants set up pairwise shared secrets through secret channels. Next, each participant P_i broadcasts a one bit message a_i, which is the XOR of all the shared one-bit secrets that P_i holds if he has no message (i.e., no veto) to send, or the opposite bit otherwise. After the broadcast round, the sent message is decoded by all participants through computing the XOR of the broadcast bits. More details can be found in [73].

However, deploying DC-nets is hampered for several reasons. First, the "unconditional secrecy channels" assumed in the protocol are difficult to achieve in practice. This problem is further compounded by the rapid increase of the total number of such channels (i.e., $O(n^2)$) when there are more participants. Second, message collisions are problematic too. Even when all participants are honest, an even number of messages will still cancel each other out, forcing retransmissions. Third, a DC-net is vulnerable to malicious jamming. For example, the last participant P_n may send $\oplus_{i=1}^{n-1} a_i$, so that the final outcome will always be '0' (i.e., non-veto). Countermeasures include setting up "traps" to catch misbehaviors probabilistically, but make the system more complex (see [73, 79, 80]).

While a DC-net is "unconditionally secure", all the other solutions are built upon public key cryptography, and are thus computationally secure. These include the circuit evaluation technique [75] and several anonymous veto protocols [76, 77, 74]. A lack of efficiency is their common problem. We will explain this in more detail in Section 5.4.

Despite the problems in the DC-net protocol, we still find it, among all the past solutions, most attractive for its simplicity and elegance. It combines all others' secret keys to encrypt data, but requires no secret keys to decrypt it. This idea is seminal, but undeservedly, has rarely been exploited in secure multiparty computations for the past twenty years.

By contrast, the mix-net protocol – a twin technique introduced by Chaum to protect anonymity – has been extensively studied and deployed in the field [78]. It encrypts messages in multiple layers using public key cryptography, and usually requires a chain of proxy servers to perform secure decryption. In comparison, a DC-net is more lightweight; it sends anonymous data by a one-round broadcast and allows rapid decryption with no servers needed.

Our solution, Anonymous Veto Network (AV-net), captures the essence of the original DC-net design [73] – it combines everyone else's public key to encrypt data, but requires no private keys to decrypt it. This, as we will show in Section 5.4, leads to the optimal efficiency of our protocol in many aspects. However, despite the similarity in the underlying design principles, the technical developments for DC-net and AV-net are completely different. In the following section, we will explain how the AV-net protocol works.

5.2 Protocol

5.2.1 Model

The setting of our protocol is simple: only an authenticated broadcast channel for every participant – no secret channels as in [73, 75] or third parties as in [76, 77]. Note that the authenticated broadcast channel is assumed in all the past work in this line of research (see Section 5.4). It suffices to know that such a channel can be realized using physical means or digital signatures [73].

Participants may collude to breach others' privacy. The *full collusion* against a participant involves all the other participants in the network. Any anonymous veto protocol, by nature, cannot preserve the vetoer's anonymity under this circumstance. However, as explained in [73], it is practically impossible to have all participants – who mistrust each other – colluding against just one; there would be no point for that person to stay in the network. Therefore, we only consider *partial collusion*, which involves only some participants, but not all.

Under the threat model of partial collusion, an anonymous veto protocol should satisfy the following three requirements.

- *Veto Privacy* – If one or more veto, the other participants cannot tell who has vetoed.
- *Veto Completeness* – If one or more veto, all participants accept the veto outcome.
- *Veto Soundness* – If the outcome is veto, all participants accept that someone has vetoed.

More requirements are defined in [76, 77] to reflect the trustworthiness of the third parties involved, but are not needed in our model.

5.2.2 Two-round broadcast

Let G denote a finite cyclic group of prime order q in which the Decision Diffie-Hellman (DDH) problem is intractable [81]. Let g be a generator in G. There are n participants, and they all agree on (G, g). Each participant P_i selects a random value as the secret: $x_i \in_R \mathbb{Z}_q$.

Round 1. *Every participant P_i broadcasts g^{x_i} and a knowledge proof for x_i.*

	x_1	x_2	x_3	x_4	x_5
x_1		$-$	$-$	$-$	$-$
x_2	$+$		$-$	$-$	$-$
x_3	$+$	$+$		$-$	$-$
x_4	$+$	$+$	$+$		$-$
x_5	$+$	$+$	$+$	$+$	

Table 5.1: A simple illustration of $\sum_{i=1}^{n} x_i y_i = 0$ for $n = 5$. The sum $\sum_{i=1}^{n} x_i \left(\sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^{n} x_j\right)$ is the addition of all the cells, where $+$, $-$ represent the sign. They cancel each other out.

When this round finishes, each participant P_i computes

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} \Big/ \prod_{j=i+1}^{n} g^{x_j}.$$

Round 2. *Every participant broadcasts a value $g^{c_i y_i}$ and a knowledge proof for c_i, where c_i is either x_i or a random value $r_i \in_R \mathbb{Z}_q$ ($r_i \neq x_i$), depending on whether participant P_i vetoes or not.*

$$g^{c_i y_i} = \begin{cases} g^{r_i y_i} & \text{if } P_i \text{ sends '1' (veto),} \\ g^{x_i y_i} & \text{if } P_i \text{ sends '0' (no veto).} \end{cases}$$

To check the final message, each participant computes $\prod_i g^{c_i y_i}$. If no one vetoes, we have $\prod_i g^{c_i y_i} = \prod_i g^{x_i y_i} = 1$. This is because $\sum_i x_i y_i = 0$ (Proposition 1). Hence, $\prod_i g^{x_i y_i} = g^{\sum_i x_i y_i} = 1$.

On the other hand, if one or more participants send the message '1', we have $\prod_i g^{c_i y_i} \neq 1$. Thus, the one-bit message has been sent anonymously.

Proposition 1 (Soundness)**.** *For the x_i and y_i defined in an AV-net, $\sum_i x_i y_i = 0$.*

Proof. By definition $y_i = \sum_{j<i} x_j - \sum_{j>i} x_j$, hence

$$\begin{aligned} \sum_i x_i y_i &= \sum_i \sum_{j<i} x_i x_j - \sum_i \sum_{j>i} x_i x_j \\ &= \sum_{j<i} \sum_{i} x_i x_j - \sum_{i<j} \sum_{j} x_i x_j \\ &= \sum_{j<i} \sum_{i} x_i x_j - \sum_{j<i} \sum_{i} x_j x_i \\ &= 0. \end{aligned}$$

Table 5.1 illustrates this equality in a more intuitive way. □

The above proposition shows that if no one has vetoed, the outcome will be non-veto. Equivalently, if the outcome is veto, someone must have vetoed. This shows that the protocol fulfills the "veto soundness" requirement defined in Section 5.2.1.

In the protocol, senders must demonstrate their knowledge of the discrete logarithms, namely the secrets x_i and c_i, without revealing them. This can be realized by using a Zero-Knowledge Proof (ZKP), a well-established primitive in cryptography [83, 84, 85, 86].

As an example, we could use Schnorr's signature [85], for it is non-interactive, and reveals nothing except the one bit information about the truth of the statement: "the sender knows the discrete logarithm". Let H be a *publicly known* secure hash function. To prove the knowledge of the exponent for g^{x_i}, one can send $(g^v, r = v - x_i h)$ where $v \in_R \mathbb{Z}_q$ and $h = H(g, g^v, g^{x_i}, i)$. This signature can be verified by anyone through checking whether g^v and $g^r g^{x_i h}$ are equal. Note that here the participant index i is unique and known to all. Adding i inside the hash function can effectively prevent a replay of this signature by other participants. Other ZKP techniques can be found in [86].

There is a variant of our protocol, in which there is no need to use any zero-knowledge proofs. Instead, participants need to commit to their announcements before each broadcast round. This can be easily realized in the physical world. For example, everyone writes down their numbers on paper before the broadcast round. However, in computer networks, this often requires additional rounds to first send the one-way hashes of the votes. This can prove costly as interaction over a network is usually the most expensive operation in distributed protocols [74].

5.3 Security analysis

In the AV-net design, each participant encrypts his vote using a collaborative form of everyone else's public key. To breach the anonymity of a participant, an observer – anyone within the broadcast range – may try to uncover the one-bit message from the announced ciphertext. In the following, we will prove that, under the DDH assumption, the proposed cryptosystem achieves *semantic security* [82]. This is equivalent to showing that under the hard-problem assumption, ciphertext is indistinguishable to observers [82].

In an AV-net, the value of y_i is determined by the private keys of all participants except P_i. The following lemma shows its security property.

Lemma 2. *In an AV-net, y_i is a secret of random value to attackers in partial collusion against the participant P_i.*

Proof. Consider the worst case where only P_k $(k \neq i)$ is not involved in the collusion. Hence x_k is uniformly distributed over $\mathbb{Z}_q$ and unknown to colluders. The knowledge proofs required in the protocol show that all participants know their private keys. Since y_i is computed from x_j $(j \neq i, k)$ known to colluders plus (or minus) a random number x_k, y_i must be uniformly distributed over $\mathbb{Z}_q$. Colluders cannot learn anything about y_i even in this worst case. □

Theorem 3 (Privacy)**.** *Under the Decision Diffie-Hellman assumption, attackers in partial collusion against P_i cannot distinguish the two ciphertexts $g^{x_i y_i}$ and $g^{r_i y_i}$.*

Proof. Besides the sent ciphertext, the data available to attackers concerning P_i include: g^{x_i}, g^{y_i} and Zero-Knowledge Proofs. The secret x_i is chosen randomly by P_i, and is unknown to the attacker. Lemma 2 shows that y_i is a random value, unknown to the

attacker. ZKPs only reveal one bit: whether the sender knows the discrete logarithm[2]; they do not leak anything about x_i, y_i, r_i. Therefore, according to the Decision Diffie-Hellman assumption, one cannot distinguish between $g^{x_i y_i}$ and a random value in the group such as $g^{r_i y_i}$ [81]. □

The above theorem states that the individual broadcast ciphertext does not leak any useful information. It is the product of all ciphertexts that tells the outcome. For each participant, the learned information from the protocol is strictly confined to the multiplied result and his own input. If a participant vetoes, the remaining participants cannot track down the vetoer without full collusion. This shows that the protocol fulfills the "veto privacy" requirement defined in Section 5.2.1.

In our protocol, since a vetoer knows his random input, he could compare it with the multiplied result and hence derive extra information: whether or not he is the only one who has vetoed. However, the derived information is only one bit, and tells nothing about who else vetoed, nor how many vetoers there are. This case is also analyzed in Groth's paper [77], and generally not considered as a threat in past work [73, 76, 77, 74].

An anonymous veto protocol must resist jamming, to which a DC-net is vulnerable. From Lemma 2, the value y_i is uniformly random over $\mathbb{Z}_q$ in the partial collusion case. Thus, given that q is a large number, say 1023-bit [81], the chance that $y_i = 0$ is negligible. This ensures that g^{y_i} is a non-identity element, hence a generator in G, for the second-round computation. Only in the full collusion case can attackers manipulate $y_i = 0$, which then makes full collusion immediately evident. The resistance to jamming in the AV-net protocol is formally proved below.

Theorem 4 (Completeness)**.** *Under the Discrete Logarithm assumption, if P_i vetoes, provided that g^{y_i} is not the identity element in the group, P_i's veto cannot be suppressed.*

Proof. Assume P_i's veto can be suppressed, and we will show that one can then solve the Discrete Logarithm problem. Given g^{r_i}, where r_i is a random value, one can compute r_i by simulating the protocol with jamming: participant P_i announces $g^{r_i y_i} = (g^{r_i})^{y_i}$, but his veto is suppressed by others. The simulator generates all other secrets, except $c_i = r_i$. By definition we have $\prod g^{c_i y_i} = 1$. That is $g^{r_i y_i} = \prod_{j \neq i} g^{-c_j y_j}$. The knowledge proofs required in the protocol show that the simulator knows the values x_j $(1 \leq j \leq n)$ and c_j $(1 \leq j \leq n$ and $j \neq i)$. Also note that g^{y_i} is not an identity element, so $y_i \neq 0$. Hence, the simulator can easily compute $r_i = y_i^{-1} \sum_{j \neq i} -c_j y_j$, where $y_i = \sum_{j<i} x_j - \sum_{j>i} x_j$. With the obtained knowledge of the r_i value, the simulation is complete. Thus, one solves the discrete logarithm of g^{r_i} by simulating the protocol with jamming. This, however, contradicts the Discrete Logarithm assumption. □

The above theorem states that jamming the protocol implies solving the Discrete Logarithm problem, which is believed to be intractable. In other words, the protocol ensures that when a participant vetoes, his veto message will be received by all. This makes the protocol fulfill the "veto completeness" requirement defined in Section 5.2.1.

Overall, Zero-Knowledge Proof (ZKP), as a crypto primitive, is important in our security analysis. Without it, several attacks would be possible. If there were no knowledge

[2]It should be noted that if we choose Schnorr's signature to realize ZKPs, we implicitly assume a random oracle (i.e., a secure hash function), since Schnorr's signature is provably secure under the random oracle model [85].

Related work	Pub Year	Rnd no	Know proof	Broad-cast	Pvt ch	Colli-sion	3rd pty	Collu-sion	Assum-ption	Sys compl
GMW [75]	1987	3	$O(n)$	yes	yes	no	no	half	trapdr	$O(n^2)$
Chaum [73]	1988	2+	–	yes	yes	yes	no	full	undond	$O(n^2)$
KY [76]	2003	3	$O(n)$	yes	no	no	yes	full	DDH	$O(n^2)$
Groth [77]	2004	$n+1$	2	yes	no	no	yes	full	DDH	$O(n)$
Brandt [74]	2005	4	4	yes	no	no	no	full	DDH	$O(n)$
AV-net	**2006**	**2**	**2**	**yes**	**no**	**no**	**no**	**full**	**DDH**	$\boldsymbol{O(n)}$

Table 5.2: Comparison to the past work

proofs in the first round, participant P_n could manipulate the value of y_1 by announcing $1/\prod_{i=2}^{n-1} g^{x_i}$, so that $y_1 = 0$. Similarly, if there were no knowledge proofs in the second round, the last participant P_n could jam the protocol by announcing $1/\prod_{i=1}^{n-1} g^{c_i y_i}$. Hence, ZKP is the technique to make the protocol self-enforcing – ensuring that participants do perform the asymmetric operations (e.g., exponentiation) as stated, rather than give out random data. It is required in all the other solutions based on public key cryptography (but not in DC-net which is built on "unconditional secrecy" [73]).

5.4 Efficiency

Over the past twenty years, there have been several techniques proposed to compute the boolean-OR function securely. They are summarized in Table 5.2.

Among all solutions, the AV-net protocol stands out for its optimal efficiency in many aspects. First, it needs only two rounds, fewer than any others. In fact, two is the best round-efficiency achievable (Theorem 5). Second, it takes only a single exponentiation to encrypt data, no matter how many participants there are. Third, the size of the broadcast ciphertext $g^{c_i y_i}$ is only half of that using the standard ElGamal encryption (see [74]). We now show that this kind of efficiency can hardly be improved further.

Theorem 5 (Lower bound of rounds)**.** *Without shared symmetric or asymmetric secrets between participants, any anonymous veto protocol relying on authenticated broadcast requires at least two rounds.*

Proof. To obtain a contradiction, assume a one-round anonymous veto protocol. Each participant holds a secret vote $v_i \in \{0, 1\}$, and has no shared secrets with others. In one round, every participant P_i broadcasts $f_i(v_i)$, where f_i is a publicly known function.

Note that the function definition f_i cannot be secret, which is known only to the P_i. Otherwise, the value of $f_i(v_i)$ would contain no useful information to the remaining participants, and could be equivalently replaced by a random value. This contradicts the veto power that P_i has on the decision making. So f_i must be a publicly known function.

The protocol allows participants to determine the Boolean-OR of all votes. Suppose every participant P_i can do so by applying a function g_i to all data available: $g_i(f_i(v_i), \ldots, f_n(v_n)) = v_1 \vee \ldots \vee v_n$. Thus participant P_i can trivially reveal the vote of another participant, say P_k, through simulating other participant's inputs as 0: $g_i(f_i(0), \ldots, f_k(v_k), \ldots f_n(0)$ $0 \vee \ldots \vee v_k \vee \ldots \vee 0 = v_k$. This contradicts the secrecy of the vote v_k, which shows that any such anonymous veto protocol requires at least two rounds. □

The AV-net design uses a ZKP primitive, which may require additional computation in verification. The exact computational cost depends on the choice of the specific ZKP

technique, whether the outcome is in doubt and the trust relationships between participants. It is also significant that since all communication is public in our protocol, any invalid ZKPs would present themselves as publicly verifiable evidence of misbehavior. With the exception of DC-net, all other solutions require ZKPs as well. As shown in Table 5.2, the AV-net protocol has the fewest zero-knowledge proofs per participant: a constant two (i.e., one for each round). Hence, under the same evaluation conditions, the verification cost in the AV-net protocol is the smallest among the related techniques. In the following, we will compare AV-net with each of the past solutions in detail.

Let us first compare AV-net with DC-net. The DC-net protocol can be implemented with different topologies. A fully-connected DC-net is "unconditionally secure", but suffers from a scalability problem when applied to a large system. For this reason, Chaum suggests a ring-based DC-net in [73], which presents a trade-off between security and system complexity. Recently, Wright, Adler, Levine and Shield showed that the ring-based DC-net described by Chaum (also by Schneier [91]) is easily attacked [88]. They compared different topologies of a DC-net and concluded that a fully-connected one is most resilient to attacks [88]. Hence, we compare AV-net with the most secure form of DC-net, i.e., a fully-connected DC-net.

As explained earlier, one of the most problematic parts in the DC-net construction is its key setup, which produces $O(n^2)$ keys. In the original description of the DC-net protocol, shared keys are established by *secretly* tossing coins behind menus. However, this requires multiple rounds of interaction between pairs of participants. It is slow and tedious, especially when there are many people involved. Other means to establish keys, as suggested by Chaum, include using optical disks or a pseudo-random sequence generator based on short keys [73]. However, such methods are acknowledged by Chaum as being either expensive or not very secure [73].

Our protocol replaces the key-setup phase in a DC-net with a simple one-round broadcast. This is achieved via public key cryptography. Although a DC-net can adopt a similar technique – the Diffie-Hellman key exchange protocol – to distribute keys, its use of the underlying technology is quite different from ours. Suppose a DC-net uses Diffie-Hellman to establish keys[3]. Each participant must perform $O(n)$ exponentiations in order to compute the shared keys with the remaining $n-1$ participants. However, our protocol requires only one exponentiation for each of the two rounds, no matter how many participants there are (the cost of multiplication is negligible as compared to that of exponentiation).

Secure circuit evaluation is an important technique for secure Multi-Party Computation (MPC) applications. It evaluates a given function f on the private inputs $x_1, \ldots, x_n$ from n participants. In other words, it computes $y = f(x_1, \ldots, x_n)$, while maintaining the privacy of individual inputs. At first glance, it appears trivial to apply this technique to build a veto-protocol – one only needs to define f as the boolean-OR function. However, this general technique proves to be unnecessarily complex and expensive in solving specific functions [74].

Yao [92] first proposed a general solution for the secure circuit evaluation in the two-party case. Later, Goldreich, Micali, and Wigderson extended Yao's protocol to the multiparty case, and demonstrated that any polynomial-time function can be evaluated securely in polynomial time provided the majority of the players are honest [75]. This

[3]Note that in this case the DC-net protocol is no longer unconditionally secure, as the Diffie-Hellman key exchange essentially rests on the Decision Diffie-Hellman assumption [81].

conclusion follows from the general assumption of the existence of a trap-door permutation. Although the general solution proposed in [75] uses an unbounded number of rounds of interaction, it was later tightened to a constant number of rounds [89]. Recently, Gennaro, Ishai, Kushilevitz, and Rabin showed that three rounds are sufficient for arbitrary secure computation tasks [90].

Although the general GMW solution is versatile, it suffers from the way this technique is evolved – by extending the general solution in the two party case to pairs in the multiparty case. This leads to $O(n^2)$ system complexity. First, it requires pairwise private channels among participants [75], which could prove problematic, especially when there are many participants. Second, it requires a large amount of traffic. Although the protocol could be completed with only three rounds [90], each round includes not only the broadcast of public messages, but also the transmission of private messages to everyone else through the pairwise secret channels [90]. The total amount of sent data is $O(n^2)$. Third, it is no longer resistant to collusion when more than half of the participants collude. In such a case, the colluders can easily breach the privacy of other inputs.

Our work shows the benefits of designing a protocol directly in the multiparty context. It has linear complexity, requires no pairwise secret channels, and provides full protection against collusion, instead of half. How to apply the underlying design principle in the AV-net protocol to compute more general functions seems worth exploring in future research.

All the other techniques in Table 5.2 are based on the Decision Diffie-Hellman assumption [76, 77, 74]. The first rounds of those protocols are the same as in the AV-net protocol: broadcasting public keys. This allows more direct comparisons with the AV-net protocol, as shown below.

Kiayias and Yung investigated the Distributed Decision Making problem, and proposed a 3-round veto protocol [76]. They used a third party – a bulletin board server – to administer the process. The bulletin board server is a common way to realize a reliable broadcast channel. However, the server is needed for some other reasons. In the Kiayias-Yung protocol, each participant publishes $O(n)$ data. The final result on the veto decision is computed from $O(n^2)$ data. In large networks, it would be too demanding for individuals to store and compute such data. The server is a natural choice to perform the intermediary processing.

Groth modified the Kiayias-Yung veto protocol in order to reduce the system complexity [77]. His approach is to trade off round-efficiency for less traffic and computation. As a result, Groth's veto protocol allows each participant to publish a smaller amount of data, but requires participants to send their messages one after another, as one's computation depends on the result sent by the previous participant. Hence, instead of finishing the protocol in 3 rounds as in [76], Groth's veto protocol requires $n + 1$ rounds, where n is the number of participants.

Brandt studied the use of ElGamal encryption techniques for multiparty computation applications, and gave a 4-round veto protocol [74]. The performance of his solution, among others, is the closest to ours. However, it requires four rounds while ours only needs two.

The difference in rounds lies in the way the veto messages are encrypted. In Brandt's veto protocol, the first round is the same as in an AV-net: all participants broadcast their public keys. It requires one exponentiation to compute a public key. In the second round, each participant applies the standard ElGamal encryption algorithm to encrypt an explicit

message: "veto" or "non-veto". Such an encryption requires two exponentiations. The third and fourth rounds are arranged to decrypt the messages, while preserving the privacy of individual inputs. It requires two and one exponentiations in each round respectively. In addition, each round requires a zero-knowledge proof per participant, which amounts to four in total. Even without counting the knowledge proofs, each participant needs to performs six exponentiations in Brandt's protocol.

The novelty of our protocol is that the veto message is encrypted in a very implicit way: by raising a base to one of two different powers. As a result, the veto decision can be immediately decoded after the second broadcast. It requires only two exponentiations in total, as compared to six in Brandt's protocol. Besides computational load, the traffic generated is also far less in our protocol.

Interestingly, running the AV-net protocol is like playing a juggling game among a group of people – if we regard a public key as a "ball". In the first round, everyone throws a random "ball", and in the second round, each participant combines the received "balls", and throws a new "ball". We call this technique "public key juggling". A recent paper on solving the Password Authenticated Key Exchange problem uses exactly the same technique [10].

5.5 Conclusion

In this chapter, we proposed the Anonymous Veto Network (AV-net) protocol, which allows one delegate to send the veto message anonymously. This protocol is not only provably secure, but also optimally efficient. It is compared with other solutions proposed in the past twenty years, including the circuit evaluation techniques, Dining Cryptographers Network, and several anonymous veto protocols. Our technique does not require any private channels or third parties; it has no message collisions, hence requires no retransmissions; being semantically secure, it provides the strongest protection of vetoer's anonymity until all the other participants are compromised; it resists robustly against jamming, hence ensures each participant's veto power; the execution of the protocol requires only two rounds, fewer than any other solutions; and finally, the computational load, bandwidth usage, and cost of verifying zero-knowledge proofs are also less than previous techniques and very close to the best possible.

Chapter 6

Conclusion

6.1 Summary

The goal of this research project was to explore how to use fuzzy data in security mechanisms effectively. For that, we made two primary contributions: 1) bridging the gap between the data fuzziness and the exactitude of cryptography; 2) devising an efficient search algorithm for a large fuzzy database. In addition, we designed an exceptionally efficient protocol for sending anonymous messages among a group of participants.

We started, in Chapter 2, with a survey of different types of fuzzy data: iris codes, token fingerprints, paper fingerprints and music fingerprints. We demonstrated how to use an ordinary camera to photograph the random translucency patterns of ordinary paper. This shows that an object's unique features can be captured at low cost.

Next, in Chapter 3, we worked on combining cryptography with biometrics: how to generate a repeatable string from a biometric in such a way that it can be revoked. Based on the study of error patterns within iris codes, we devised a two-layer error correction scheme, which first corrects the background-noise errors of iris codes using a Hadamard code, and then the burst errors using a Reed-Solomon code. The experiment results show that a 140-bit error-free key can be reliably generated from genuine iris codes with a 99.5% success rate.

In designing the above system, we also took into account security engineering aspects, including: the irrevocability of biometrics, their low level of secrecy, the requirement of key diversity, correlations within biometric data, and the likely industrialization of attacks when biometrics become widely used. To tackle these issues, we proposed a two-factor scheme, based on biometrics and a token, and show how it can be easily extended to a three-factor scheme with an added password. In each case, we argued that the security is the best achievable – all factors are needed to compromise the key. And the key can be easily updated or revoked.

In Chapter 4, we went on to study the search problem in iris recognition. Currently, all public deployments of iris recognition adopt an exhaustive search strategy to look for matching iris codes. That is too expensive for a large-scale deployment. Our proposed algorithm, Beacon Guided Search (BGS), is devised based on a new "multiple colliding segments principle". Applying this principle, together with an early termination strategy, BGS shrinks the search range dramatically, followed by a close examination of a few selected records. We evaluated BGS using 632,500 real-world iris codes, showing a

substantial speed-up over the exhaustive search with a negligible loss of precision. In addition, we showed that the experimental findings match theoretical analysis.

Finally, in Chapter 5, we studied the veto problem in a biometrically-enabled threshold control scheme. In essence, this problem requires a secure multiparty computation on the boolean-OR function, and has been studied for the past twenty years with a few solutions proposed. Our solution, Anonymous Veto Network (AV-net), captures the essence of Chaum's original DC-net design: it combines participants' secret keys to encrypt data, but requires no secret keys to decrypt it. Overall, the AV-net construction achieves the best efficiency of all available solutions; in addition, it does not require any private channels or third parties.

To sum up, we proposed three techniques in this dissertation – the first two on applying biometrics to enhance authentication in cryptography with or without a central database present (Chapter 3 and 4) and the third on applying cryptographic primitives to address the privacy issues in biometric enrollments (Chapter 5). Between them, our ideas show how to link up biometrics with traditional information security mechanisms in ways that are efficient, robust and scalable.

6.2 Future work

Future work is suggested as follows.

- The paper-fingerprinting method presented in Chapter 2 would be useful for applications requiring the authenticity of paper documents (such as passports). To make it viable in practice, further work need be done to encode the captured patterns into a compact *paper code*. It could also be combined with the techniques introduced in Chapter 3 or 4, depending on whether a central database is present.

- In Chapter 3, it is possible to let the Hadamard decoder return a list of probable codewords, instead of just one. Hence, by applying exhaustive search among the codewords or list decoding techniques [23, 24], the error correction capability can be improved. This will allow fine tuning performance in practical deployments.

- In Chapter 4, since the mask functions inform the varying qualities of iris-code bits, we could use this information to determine whether some beacons are more reliable than others. This makes it possible to assign different weights to beacon collisions, so that the counter is increased differently for each collision. However, finding the optimal weights is not trivial.

- It is worth trying to extend the techniques presented in Chapter 3 and 4 to fingerprint biometric – deriving error-free keys from fingerprints, and searching fingerprint databases efficiently. A technical challenge will then be how to deal with the non-linear distortions that exist in fingerprints, but not in iris codes.

- Finally, in Chapter 5, it might be possible to add a tallying function to the AV-net protocol, so participants know how many veto votes there are. This would be useful for threshold veto schemes. It might also be interesting to apply the underlying design principle of AV-net to compute more general functions.

Bibliography

[1] R.J. Anderson, *Security Engineering : A Guide to Building Dependable Distributed Systems*, New York, Wiley 2001.

[2] R.J. Anderson, "Why cryptosystems fail," Proceedings of the 1st ACM conference on Computer and Communications Security, pp. 215–227, 1993.

[3] J. Daugman, "Biometric decision landscapes," Technical Report UCAM-CL-TR-482, Computer Laboratory, University of Cambridge, 2000.

[4] J. Daugman, "Gabor wavelets and statistical pattern recognition," *The Handbook of Brain Theory and Neural Networks*, 2nd edition, MIT press, pp. 457–463, 2002.

[5] J. Daugman, "The importance of being random: statistical principles of iris recognition," *Pattern Recognition,* Vol. 36, No. 2, pp. 279–291, 2003.

[6] J. Daugman, "Demodulation by complex-valued wavelets for stochastic pattern recognition," *International Journal of Wavelets, Multi-resolution and Information Processing*, Vol. 1, No. 1, pp. 1–17, 2003.

[7] J. Daugman, "Probing the uniqueness and randomness of IrisCodes: Results from 200 billion iris pair comparisons," *Proceedings of the IEEE*, Vol. 94, No. 11, pp. 1927–1935, 2006.

[8] J. Daugman, "Results from 200 billion iris cross-comparisons," Technical Report UCAM-CL-TR-635, University of Cambridge, 2005.

[9] F. Hao, J. Daugman, P. Zieliński, "A fast search algorithm for a large fuzzy database," to be published in *IEEE Transactions on Information Forensics and Security*, June 2008.

[10] F. Hao, P. Ryan, "Password Authenticated Key Exchange by Juggling," Proceedings of the 16th Workshop on Security Protocols, Cambridge, UK, April 2008.

[11] F. Hao, "Kish's key exchange scheme is insecure," *IEE Information Security*, Vol. 153, No. 4, pp. 142–142, 2006.

[12] F. Hao, R. Anderson, J. Daugman, "Combining crypto with biometrics effectively," *IEEE Transactions On Computers*, Vol. 55, No. 9, pp. 1081–1088, 2006.

[13] F. Hao, P. Zieliński, "A 2-round anonymous veto protocol," the 14th International Workshop on Security Protocols, to appear in LNCS, 2006.

[14] 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.

[15] F. Hao, C.W. Chan, "Online signature verification using a new extreme points warping technique," *Pattern Recognition Letters*, Vol. 24, No. 16, pp. 2943–2951, 2003.

[16] D. Stinson, *Cryptography Theory and Practice*, Third edition, CRC Press, 2005.

[17] H. Delfs, H. Knebl, *Introduction to Cryptography*, Springer, 2006.

[18] N. Koblitz, A.J. Menezes, "Another look at "provable security"," *Journal of Cryptography*, Vol. 20, No. 1, pp. 3–37, 2007.

[19] I.L. Chuang, Y. Yamamoto, "Simple quantum computer," *Physics Review*, Vol. 52, No. 5, pp. 3489–3496, 1995.

[20] G. Brassard, I.L. Chuang, S. Lloyd, and C. Monroe, "Quantum computing," the Ninth Annual Frontiers of Science Symposium, 1997.

[21] C.H. Bennett, "Quantum cryptography using any two nonorthogonal states," *Physics Review Letters*, Vol. 68, No. 21, pp. 3121–3124, 1992.

[22] M. Bond, "Understanding security APIs," PhD thesis, University of Cambridge, 2004.

[23] V. Guruswami, M. Sudan, "List decoding algorithms for certain concatenated codes," Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, pp. 181–190, 2000.

[24] P. Elias, "Error-correcting codes for list decoding," *IEEE Transactions on Information Theory*, Vol. 37, No. 1, pp. 5–12, 1991.

[25] L.B. Kish, "Totally secure classical communication utilizing Johnson (-like) noise and Kirchoff's law," *Physics Letters*, Vol. 352, pp. 178–18, 2006.

[26] J.D.R. Buchanan, R.P. Cowburn, A.V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D.A. Allwood, M.T. Bryan, "'Fingerprinting' documents and packaging," *Nature*, Vol. 436, No. 28, p. 475, 2005.

[27] Private communication with Russell P. Cowburn.

[28] R. Pappu, B. Recht, J. Taylor, J. Gershenfeld, "Physical one-way function," *Science*, Vol. 297, No. 5589, pp. 2036–2030, 2002.

[29] R. Pappu, "Physical one-way function," PhD thesis, MIT, 2001.

[30] Nikon Coolpix 4500 User's Guide, available at (accessed on 23 April 2007): `http://www.nikonimaging.com/global/products/digitalcamera/coolpix/4500/index.htm`

[31] M. Miller, M.A. Rodriguez, I.J. Cox, "Audio fingerprint: nearest neighbor search in high dimensional binary spaces," IEEE Multimedia signal Processing Workshop, 2002.

[32] J. Haitsma, T. Kalker, "A highly robust audio fingerprinting system," International Symposium on Musical Information Retrieval, pp. 144–148, 2002.

[33] E. Métois, P.M. Yarin, N. Salzman, and J.R. Smith, "FiberFingerprint identification," Proceedings of the Third Workshop on Automatic Identification, Tarrytown, NY, pp. 147–154, 2002.

[34] Oana G. Cula, Kristin J. Dana, Frank P. Murphy and Babar K. Rao, "Bidirectional imaging and modeling of skin texture," The Third International Workshop on Texture Analysis and Synthesis, pp. 12-18, Nice, France, 2003.

[35] Oana G. Cula, Kristin J. Dana, Frank P. Murphy and Babar K. Rao, "Skin Texture Modeling," International Journal of Computer Vision, Vol. 62, No. 1, pp. 97–119, 2005.

[36] K.W. Bowyer, K. Hollingsworth, and P.J. Flynn, "Image understanding for iris biometrics: a survey," University of Notre Dame, CSE Technical Report, 2007.

[37] T. Matsumoto, H. Matsumoto, K. Yamada, S. Hoshino, "Impact of artificial gummy fingers on fingerprint systems," Proceedings of SPIE Vol. 4677, pp. 275–289, 2002.

[38] T. Matsumoto, "Gummy finger and paper iris: an update," the 2004 Workshop on Information Security Research, 2004.

[39] B. Toth, "Biometric liveness detection," *Information Security Bulletin*, Vol. 10, pp. 291–297, 2005.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy and B.V.K.V. Kumar, "Biometric Encryption," *ICSA Guide to Cryptography*, McGrow-Hill, 1999, also available at (accessed on 23 April 2007)
`http://www.bioscrypt.com/assets/Biometric_Encryption.pdf`

[48] 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.

[49] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," Proceeding of the 6th ACM Conference on Computer and Communication Security, CCCS, 1999.

[50] A. Juels and M. Sudan, "A fuzzy vault scheme," Proceedings of IEEE International Symposium on Information Theory, 2002.

[51] G.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.

[52] G.I. Davida, Y. Frankel, B.J. Matt, "On enabling secure applications through off-line biometric identification," IEEE Symposium on Security and Privacy, pp. 148–157, 1998.

[53] David Wheeler, "Protocols using keys from faulty data," Security Protocols Workshop, Cambridge, 2001.

[54] S.S. Agaian, *Hadamard Matrices and Their Applications,* LNM, Springer Verlag, 1985.

[55] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-correcting Codes*, North Holland, 1991.

[56] R.J. McEliece, *The Theory of Information and Coding*, Cambridge University Press, 2002.

[57] 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, pp. 523–540.

[58] X. Boyen, "Reusable cryptographic fuzzy extractors," CCS 2004, pp. 82–91, ACM Press.

[59] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and Adam Smith, "Secure remote authentication using biometric data," Eurocrypt 2005, Lecture Notes in Computer Science 3494, pp. 147–163.

[60] P. Zezula, G. Amato, V. Dohnal, M. Batko, *Similarity Search: The Metric Space Approach*, Springer, 2006.

[61] B. Bustos, G. Navarro, E. Chávez, "Pivot selection techniques for proximity searching in metric spaces," Pattern Recognition Letters, Vol. 24, No. 14, pp. 2357–2366, 2003.

[62] P. Ciaccia, M. Patella, P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces," the 23th VLDB Conference, pp. 426–435, 1997.

[63] M. Batko, C. Gennaro, P. Savino, P. Zezula, "Scalable similarity search in metric spaces," the 6th DELOS Workshop on Digital Library Architectures, pp. 213-224, 2004.

[64] P. Indyk, R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," the 30th Annual ACM Symposium on Theory of Computing, pp. 604–613, 1998.

[65] A. Gionis, P. Indyk, R. Motwani, "Similarity search in high dimensions via hashing," the 25th VLDB Conference, 1999.

[66] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," the 12th Symposium on Computational Geometry, pp. 253–262, 2004.

[67] A. Andoni, P. Indyk, "E^2LSH User Manual," June 2005, available at LSH manual (accessed on 23 April 2007): `http://web.mit.edu/andoni/www/LSH/manual.pdf`.

[68] The British government website about the ID card scheme (accessed on 23 April 2007): `http://www.identitycards.gov.uk/`

[69] The official site about the Ration Card scheme (accessed on 23 April 2007): `http://hyderabad.ap.nic.in/rationcard.html`

[70] N.K. Ratha, K. Karu, S. Chen, A.K. Jain, "A real-time matching system for large fingerprint databases," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, 1996, pp. 799–813.

[71] M. Mattews, J. Cole, J.D. Gradecki, *MySQL and Java Developer's Guide*, Ryan Publishing Group, Inc., 2003.

[72] R. Rivest, "The early days of RSA – history and lessons," Presentation at the ACM Turing Award Lecture. Available at (accessed on 23 April 2007): `www.acm.org/fcrc/PlenaryTalks/rivest.pdf`.

[73] D. Chaum, "The dining cryptographers problem: unconditional sender and recipient untraceability," *Journal of Cryptology*, Vol. 1, No. 1, pp. 65–67, 1988.

[74] F. Brandt, "Efficient cryptographic protocol design based on distributed El Gamal encryption," Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC), LNCS 3935, pp. 32–47, 2005.

[75] O. Goldreich, S. Micali and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," Proceedings of the 19th Annual ACM Conference on Theory of Computing, pp. 218–229, 1987.

[76] A. Kiayias and M. Yung, "Non-interactive zero-sharing with applications to private distributed decision making," Financial Cryptography 2003, LNCS 2742, pp. 303–320, 2003.

[77] J. Groth, "Efficient maximal privacy in boardroom voting and anonymous broadcast," Financial Cryptography 2004, LNCS 3110, pp. 90–104, 2004.

[78] D. Chaum, "Untraceable electronic email, return addresses, and digital pseudonyms," *Communications of the ACM*, Vol. 24, No. 2, pp. 84–88, 1981.

[79] P. Golle and A. Juels, "Dining Cryptographers Revisited," Eurocrypt'04, LNCS, vol. 3027, pp. 456–473, 2004.

[80] M. Waidner and B. Pfitzmann, "The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability," Eurocrypt'89, LNCS, vol. 434, p. 690, 1989.

[81] D. Boneh, "The Decision Diffie-Hellman Problem," Proceedings of the Third International Symposium on Algorithmic Number Theory, LNCS 1423, pp. 48–63, 1998.

[82] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, Vol. 28 pp. 270–299, 1984.

[83] D. Chaum, J.H. Evertse, J.V.D. Graaf and R. Peralta, "Demonstrating possession of a discrete log without revealing it," Advances in Cryptology Crypto'86, LNCS 263, pp. 200–212, 1987.

[84] D. Chaum, J.H. Evertse and J.V.D Graaf, "An improved protocol for demonstrating possession of a discrete logarithm and some generalizations," Advances in Cryptology Eurocrypt'87, LNCS 304, pp. 127–141, 1988.

[85] C.P. Schnorr, "Efficient signature generation by smart cards," Journal of Cryptology, Vol. 4, No. 3, pp. 161–174, 1991.

[86] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," Technical Report TR 260, Department of Computer Science, ETH Zürich, March 1997.

[87] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," Proceedings on Advances in Cryptology, Crypto'86, LNCS 0263, pp. 186–194, 1987.

[88] M. Wright, M. Adler, B.N. Levine, and C. Shields, "The predecessor attack: an analysis of a threat to anonymous communications systems," ACM Transactions on Information and Systems Security (TISSEC), Vol. 7, No. 4, 2004.

[89] D. Beaver, S. Micali and P. Rogaway, "The round complexity of secure protocols," Proceedings of the twenty-second annual ACM Symposium on Theory of Computing, pp. 503–513, 1990.

[90] R. Gennaro, Y. Ishai, E. Kushilevitz and T. Rabin. "On 2-round secure multiparty computation," Crypto 2002, LNCS 2442, pp. 178–193, 2002.

[91] B. Schneier, *Applied Cryptography*, J. Wiley and Sons, 1996.

[92] A. Yao, "How to generate and exchange secrets," Proceedings of the twenty-seventh annual IEEE Symposium on Foundations of Computer Science, pp. 162–167, 1986.