SECURE KEY AGREEMENT USING PURE BIOMETRICS

Dilara Akdoğan

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University

December, 2015

SECURE KEY AGREEMENT
USING PURE BIOMETRICS

APPROVED BY:

Prof. Albert Levi .....
(Thesis Supervisor)

Prof. Berrin Yanıkoğlu .....

Asst. Prof. Ahmet Onur Durahim .....

DATE OF APPROVAL: .....

SECURE KEY AGREEMENT USING PURE BIOMETRICS

Dilara Akdoğan

Computer Science and Engineering, Master's Thesis, 2015

Thesis Supervisor: Prof. Albert Levi

Abstract

In this thesis, we propose a novel secure key agreement protocol that uses biometrics with unordered set of features. Our protocol enables the user and the server to agree on a symmetric key, which is generated by utilizing only the feature points of the user's biometrics. It means that our protocol does not generate the key randomly or it does not use any random data in the key itself. As a proof of concept, we instantiate our protocol model using fingerprints. In our protocol, we employ a threshold-based quantization mechanism, in order to group the minutiae in a predefined neighborhood. In this way, we increase the chance of user-server agreement on the same set of minutiae. Our protocol works in rounds. In each round, depending on the calculated similarity score on the common set of minutiae, the acceptance/rejection decision is made. Besides, we employ multi-criteria security analyses for our proposed protocol. These security analyses show that the generated keys possess acceptable randomness according to Shannon's entropy. In addition, the keys, which are generated after each protocol run, are indistinguishable from each other, as measured by the Hamming distance metric. Our protocol is also robust against brute-force, replay and impersonation attacks, proven by high attack complexity and low equal error rates. At the end, the complexity analysis and the memory requirements of the protocol are discussed and it is showed that they are in acceptable limits. As shown by comparative analyses, this work outperforms the existing fuzzy vault method in terms of verification performance and the attack complexity.

Saf Biyometrik Kullanımıyla Güvenli Anahtar Anlaşması

Dilara Akdoğan

Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans, 2015

Tez Danışmanı: Prof. Dr. Albert Levi

Özet

Bu çalışmada, özellik noktaları sırasız eleman dizilerinden oluşan biyometriklerin kullanımı ile güvenli anahtar mutabakatı protokolü tasarlanmıştır. Önerilen protokol, servis sağlayıcı ile kullanıcının yalnızca biyometrik özellik noktalarını kullanarak simetrik bir anahtar oluşturmalarını sağlamaktadır. Diğer bir deyişle, protokol anahtarı rastgele oluşturmamakta veya anahtar oluşturma sürecinde hiçbir rastgele veriden yararlanmamaktadır. Önerilen yöntemin kuram ispatı olarak, bu protokol modeli parmak izi kullanılarak gerçeklenmiştir. Protokolde eşik tabanlı nicemleme kullanılarak belirli bir komşuluk ilişkisi içerisindeki özellik noktaları gruplanmışlardır. Bu sayede, servis sağlayıcı ile kullanıcı arasında ortak özellik noktaları üzerinde anlaşılması ihtimali artırılmıştır. Protokol rauntlar halinde çalışmaktadır. Her rauntta ortak bulunan özellik noktası sayısı kullanılarak bir benzerlik skoru hesaplanmakta ve bu skora göre kullanıcının sisteme kabul/ret kararı verilmektedir. Bunun yanı sıra, çoklu değerlendirme ölçütleri kullanılarak güvenlik analizleri yapılmıştır. Güvenlik analizleri, oluşturulan anahtarların rastgelelik oranlarının Shannon'un entropisi metriğine göre güvenli seviyelerde olduğunu göstermiştir. Ayrıca protokolün sonunda oluşturulan tüm anahtarların birbirlerine benzer olmadıkları Hamming uzaklık metriği ile gösterilmiştir. Öte yandan protokolün, kaba kuvvet saldırısı, tekrarlama ve taklit etme ataklarına karşı dayanıklı olduğu yüksek atak zorluğu ve düşük hata oranları ile kanıtlanmıştır. Protokolün

karmaşıklığının ve hafıza gereksinimlerinin sistemin gerçeklenmesine uygun olduğu raporlanmıştır. Son olarak, protokol ile literatürde var olan bir yöntemin karşılaştırılması ile, protokolün performans ve atak dayanıklılığı açısından daha başarılı olduğu gösterilmiştir.

in memory of my father...

Acknowledgments

This thesis would not have been possible without the support of my supervisor, committee, friends and family.

Foremost, I would like to express the sincere gratitude to my thesis supervisor Prof. Albert Levi. The presented work existed and developed with the help of his knowledge, as well as his continues guidance, encouragement and patience throughout my masters studies. I also would like to thank my thesis jury, Prof. Berrin Yanıkoğlu and Asst. Prof. Ahmet Onur Durahim for their valuable suggestions and inquiries.

I am grateful to all the members of our Cryptography and Information Security Lab for the great environment they provided in terms of both research and friendship. I am also thankful to my project partners Duygu Karaoğlan Altop, Naim Alperen Pulur and Laleh Eskandarian. They always supported and guided me with their valuable ideas. Every one of them is important to me, but Duygu Karaoğlan Altop deserves my special appreciation. I am so grateful for her guidance and clever ideas, her unconditional moral and material support aided me during my studies. In addition, I would like to thank my old friends Gamze Tillem, Naim Alperen Pulur, Atıl Utku Ay and Berkay Dinçer for their presence. They always stand by me whenever I need them. I also would like to thank my best friend Deniz Antepoğlu and my beloved Mert Konukoğlu for their presence and moral support.

Last but not least, I would like to express my special appreciation and thanks to my parents; my mother Nimet Akdoğan and my father Uğur Akdoğan. I would not be here without the unlimited love and support of my parents provided throughout my life.

I would like to thank TÜBİTAK BİDEB 2228-A for financially supporting me.

This thesis has been supported by TÜBİTAK under grant 114E557.

Contents

List of Algorithms

List of Figures

List of Tables

1 Introduction

In generic cryptographic applications, unique and user-specific secret keys are used. However, these keys can be stolen, lost or willingly shared. On the other hand, biometric traits are used to identify or verify users, since they are strictly bound to the user. The reason behind this is that the biometric traits are unique physiological or behavioral characteristics of individuals. In order to provide higher security and privacy, biometrics and cryptography are combined as this combination provides the binding of user's personal characteristics to cryptographic keys. The combination of biometrics and cryptography is referred to as *crypto-biometric* or *bio-cryptographic systems*. This way, the secret keys, which are used for encryption and decryption in cryptographic applications, are derived from the biometric data with the help of their unique features.

The key used in a cryptographic application must be secure. A secure key must be random enough, contain sufficient entropy and be distinct from each other. Additionally, the length of the key must be long enough for not being guessed simply by trying all possibilities. Due to the invariant nature of the biometrics, satisfying the requirements of a secure key derived from the biometrics is a challenging task in bio-cryptographic systems. In addition, the main problem in bio-cryptographic approaches is that the cryptographic applications require exactly the same keys in encryption and decryption operations, whereas in biometric applications, such an exact match is not needed. This difference in their approaches should also be considered while designing bio-cryptographic systems.

Each biometric data has its own distinctive features. These features can be represented with either ordered or unordered sets. Ordered sets are typically binary strings. Iris is an example of a biometric with ordered features, as iris code is a binary string

retrieved from the unrolled iris texture. On the other hand, unordered sets are generally a list of points on a coordinate system with insignificant orders. Biometric features from which independent points can be extracted are examples of unordered feature sets, such as fingerprints.

1.1 Our Contribution Summary

In this thesis, we propose a novel biometric key agreement protocol that uses unordered set of features. As an example, we implement and evaluate our protocol using fingerprints, which are represented with unordered set of minutiae points. Our protocol generates the keys by using only the minutiae points, without any other helper component. In this key agreement protocol, hash functions and threshold mechanisms are employed. Moreover, in order to hide the genuine minutiae points, fake minutiae points are generated according to a strategy. This strategy is developed to properly manage the trade-off between information leakage to the attacker and acceptable verification results. For this reason, a distance threshold and a neighborhood relation are defined such that there cannot be more than one point (genuine and/or fake) in a pre-defined distance neighborhood. This process is analogous to quantization. Thus, in the rest of the thesis, we will call this method as quantization.

Our key agreement protocol runs in a round-manner such that at each round, the user and the server tries to find a common set of minutiae points. At the end of the protocol, either the user is rejected, due to the reason that the similarity score is below the acceptance threshold, or the user and the server agree on a secure symmetric key.

We analysed the security performance of our system from different perspectives. From biometrics point of view, our model shows high verification performance, proven by low Equal Error Rates (EER). We also analysed the resistance of our protocol against some known attacks, such as brute-force, replay and impersonation attacks. Moreover, the quality of the agreed keys is analysed in terms of randomness and distinctiveness. These analyses show that our system is quite resistant to these attacks; the generated keys are random enough to be used as cryptographic keys; and each key is distinct from the other agreed keys. It is also important to note that this protocol generates different

keys after each protocol run using a time invariant biometric data. In addition, we report the complexity and the memory requirements analyses of our protocol and show that they are suitable for real time applications. Finally, we compare our protocol with an existing key binding method, called fuzzy vault [1], and show that our protocol outperforms this method in terms of performance and attack complexity. The preliminary results of this thesis are presented in [2].

1.2 Outline of Thesis

The rest of this thesis is organized as follows. Section 2 gives background information and Section 3 summarizes the related work in the literature. In Section 4, we introduce our proposed secure key agreement protocol. Section 5 evaluates the performance of our proposed protocol, discusses its security, complexity and memory requirement analyses, as well as providing comparison results with an existing method. Finally, in Section 6 we conclude the thesis and provide some future works.

2 Background Work

Firstly, this section briefly discusses biometrics and cryptography. After the security mechanisms, which are used in this work, are shortly explained, the details of the combination of biometrics and cryptography are mentioned in this section.

2.1 Biometrics

Identifying or verifying people can be done in different ways. Human brain can recognize people by looking at their faces or by hearing their voices. However, in an automatic world, automatic recognition of people by using biometrics is gaining more and more importance and interest. In [3], *biometric systems* are defined as automated methods of identity verification of a person based on their distinctive physiological characteristics, like fingerprint, iris or face, or their distinctive behavioral characteristics, like signature, voice or keystroke dynamics.

There are two types of biometric systems, namely *verification* and *identification*. If a person claim to have an identity, the verification system checks the correctness of this claim. In other words, a single match is performed in order to authenticate the user. Verification systems are needed mostly in credit card and banking transactions, network logins and cellular phones. On the other hand, in an identification system, the personal biometric characteristic is presented and the system tries to find this particular person's identity in a set of characteristics. Identification systems are mostly used in criminal investigations and border controls. The biometric verification is a more difficult task than the biometric identification, since it requires a database search, instead of a single match. Above all else, in order to run an identification or a verification process, firstly the person should be enrolled to the system. At the enrollment stage, distinctive

features of the biometric trait are extracted and stored in the system's database. After that, the person can use the application by proving the identity at the verification or identification stage. In both of these processes, an algorithm or a function is applied to the existing and the query templates, and their similarity is measured. If they are close enough, in other words, if their difference is below a threshold, the user is expected to be identified or verified.

As also mentioned above, biometrics can be physiological or behavioral characteristics of a person. Figure 1 shows some examples of these characteristics. Their usage in biometric identification and/or verification systems along with their strong and weak aspects are summarized in [4]. As an example of physiological characteristics, fingerprints are represented with detail points on the ridges, called minutiae. Fingerprints are the most commonly employed biometric data, because they have been used by police to recognize criminals from far in the past. Hence, people are more familiar to it. Iris is represented using iris code, which is a binary string that shows the patterns on the iris. Although, iris patterns are more detailed than fingerprints, establishing a perfect acquisition environment is a bit harder for iris than fingerprint scanning. Besides, faces are the mostly used recognition traits of human brains. Digital system recognizes the faces by the relative positions, sizes and shapes of the features. Therefore, face recognition is very sensitive to changes, such as head position, expressions and facial hair. Hand geometry builds a three dimensional template of the hand. But a hand geometry biometric system can be faked by a fabricated hand and is not very distinctive among people. As an alternative to the physiological characteristics, the behavioral characteristics are easier to capture. Signature dynamic is an example of the behavioral characteristic, which is needed mostly in financial applications, in order to automate the signature verification process. Besides, voice can be recorded using a regular device, and after that the waveforms of the voice are used in the recognition process. Although the voice can be imitated, the characteristics of speech involves some physiological aspects as well, and they are almost impossible to impersonate. Keystroke dynamics are very easy to capture from the keyboard. Its verification process does not disturb the person to be identified, because it is hidden in the regular flow of an application.

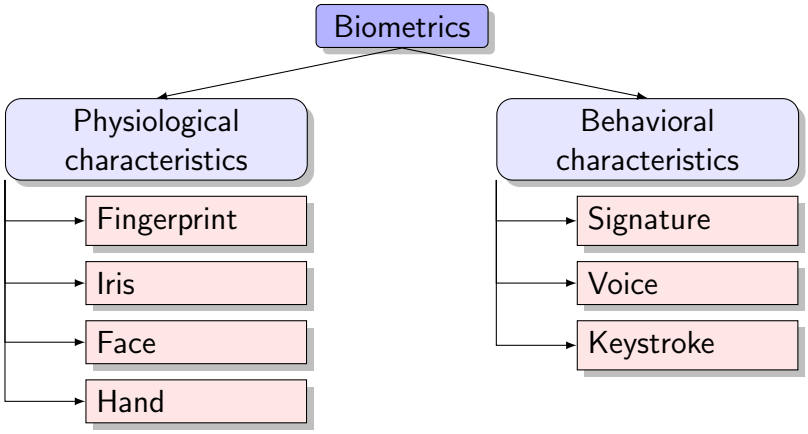

Figure 1: Biometric data types

The main difference between physiological and behavioral traits is their persistence. Physiological characteristics are more stable than the behavioral ones. They stay almost the same throughout a person's life, unless getting an injury. On the other hand, behavioral characteristics change over time, due to both controllable and uncontrollable reasons, such as emotions, neurological diseases, hoarseness, etc. Therefore, biometric systems which use behavioral characteristics update their template after each use. Hence, behavioral biometric systems work better when they are used frequently [3]. The systems that use physiological characteristics perform better than behavioral ones in terms of permanence, universality, distinctiveness and performance [5]. On the other hand, behavioral characteristics are easier to collect and people are more willing to use them [5]. As a result, while designing a biometric recognition system, the type of the biometric data to be used and the system requirements should be considered seriously.

Any biometric system should satisfy some requirements as mentioned in [3, 6, 7]. While designing a biometric system, apart from the type of the characteristics, the privacy of the biometric data should be deliberated. The biometric trait should be kept secret and used only for the intended purpose. In addition, it should be easy to use and should not bother people while capturing the biometric data. In terms of accuracy, the system should accept all authorized people, while rejecting all impostors even in large databases. Besides, the timing and memory costs should be small. The queries should be processed quickly without causing delay in real-time applications. The templates should have acceptable sizes even in very large databases of millions of people.

2.2 Cryptography

People need privacy, security and trust in all aspects of life. Especially, in digital world, the number of transactions, interactions and the size of personal data are growing very fast. At the same time, the need for privacy, security and trust is becoming more and more important. Not only today, even thousands of years ago, emperors had to establish secure communication channels between themselves, governmental officials and commandants for success. Moreover, they had to protect their sensitive, secret and strategic messages against the enemies. In order to prevent their messages to be read by adversaries, the cipherment techniques were developed, such that only the authorized receivers can read those messages [8]. Nearly 4000 years ago, in ancient Egypt, the recorded history of cryptology was started [9]. Symbol substitutions were found on the remains of the hieroglyphs. Additionally, Julius Caesar was one of the emperors who had used a cipher method which has been named after him [10]. While the national officers were trying to find out the best methods to hide their messages, the cryptanalysts were trying to break the ciphers. If a code was broken, the officers had to find a new and stronger cryptographic technique to hide their messages. Although the communicating entities have changed a little nowadays, this progress is still almost the same and conduces to the advances in cryptography. It was an art in ancient times; however, now it is a science [11].

In a general cryptographic model, the building blocks are as follows: plaintext, encryption algorithm, secret key, ciphertext and decryption algorithm [12, 13]. *Plaintext* is the original message in clear text. *Encryption algorithm* is mostly a mathematical function that transforms plaintext into unintelligible form. Plaintext and the secret key are the inputs of the encryption algorithm. *Secret key* is a value, using which the encryption algorithm performs the transformations. Secret key is selected independently of the plaintext and the encryption algorithm. *Ciphertext* is the output of the encryption algorithm. It is a random looking, unintelligible data which is produced depending on the plaintext and the key. Even two exactly the same plaintexts produce different ciphertexts with different keys. Besides, two different plaintexts are converted into two different ciphertexts even with the same keys. *Decryption algorithm* is the reverse of

the encryption algorithm. Its task is the regeneration of the original plaintext using the ciphertext and the key. In Figure 2, the general flow of a secure communication is visualized. In this figure, there are two subjects, Alice and Bob, who want to communicate with each other. Their personal messages should be protected against unauthorized accesses. Therefore, Alice uses an encryption algorithm that transforms her plaintext into the ciphertext. Since the channel is not secure, interceptors and eavesdroppers can capture the ciphertext, but the content does not make sense to them without the secret key. At the receiver side, Bob decrypts the ciphertext using the secret key and can read the original message.

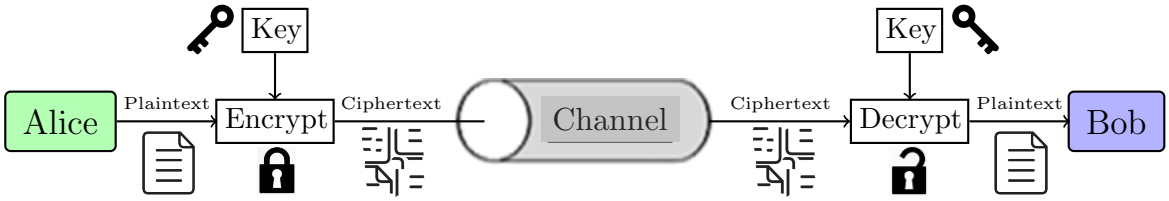

Figure 2: General flow of a secure communication

Cryptosystems should provide many functionalities, such as confidentiality, authentication, integrity and non-repudiation [12, 13]. *Confidentiality* means hiding the content of the message transmitted. It protects the message from unauthorized access. This functionality is ensured by encrypting the data. *Authentication* means that the owner of the message is the correct entity. The authorized entity is the one that is verified and trusted. In addition, by the *integrity*, it is provided that the messages cannot be modified by unauthorized parties. It is for sure that the received message is as exactly the same as the sent message; no modification, no insertion, no deletion. *Non-repudiation* provides that none of the communicating entities can deny any of the transactions they made.

Cryptographic systems are classified according to three different characteristics: (i) the transformation method used in the encryption algorithm, (ii) the way in which the transformation is applied, and (iii) the number of keys [12]. The transformation method could be substitution and/or transposition. In the *substitution* method, each element of the plaintext is mapped into a different element. On the other hand, in the

transposition method, the position of each element in the plaintext is changed. These methods can be used individually or together in an encryption algorithm. The important issue which must be considered here is that the transformation must be reversible without any loss of information. Moreover, the plaintext elements can be transformed into the ciphertext elements either one by one or block by block. If they are processed one by one, this system is called as a *stream cipher*. However, if the elements are processed as blocks, then the system is a *block cipher*. In terms of the number of keys, there are single or two keyed systems. In a single keyed system, the key is known by both the sender and the receiver, and the keys are identical in both ends of the communication. This type of systems is named as *symmetric key encryptions*. In a two keyed system, encryption and decryption are done using different keys. They are referred to as *asymmetric key encryptions*. Since our work provides that the sender and the receiver agree on a private key, which can further be used in a symmetric key encryption algorithm, only the symmetric key cryptography is explained in the following subsection. After that, the mechanisms and examples of hash functions are discussed in the next subsection, since the work proposed in this thesis makes use of the hash functions.

2.2.1 Symmetric Key Cryptography

All of the classical cryptosystems were symmetric [10]. In symmetric key cryptography, encryption and decryption keys are the same and known to both of the communicating entities. Therefore, symmetric key cryptography is also referred to as *private key*, *single key* or *secret key cryptography*. It is also named as *conventional encryption*. One of the most important hypotheses in cryptography, which is known as *Kerckhoffs's principle*, was stated by Auguste Kerckhoffs in 1883 [10]. He stated that the security of a cryptosystem is not built upon the secrecy of the encryption methods being used; only the key is the secret. Due to the fact that the encryption/decryption algorithms are public and the keys are the same, in a symmetric key cryptography setting, the security of the system depends only on the strength of the encryption algorithm and the secrecy of the symmetric keys. The encryption algorithm should be strong enough not to allow

an attacker to figure out the secret key, even if the attacker has access to many ciphertexts and their corresponding plaintexts. The possibility of having public algorithms for the symmetric key cryptography makes it practical to be used commonly [12].

The communication flow which is shown in Figure 2 exemplifies a usage of private key cryptographic algorithm, because the keys are identical. However, the main problem in such systems is the randomness (unpredictability) of the utilized keys and the key management. Actually, the most fundamental problem in cryptography is the secure key exchange and maintenance, as explained in [14]. The details, weak and strong aspects of generic key management techniques are explained in the following paragraph.

Key Management in Symmetric Key Cryptography In an automated world, the communicating entities should agree on the same symmetric keys without physically being together and without compromising future communication. In addition, the keys should be updated frequently, in order to limit the compromised information, if a key is learned by an attacker [13]. Key management refers to the establishment, distribution or agreement of the secret keys [12, 14]. *Key establishment* is a process at the end of which a secret is shared among the communicating entities. A key establishment process can be a distribution or an agreement. In a *key distribution* scheme, one of the entities selects a secret key, and transmits this key to the other entity in a secure way. If the key is generated at the sender side, it should be sent to the destination through a secure channel. If this is not the case, a third party can generate and deliver the key to both of the communicating parties. However, assuming a secure channel between the parties is not realistic in digital world. On the other hand, *key agreement* is a mechanism in which a shared secret is derived by the both ends of the communication as a result of a function or a sequence of steps taken in a protocol. This kind of a protocol should be carefully designed not to leak information to the attacker. In a key agreement process, none of the entities know the key before the process starts. To sum up, a secure and efficient key management is the most crucial aspect of the symmetric key cryptography.

This thesis proposes a secure key agreement protocol between the server and the user. In this protocol, the communicating parties agree on the common parts of a biometric data and generate the secret key using *only* those shared biometric features.

2.2.2 Hash Functions

A *hash function* is defined as a function whose input is an arbitrary-length message and whose output is a fixed length digest [12]. Digest can be referred to as *hash code*, *hash value* or shortly *hash*. The abbreviation of the hash functions $h = H(M)$ means that the hash function H is applied to the message M and the output h is produced. Basically, applying a hash function to a message is like applying a compression function to it several times.

A hash function has to satisfy some requirements which are listed in [10, 12–14]. Firstly, applying the hash function to a large set of inputs should produce uniformly distributed random outputs. Secondly, a single bit change in the input should result in the change of the output completely. Additionally, a cryptographic hash function which is used in security applications should have some more features. A cryptographic hash function must have the one-way property. In other words, calculation of a digest should be easy and quick, whereas it must be computationally infeasible to revert back from the hash value to the original message. Further, a cryptographic hash function must be collision-free. In a weak collision resistance, given a message M, it must be computationally infeasible to find a different message M' whose hash values are the same. In terms of a strong collision resistance, it must be computationally infeasible to find any two different messages whose hash values are the same. Due to these properties, hash functions are mostly used in applications in which the data integrity should be guaranteed. Additionally, hash functions are used for confirmation of knowledge, key derivation and pseudorandom number generation [14]. In the following paragraphs, two hash methods which are used in this work will be explained.

Secure Hash Algorithm (SHA) SHA was proposed by National Institute of Standards and Technology (NIST) in 1993 [10]. It is a function of modular arithmetic and logical binary operations. This version is known as SHA-0. After some weaknesses were found in this algorithm, a new standard, SHA-1, was proposed in 1995 [15]. However, this was not the last. NIST produced a new version of the standard which defines three new versions of SHA [16]. These three versions are known as SHA-2 family of hash functions. In 2005, NIST requested to terminate approval of SHA-1 until 2010. SHA-2 is a standardized hash algorithm since 2005 [12].

One version of SHA-2 functions is SHA-256, and this version is used in this work as the specific hash function. SHA-256 is a hash function which produces 256 bit outputs (hash values). The security of the SHA-2 family functions is proved in [17].

Keyed Hash-based Message Authentication Code (HMAC) Hash-based message authentication codes (MAC) were used extensively with the symmetric encryption techniques. However, cryptographic hash functions run generally faster than the symmetric encryption algorithms [12]. Therefore, keyed hash functions were developed to satisfy the need of a fast hash-based MAC. The mostly known and supported algorithm of such is the HMAC [18]. It is a keyed hash function which is generally used between two parties, if they share a secret key and want to ensure the data authentication and integrity. Any hash function can be embedded to the HMAC algorithm. In this work, we use SHA-256-based HMAC implementation.

2.3 Combining Biometrics and Cryptography: Bio-cryptography

In order to establish trust in a communication, we need cryptographic primitives. On the other hand, we need biometrics to link the user with the application for authentication. For these reasons, researchers combine biometrics with cryptography as this combination provides effective and complementary solutions to data security from different aspects. The combination of cryptography and biometrics is named as *bio-cryptography* or *crypto-biometrics*.

Since the world becomes more and more automated and digital, remote and automatic recognition systems are needed in many areas such as banking, government, military, healthcare and security agencies. Although smart cards, passwords and PINs (Personal Identification Number) are used for the aim of automatic recognition and authentication, this kind of a system can be fooled easily. Some passwords are very simple and can be guessed or broken. Besides, complex passwords are hard to remember, and in this case people tend to note down their passwords. This situation also creates a security threat [4, 5]. Biometric recognition systems solve the problem of automated personal identification; they are more difficult to cheat and they simplify the recognition process [3–5]. In addition, due to the fact that biometrics cannot be lost, copied and shared, a biometric recognition system is more reliable than a password-based system [6]. In terms of security, the biometric authentication systems are equivalent to using a long password, whereas it is as simple as a short password system [19].

Bio-cryptographic approaches are mostly related to the key management. In other words, researchers propose methods to combine biometrics with cryptographic keys. As it is addressed in [20, 21], the main difficulty of using a biometric in a key management process is the noisy nature of the biometrics. In a general biometric system, a partial match above a threshold is acceptable. However, in a cryptographic system, the keys must be identical. Therefore, the researches who are willing to work on this area must design methods to handle this difficulty. There are many works in the literature which are related to the key management using irises, fingerprints, keystroke dynamics and voices. For example, in [22], the authors describe a method to generate a cryptographic key from the iris biometrics. This system uses the passwords as well, but with the help of biometrics, the entropy of the keys increases significantly. [23] introduces a fingerprint based key locking technique. In this technique, the authors locks a random key into the phase information retrieved from the fingerprint image. In addition, in [24], the authors propose a method to harden passwords using the keystroke dynamics of each particular user. A scheme of cryptographic key generation from voice while the user is speaking a password is presented in [25].

While designing a bio-cryptographic application, there are number of constraints that should be taken into consideration. Firstly, the error rates of the system should be reasonable. Secondly, the system should deal with the irrevocability of the biometrics, because a compromised biometric cannot be changed. In addition, since the biometrics is not time varying, template protection and the key diversity should be accomplished. Finally, the length and the randomness of the keys are also very important issues in a secure system.

3 Related Work and Problem Statement

In this section, related work in the literature is discussed and the problem statement of this work is also mentioned.

3.1 Related Work

Bio-cryptographic systems are threefold: (i) key release, (ii) key generation and (iii) key binding. In [4, 5] and [26], the authors describe these bio-cryptographic methods and discuss their problems. In key release mechanism, firstly a biometric authentication process is run. In this process, the input biometric template is compared with the one in the system database. If the matching is successful, a key is released. However, the fact that the user authentication and key release are independent processes is a disadvantage of this mechanism. The key and the user biometric are not strictly bound to each other. On the other hand, in key generation or key binding mechanisms, biometrics and cryptography are integrated; a cryptographic key is bound to the biometric data of the user. In these methods, neither the biometric template, nor the cryptographic key is accessible to the attacker. The correct cryptographic key could only be generated when a valid biometric template is presented by the user. Biometric matching is not performed, because when the correct key cannot be generated from the biometric template, the decryption function fails and the user is rejected automatically. The reason behind this is that cryptographic encryption/decryption functions require exactly the same key.

The main issue in biometric key generation or key binding methods is the variance of biometric template. Due to variations in biometrics, if one bit of the generated key is different than that of the correct one, the genuine user may be rejected. In order

to avoid these false rejects, fuzzy key binding methods are proposed, namely *fuzzy commitment* and *fuzzy vault*. The *fuzzy commitment* scheme was proposed by Juels and Wattenberg [27]. In this scheme, the user selects a secret word W. The difference (XOR) between the user's biometric template X and the codeword W is denoted as d. The difference vector d and the hash of the secret word $y = H(W)$ together constitute the encrypted message. At the verification stage, the user provides a query biometric Y. The difference between Y and d is $W^{'}$, and y is used to check the correctness of the extracted $W^{'}$. This scheme is applicable to be used with ordered set of features, such as iris code. In [28], the authors propose a mechanism to obtain cryptographic keys from iris codes using fuzzy commitment, which is based on the method described in [21]. However, [29] and [30] show that the fuzzy commitment methods are vulnerable, because the attacker can reconstruct not only the key but also the iris code by making use of the error correction codes and statistical attacks.

The *fuzzy vault* scheme is proposed by Juels and Sudan [1]. In contrast to the fuzzy commitment scheme, the fuzzy vault scheme is applied to the unordered set of features, such as minutiae in fingerprints. In this method, a secret word W is mapped to the coefficients of a polynomial $P(x)$. This polynomial is evaluated on the feature points of the biometric template. In addition to the evaluated points $(x, P(x))$, a large number of chaff points that do not lie on the polynomial are generated. These genuine and chaff points are mixed and the total set is named as the *vault*. In the verification stage, the user provides a query template, and with the help of this template, genuine points are determined. Using Lagrange Interpolation, the polynomial $P(x)$ is reconstructed and its coefficients are mapped to the secret $W^{'}$. With the use of error correction codes, the correct secret W is obtained. In [31] and [32], fingerprint-based fuzzy vault methods are presented. However, it is proven that the fuzzy vault is also vulnerable to some known attacks; such as brute-force [33], stolen-key inversion [34] and correlation based attacks [35]. Although in [36], the authors improve fuzzy vault for fingerprint verification, they still leak some information to the attacker by inserting chaff points into the vault that are close to each other but away from the genuine points. With this strategy, the attacker can make sure that if there are two points which are close

enough to each other and if it is known that one of these points is chaff, the other point is definitely a chaff point.

3.2 Problem Statement

The existing key release, key generation and key binding methods do not address the issue of key establishment purely from the biometrics without using random data. In both of the fuzzy commitment and the fuzzy vault mechanisms, the secret word is selected by the user or it is randomly generated. In other words, the secret word is not derived directly from the biometric template. On the contrary, the biometric template is used as a component to hide the secret word. In contrast to all of these methods, in this thesis, the secret word (key) is directly generated from the feature points of the biometric template. In addition, from a timely invariant biometric, generating many different keys is a challenging task which is also achieved in this work.

4 Proposed Secure Key Agreement Protocol using Pure Biometrics

In this section we describe our proposed secure key agreement protocol, which uses fingerprint biometrics without any other type of helper data. The definitions of the symbols used in the protocol definition are given in Table 1. Our key agreement protocol can be divided into two phases: (i) enrollment and (ii) verification, each of which is explained in the following subsections.

4.1 Enrollment Phase

The enrollment is performed only at the server side and the corresponding template generation algorithm is given in Algorithm 1. At the enrollment stage, the user provides three fingerprint images, FP_1, FP_2, FP_3, of the same finger. Then, the minutiae of these fingerprints are extracted. Each minutia is represented with three attributes: x-coordinate, y-coordinate and *type*. The type of a minutia can be *end* or *bifurcation*. *End* type of a minutia indicates a ridge ending. On the other hand, if the ridge branches into two, the branching point is a *bifurcation* type of a minutia. The minutiae list of a fingerprint image constitutes the template of this particular fingerprint image. While generating the template, we quantize the minutiae by selecting representatives from the groups that are determined by the predefined distance threshold, T_{dist}. In this quantization step, the minutiae which are at most T_{dist}-away to any other minutia with the same type are mapped to one minutia by picking the one with the smallest y-coordinate value. After that, the server puts these fingerprint templates on top of each other, in order to find out the most reliable minutiae. The minutiae which are present in

at least two out of three fingerprint templates are considered as reliable minutiae. Only the reliable minutiae are kept in the final template. For the reason that the minutiae are close to each other more than T_{dist} are considered as one minutia, a T_{dist}-*neighborhood* relation is defined as follows: All of the points in the coordinate system which have x-coordinate in $[x_j - T_{dist},\ x_j + T_{dist}]$ and y-coordinate in $[y_j - T_{dist},\ y_j + T_{dist}]$ are the neighbors of the minutia with $(x_j,\ y_j)$ in the T_{dist}-*neighborhood*. This neighborhood relation is exemplified in Figure 3, where the original minutia is located at (49, 91) and T_{dist} is 2.

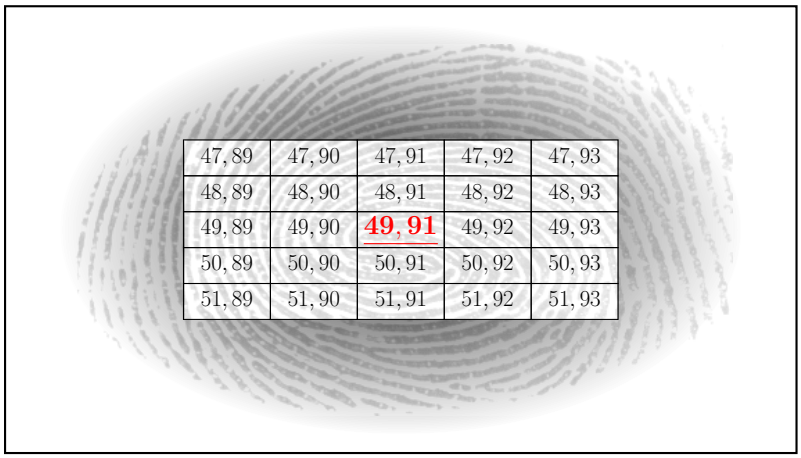

Figure 3: Neighborhood relation when $T_{dist} = 2$

Thereafter, x-coordinate, y-coordinate and *type* of each minutia point and its neighbors in T_{dist}-neighborhood together with the *type* of this particular minutia are concatenated and hashed one by one as $H^1(x||y||type)$. These hashes constitute a particular user's template in the server. Note that although double hashes will be needed in the verification stage, storing single hashes is enough, since the double hashes can be calculated by re-hashing the stored values once again if necessary.

Table 1: Symbols used in Protocol Definition

Symbol		***Description***
FP		Fingerprint
x		x-coordinate of a minutia
y		y-coordinate of a minutia
$type$		Type of a minutia
n_u		Total number of genuine minutiae on the user side
n_s		Total number of genuine minutiae on the server side
n_{com}		Number of common minutiae found by the server
n_{com}^{key}		Number of minutiae used in the final key agreement
$H^i(\cdot)$		Hash function applied i times ($i \geq 1$)
G_s		Set of genuine minutiae on the server side
G_u		Set of genuine minutiae on the user side
C		Set of fake minutiae on the user side
Q_u		Set of shuffled $(H^2(g_u) \cup H^2(c))$ s.t. $g_u \in G_u \,\&\, c \in C$
G'_s		Set of minutiae $\in \{Q_u \cap G_s\}$
$G''_{s,j}$		Any subset of G'_s s.t. $\lvert G''_{s,j}\rvert = \lvert G'_s\rvert - j \;\; (j \geq 1)$
G'_u		Any subset of G_u s.t. $\lvert G'_u\rvert = \lvert G'_s\rvert$
$G''_{u,j}$		Any subset of G'_u s.t. $\lvert G''_{u,j}\rvert = \lvert G'_s\rvert - j \;\; (j \geq 1)$
S		Similarity score
T_{sim}		Acceptance similarity score threshold
T_{dist}		Distance threshold used in neighborhood definition
$K^i_{(us,su)}$	K^i	i^{th} key generated ($i \geq 0$)
	us	by the user to communicate with the server
	su	by the server to communicate with the user
$HMAC(\cdot)$		Keyed Hash-based Message Authentication Code (HMAC) [18]
$HMAC_{K_{us}}(\cdot)$		$HMAC$ generated using K_{us}
$HMAC_{K_{su}}(\cdot)$		$HMAC$ generated using K_{su}
$HMAC_{K^i_{su}}(\cdot)$		$HMAC$ generated using K^i_{su}
att_c		Attack complexity

Algorithm 1 Template Generation Algorithm

INPUT: FP_1, FP_2, FP_3
OUTPUT: G_s

$G_s^{init} = ExtractMinutiae(FP_1, FP_2, FP_3)$
for $i = 1 : |G_s^{init}| - 1$ **do**
  $m_1 = G_s^{init}(i)$
  for $j = i + 1 : |G_s^{init}|$ **do**
    $m_2 = G_s^{init}(j)$
    if $m_1.x \geq m_2.x - (2 * T_{dist})$ &
     $m_1.x \leq m_2.x + (2 * T_{dist})$ &
     $m_1.y \geq m_2.y - (2 * T_{dist})$ &
     $m_1.y \leq m_2.y + (2 * T_{dist})$ &
     $m_1.type == m_2.type$ **then**
      $m_1.visited++$
    end if
  end for
end for
for $i = 1 : |G_s^{init}|$ **do**
  if $G_s^{init}(i).visited < 2$ **then**
    Remove i^{th} *minutia from* G_s^{init}
  end if
end for
$ind \leftarrow 1$
for $i = 1 : |G_s^{init}|$ **do**
  $m_1 = G_s^{init}(i)$
  for $j = (-1) * T_{dist} : T_{dist}$ **do**
    for $k = (-1) * T_{dist} : T_{dist}$ **do**
      $G_s(ind) = H^1(m_1.x + j||m_1.y + k||m_1.type)$
      $ind \leftarrow ind + 1$
    end for
  end for
end for

4.2 Verification Phase

At the verification stage, three different fingerprint images of the same finger are used. As in the enrollment phase, the minutiae points are extracted from these fingerprints. Similarly, at most T_{dist}-away minutiae are mapped to one minutia by selecting the one with the smallest y-coordinate value. After that, three fingerprint templates are put on top of each other and the most reliable minutiae are selected. In order to mask the genuine minutiae points at the user side, $10 \times |G_u|$ fake minutiae points are

generated randomly. Fake minutiae point generation is an important process, since the fake points should not leak any information to the attacker. For this reason, a fake point must be indistinguishable from a genuine minutia point from an attacker's point of view. Since we make sure that all of the genuine minutiae points are at least T_{dist}-away from each other, fake minutiae points must be T_{dist}-away from all the other points as well. Therefore, the fake minutiae points must also preserve the T_{dist}-neighborhood relation.

After the fake minutiae point generation process ends, each minutia point's (genuine and fake) x-coordinate, y-coordinate and *type* are concatenated. Each value is double hashed as follows: $H^2(x||y||type)$. As the key will be generated using single hashed values of the genuine minutiae, the user keeps $H^1(x||y||type)$ only for the genuine minutiae. Note that in contrast to the enrollment phase, the points in the T_{dist}-neighborhood are neither hashed nor sent to the server.

The protocol flow can be seen in Figure 4. Double hashed points' list together with the ID of the user is transmitted to the server. In order to extract the genuine points from the list, the server compares each point with this particular user's double hashed template. Since the server has the neighbor minutiae points as well, if a genuine minutia of the user is in the T_{dist}-neighborhood with a minutia in the server side, it is counted as a common genuine minutia. However, it may or may not be a genuine minutia. Our protocol provides solutions in the following steps for the cases that fake minutiae are considered as genuine minutiae.

After the comparison is completed, a similarity score is calculated. There are two well-known methods to calculate the similarity score of two fingerprints as given in Equation 1 and Equation 2 [37], where n_{com} is the number of common minutiae, n_u is the number of genuine minutiae on the user side, n_s is the number of genuine minutiae on the server side.

$$S = \frac{n_{com}^2}{n_u \times n_s} \times 100 \tag{1}$$

$$S = \frac{2 \times n_{com}}{n_u + n_s} \times 100 \tag{2}$$

If the calculated score is above a certain acceptance threshold, T_{sim}, the user is accepted and the key agreement process starts. In the key agreement process, the server concatenates single hashes of all common minutiae, $H^1(g'_{s,i})$, and everything is rehashed to generate K_{su}, which is the key to be used while communicating with the user. In order to make sure that the user will generate the same key, the server computes the HMAC of a predefined message msg using K_{su} and transmits this value together with the number of common found minutiae, $|G'_s|$, to the user.

Upon receiving the message, the user generates a key using one of the possible subsets of the genuine minutiae whose size is the same as the number of found minutiae on the server side. If the user can verify the HMAC using this generated key, (s)he sends a positive acknowledgment to the server. Otherwise, the user generates another key using another subset, until either the HMAC is verified or all possible subsets are exhausted. In the case that the HMAC is not verified, *RETRY* message is transmitted to the server.

If the protocol continues with the *RETRY* message, the server computes the similarity score using $|G'_s| - 1$ as the number of common minutiae. If the score is above the acceptance threshold T_{sim}, the server generates all possible keys using all possible subsets of the found minutiae, whose size is equal to $|G'_s| - 1$. The server then transmits all of the HMAC values generated using these keys to the user. If the user can verify any one of these HMAC values using any one of the keys generated with any possible subset of the genuine minutiae, whose size is equal to $|G'_s| - 1$, the user transmits a positive acknowledgment and the index, i, of the verified HMAC to the server. Otherwise, another *RETRY* message is transmitted to the server. In this case, the same process with $|G'_s| - 2$ is carried out. The protocol stops at the j^{th} step, if either the similarity score computed using the number $|G'_s| - j$ is less than the acceptance threshold, or any HMAC value is verified by the user. If any HMAC value is verified at the end of this protocol, the server and the user can agree on a symmetric cryptographic key without using any non-biometric or random value. On the other hand, if the protocol stops without generating a symmetric key, it can start from scratch upon request.

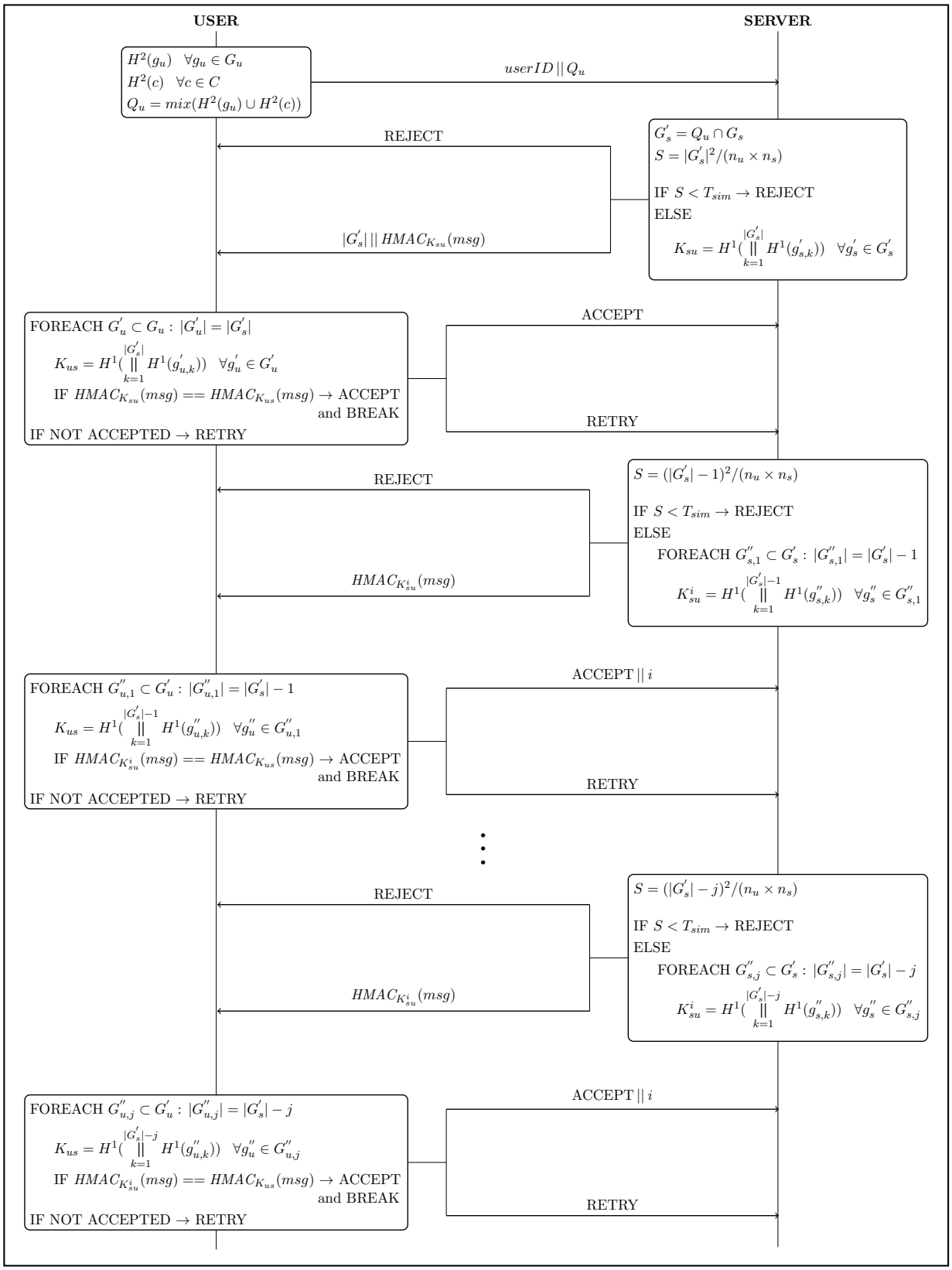

Figure 4: Our proposed secure key agreement protocol

5 Performance Evaluation

Our proposed protocol is tested with two different datasets. First dataset consists of 30 subjects from Verifinger Sample Database [38], which includes fingerprints scanned using Cross Match Verifier 300 at 500 ppi [39]. In this dataset, each subject has 8 fingerprint images of the same finger. Second dataset includes 292 fingerprints, which are scanned using Papilon DS22N at 500 ppi [40]. This dataset is constructed by collecting fingerprints from volunteers in Sabancı University by TÜBİTAK 114E557 project team. In this dataset, we have 10 impressions of each finger. All fingerprint images in both datasets are aligned using their intensity values in MATLAB R2015a. The minutiae of each fingerprint is extracted using the Neurotechnology Biometric SDK 5.0 Verifinger [38]. First 3 fingerprint images are used to generate the template on the server side. For the first dataset, the remaining 5 fingerprint images are used as combinations of 3 at the user side. Hence, each subject is tested $\binom{5}{3} = 10$ times. On the other hand, in the second dataset, there are 7 fingerprints reserved for user side operations, which are employed as combinations of 3. Thus, each subject in this dataset is tested $\binom{7}{3} = 35$ times. In addition to the genuine tests, impostor tests are also carried out. In these impostor tests, each subject's template is tested against all other subjects' queries. The hash function used in the protocol is SHA-256 [16] as mentioned in Section 2.2.2; hence all of the generated keys are 256 bits long.

In the subsections given below, we introduce the performance evaluation metrics, discuss the verification results of the system and provide security analyses of the protocol, as well as representing the randomness and distinctiveness analyses of the generated keys. After that, we analyse the complexity and memory requirements of the system. We conclude the section with comparative analysis of our protocol with fuzzy vault.

5.1 Performance Metrics and Parameters

In biometric authentication systems, the performance is measured using error rates, namely False Accept Rate (FAR) and False Reject Rate (FRR). FAR is the percentage of the impostor subjects who are accepted as genuine users; whereas FRR is the percentage of the genuine subjects who are rejected. In order to evaluate the performance and minimize both FAR and FRR at the same time, their intersection point is considered. This point determines the Equal Error Rate (EER). The lower the EER is, the better is the performance of the biometric authentication.

In terms of randomness, the mostly known and accepted measure is the Shannon's Entropy. This metric was proposed by Claude E. Shannon in 1948 [41]. The entropy metric measures the randomness (unpredictability) of the strings. The entropy value of a string is between [0, 1]. The maximum value, 1, indicates perfect randomness. In this analysis, Shannon's Entropy values of the keys are calculated using Equation 3, in which K_i represents the i^{th} bit of the key and $P(\cdot)$ is the probability function.

$$H = -\sum_{i} P(K_i) log_2 P(K_i) \tag{3}$$

Distinctiveness of the keys are measured with the Hamming distance metric. It was proposed by Richard Hamming in 1950 [42]. Hamming distance is the number of elements which are different at the same positions of two equal length strings. An example of the Hamming distance computation is given in Figure 5. Hamming distance helps to assess how much different is two keys, since we want to establish different keys after each protocol run. The larger Hamming distance values are, the more different are the keys.

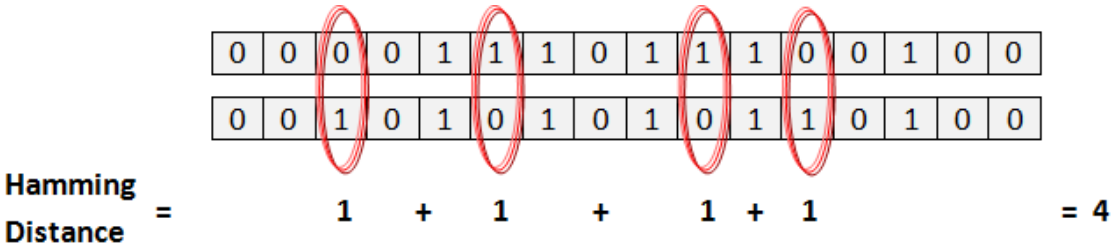

Figure 5: Hamming distance of two binary strings

5.2 Verification Results

For each test, a similarity score is calculated as given in Equation 1 and Equation 2 in Section 4.2. The minimum score, the maximum score and the average score of the system are calculated for each subject. These scores are used as acceptance thresholds of the system one by one. For each different threshold, the corresponding FAR and FRR values of the system are calculated. As a result of these operations, the best EER (the point where FAR = FRR), percentages are obtained when the maximum score of the system is picked as the acceptance threshold.

For the first dataset, Figure 6 shows the FAR and FRR percentages when the maximum of the similarity scores, which are calculated using Equation 2, is picked as the acceptance threshold. As can be seen in this figure, our protocol achieves 0.57% EER when the optimum acceptance threshold is 24.48. If the acceptance threshold is selected as the average score of the system, the EER lies at 6.66% with 13.57 acceptance threshold, as can be seen in Figure 8. On the other hand, if the acceptance threshold is selected as the minimum score of the system as in Figure 10, the EER is 10% and the acceptance threshold is 8.59.

Besides, for the second dataset, when the maximum of the similarity scores is picked as the acceptance threshold, the system has 0.48% EER and the optimum acceptance threshold is 25.49, as can be seen in Figure 7. Figure 9 shows the verification results, when the average score is selected as the acceptance threshold. In this case, the system reaches the EER of 2.61% with 15.06 acceptance threshold. Moreover, when the minimum score is selected as the acceptance threshold as in Figure 11, the EER is 19.4% and the acceptance threshold is 6.13.

In Table 2, all of these EER values that are calculated according to the two different predefined similarity score equations, for both datasets, are listed. The explained results are summarized in this table as well. The EER values, obtained when the maximum score of the system is selected, are quite sufficient for a secure biometric authentication system.

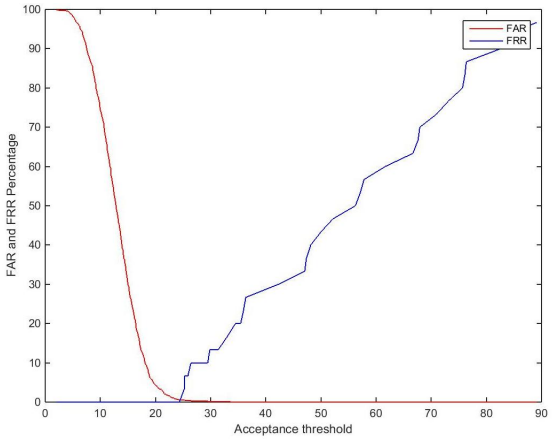

Figure 6: Max scores picked as threshold for the 1^{st} dataset

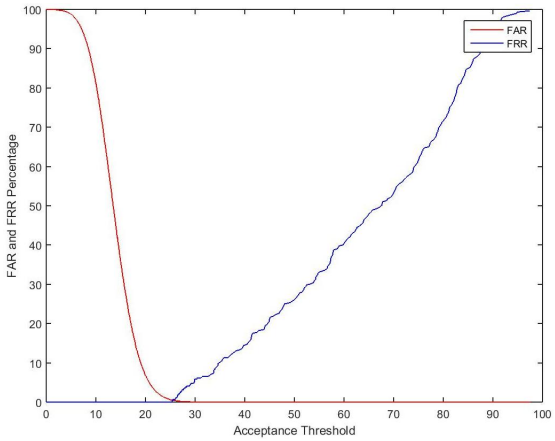

Figure 7: Max scores picked as threshold for the 2^{nd} dataset

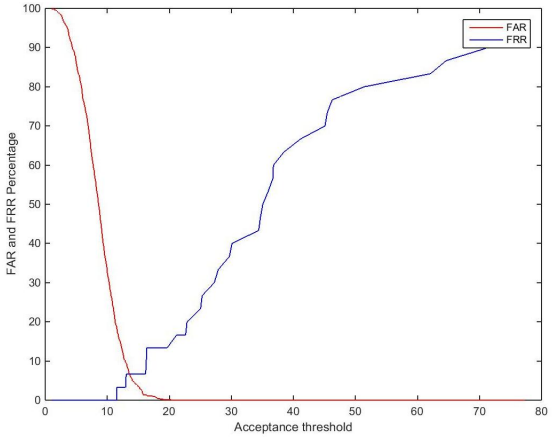

Figure 8: Avg scores picked as threshold for the 1^{st} dataset

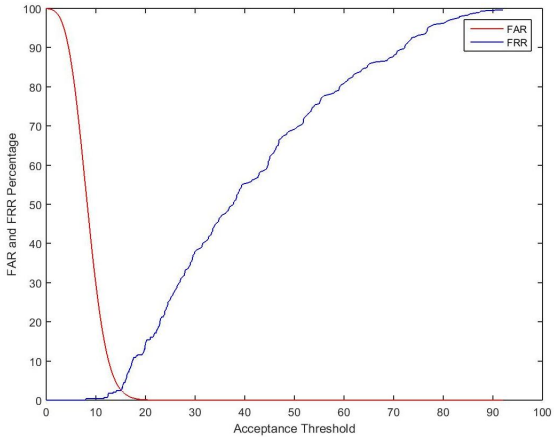

Figure 9: Avg scores picked as threshold for the 2^{nd} dataset

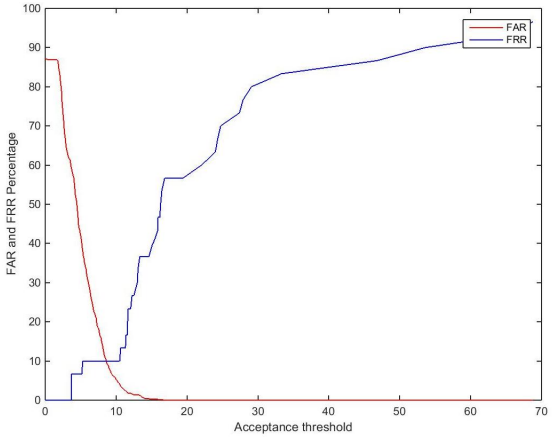

Figure 10: Min scores picked as threshold for the 1^{st} dataset

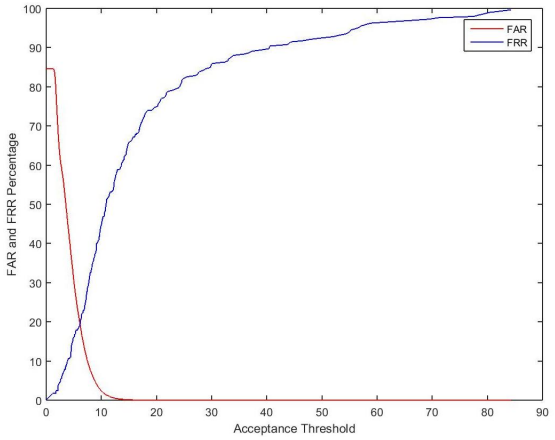

Figure 11: Min scores picked as threshold for the 2^{nd} dataset

Table 2: All EER Values

Equation	***Strategy***	1^{st} ***Dataset EER*** (%)	2^{nd} ***Dataset EER*** (%)
Equation 1	*min*	10	19.86
	max	0.57	0.55
	avg	5	2.60
Equation 2	*min*	10	19.4
	max	0.57	0.48
	avg	6.66	2.61

5.3 Security Analyses

In this section, firstly our threat model is given. After that, the strength of our protocol against brute-force, replay and impersonation attacks are analysed. In addition, the quality of the generated keys are also examined via entropy and Hamming distance analyses.

5.3.1 Threat Model

The attacker's main aim is twofold: (i) to impersonate a genuine user, and (ii) to learn the key between the server and any victim user for eavesdropping purposes. We do not assume a secure channel. Thus, the attacker can obtain all protocol messages including the hash values and HMACs exchanged between the user and the server. Consequently, the attacker learns the number of minutiae points used for the key agreement. The attacker may apply brute-force attack in passive mode by making use of the exchanged messages and try to guess the key. Similarly, in order to impersonate a genuine user, the attacker may apply a replay attack in active mode. However, our protocol resists these type of attacks to some extent as discussed in the upcoming subsections.

5.3.2 Resistance Against Brute-Force Attacks

The attacker can always launch a brute-force attack by trying all possible key combinations. Since the key is 256 bits long, this attack is infeasible. However, in this section, we will give a more intelligent brute-force attack by making use of the protocol messages.

This attack is applied by generating all possible minutiae locations and types. In the first dataset, one fingerprint can have at most 512 x and y values, because of the sizes of the fingerprints in this database. A minutia can have two different types: *end* or *bifurcation*. It means that the attacker must generate $512 \times 512 \times 2 = 2^{19}$ points, and hash them once and twice. In the second dataset, x and y coordinates of a fingerprint are at most 850. Similarly, each minutia can be *end* or *bifurcation*. Thus, the attacker must generate $850 \times 850 \times 2 \doteq 2^{21}$ points, and hash them once and twice. Due to the fact that the user sends the genuine and fake minutiae list, Q_u, to the server, attacker's search space decreases to $|Q_u|$, regardless of the datasets. However, our analysis shows that this brute-force attack is still infeasible as discussed below.

The attacker has the list Q_u and the number of minutiae with which the key is generated, n_{com}^{key}. In order to find the generated key, the attacker should try all possible subsets of the set Q_u with size n_{com}^{key}, yielding the attack complexity att_c (Equation 4).

$$att_c = \binom{|Q_u|}{n_{com}^{key}} = \frac{|Q_u|!}{n_{com}^{key}! \times (|Q_u| - n_{com}^{key})!} \tag{4}$$

For instance, if the user sends a list of 440 points, i.e. $|Q_u| = 440$, in which 40 of them are genuine, i.e. $|G_u| = 40$, and if the key agreement is completed with 16 common minutiae, i.e. $n_{com}^{key} = 16$, then the attack complexity becomes $\binom{440}{16} = \frac{440!}{16! \times (440-16)!} \cong 2^{96}$.

In order to calculate the overall attack complexity of the system, the combination in Equation 4 is calculated after each key agreement. Then, we take the average of the complexity results. The analysis shows that the average attack complexity of the system with the first dataset is 94 bits (i.e. it requires 2^{94} hash and HMAC verifications). On the other hand, for the second dataset, the attack complexity of the system is 118 bits (i.e. it requires 2^{118} hash and HMAC verifications) on the average. As dis-

cussed in [43], even with custom hardware implementation, computation of one block of HMAC-SHA256 takes approximately 0.8977 microseconds. Thus, the above mentioned complexities correspond to 5.6×10^{14} years of attack for the first dataset and 9.46×10^{21} years of attack for the second dataset. As a result, we can conclude that our protocol efficiently resists intelligent brute-force attacks.

5.3.3 Resistance Against Replay Attack and Impersonation

The aim of replay attack is to impersonate a genuine user and get the legitimate key. In order to do this, the attacker replays the previously exchanged messages between the victim user and the server. The attacker needs to know the genuine minutiae points to effectively calculate the generated key; otherwise, (s)he must try all possible combinations out of Q_u. Since the attacker does not know the genuine minutiae points, the complexity of this attack becomes the same as that of the brute-force attack given in Equation 4.

Moreover, the attacker may use his/her own fingerprint instead of the genuine user's fingerprint. The resistance of the protocol against this type of classical impersonation attack is shown to be very low, since the FAR is 0.57% in the first dataset and 0.48% in the second dataset. The readers should also note that such counterfeiting attacks are general problems of all biometrics and related protocols; not specific to ours.

5.3.4 Randomness of the Generated Keys

In the first dataset, we generate 300 keys (30 subjects, 10 keys per subject). The entropy values of these keys are given in Figure 12. Since each key is analysed one by one, key ID in this figure indicates which key has which entropy value. The more the entropy value approaches to 1, the more random the key is. In this case, approximately 84% of the keys have entropy values that are greater than 0.994, and also all of the keys have entropy values that are greater than 0.98, which implies very good randomness. It is important to note that these keys are generated by hashing the common minutiae. It is an expected result to have high entropy for the hash results, because the hash functions kind of randomize the input string. Therefore, the entropy values of the

concatenation of common minutiae are calculated as well. The concatenation is as follows, $x||y||type$. The entropy values of these concatenations are given in Figure 13. Although the entropy values decrease a little, 92.30% of the keys have entropy value above 0.98. Thus, they are still random enough.

In the second dataset, we generate 10220 keys (292 subjects, 35 keys per subject). Figure 14 shows the entropy values of these keys. In the second dataset case, approximately 99% of the keys have entropy values that are greater than 0.98, and also all of the keys have entropy values that are greater than 0.94. These values are sufficient for being a secure and random key. As mentioned before, these keys are generated by hashing the common minutiae and their randomness can be as expected. For this reason, the entropy values of the concatenation of common minutiae, $x||y||type$, are calculated and given in Figure 15. Despite the fact that the entropy values decreases, 90.23% of the keys have entropy value above 0.96. Although there are some entropy values around 0.78, it is normal to have such outliers in big datasets like ours. Hence, their randomness is sufficient.

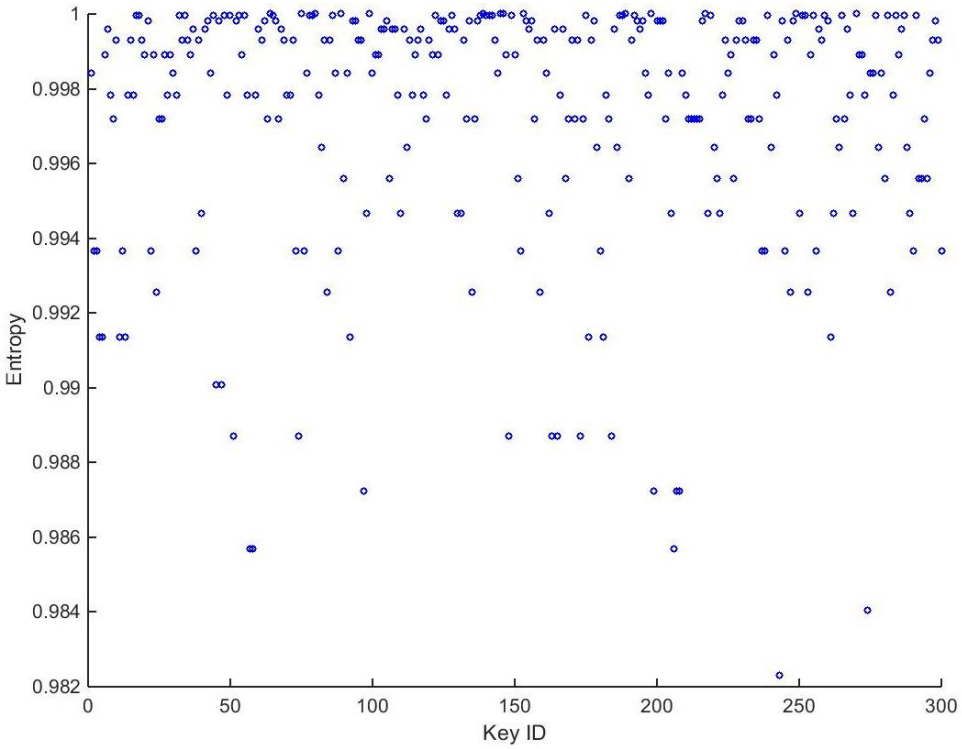

Figure 12: Entropy values of the generated keys for the 1^{st} dataset

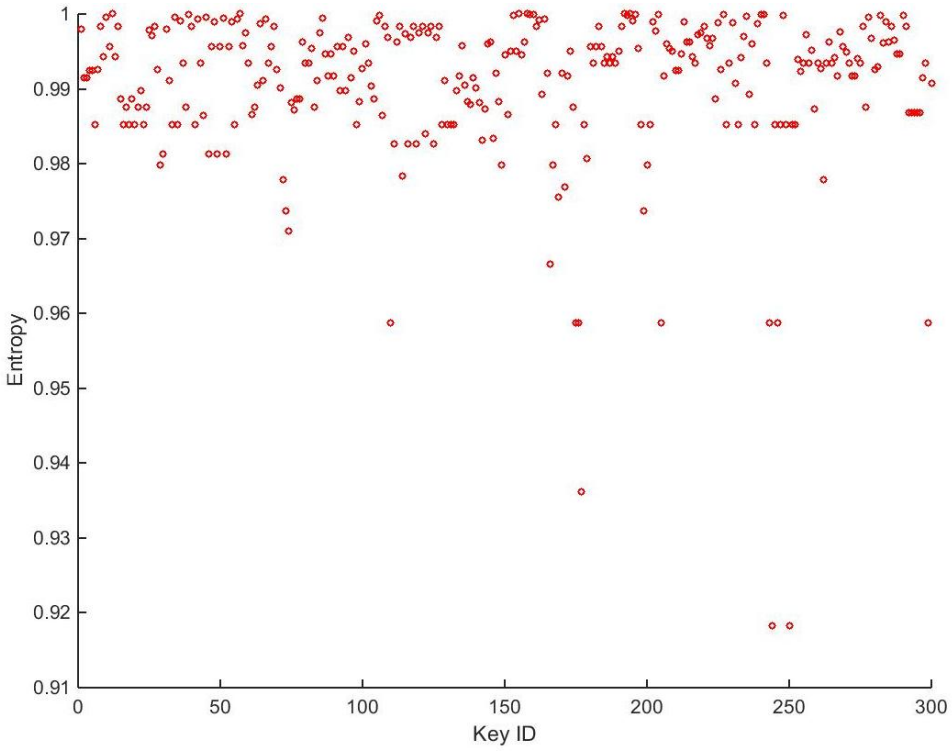

Figure 13: Entropy values of the minutiae concatenations for the 1^{st} dataset

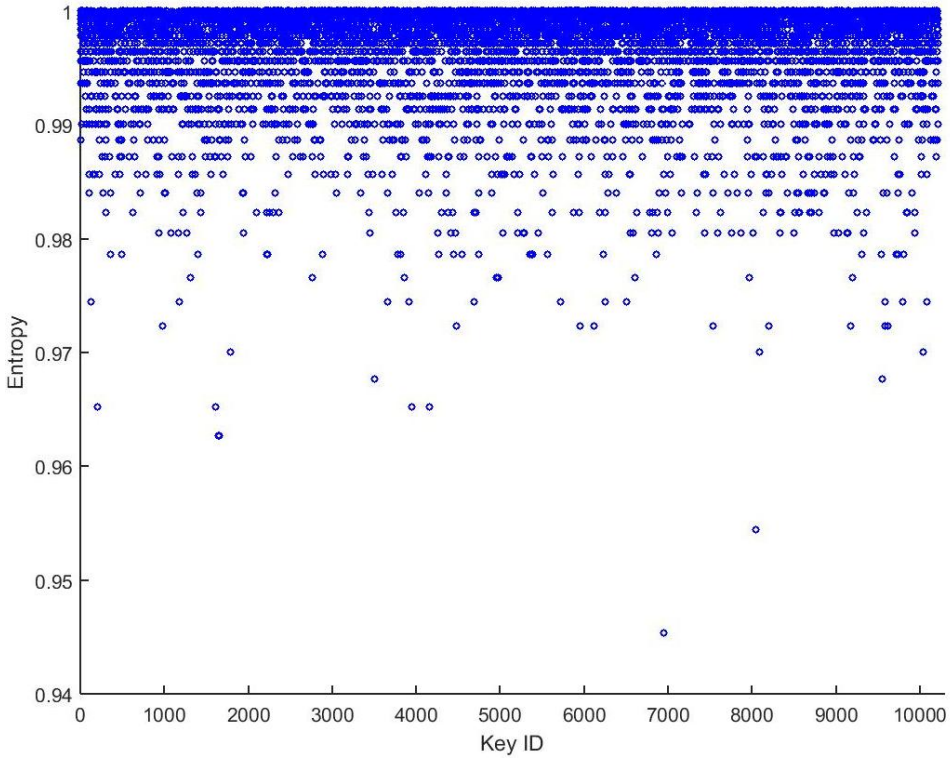

Figure 14: Entropy values of the generated keys for the 2^{nd} dataset

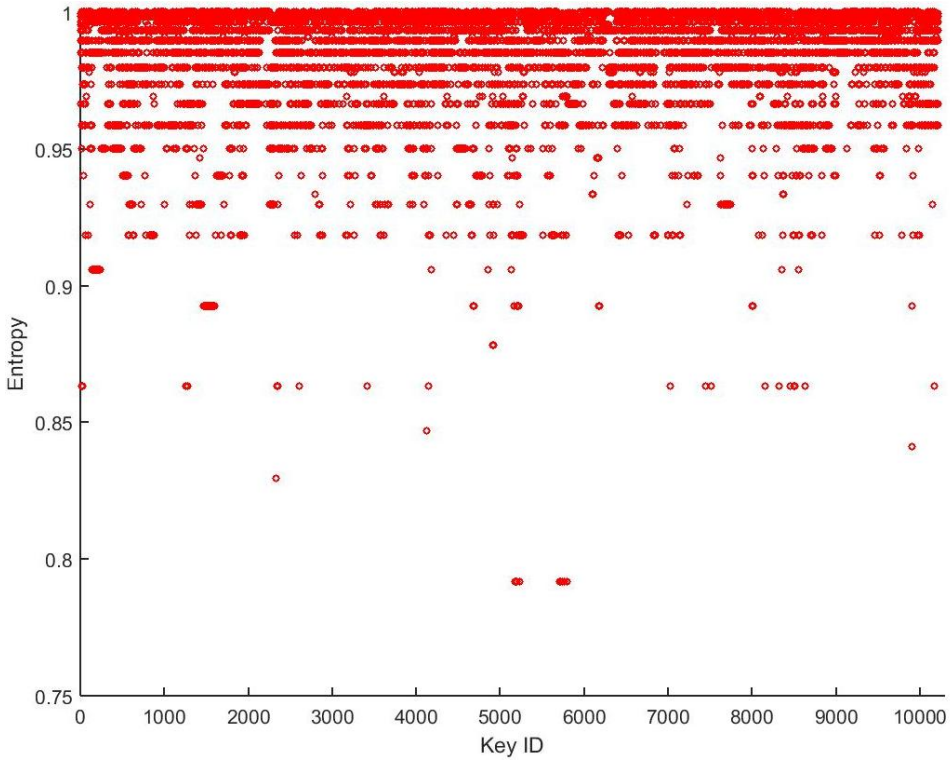

Figure 15: Entropy values of the minutiae concatenations for the 2^{nd} dataset

5.3.5 Distinctiveness of the Generated Keys

The fingerprints are time invariant biometrics. However, generating the same key in each key agreement is undesirable, because compromise of a key should not risk the confidentiality of the messages in other sessions. Thus, it is important to have a different key in each key agreement. The minutiae quality and ordering change according to the fingerprint scanner, pressure of the finger on the scanner, acquisition environment, etc. This situation has both negative and positive effects on the key generation process. The negative effect is the difficulty of agreement on the same key in a particular protocol run. On the other hand, the positive effect is the generation of different keys in each attempt.

In order to measure the difference of the keys for the same user, we calculate the Hamming distances of the keys of the same user obtained after different protocol runs. As can be seen in Figure 16, the average Hamming distances of the keys vary between approximately 120-130 bits out of 256 bits for each user in the first dataset. Also the

minimum difference is 97 bits. On the other hand, Figure 17 shows that the average Hamming distance values of the keys are between 120-130 bits out of 256 bits for each user in the second dataset. The minimum difference is 93 bits. These results show that the users have distinct keys after each key agreement phase.

In addition to the same users' keys comparison, we compare different users' keys to show that they are also different from each other. As can be seen in Figure 18, the average Hamming distances of the keys that are generated for different users of the first dataset vary between approximately 120-135 bits. Also, as can be seen in Figure 19, the average Hamming distances of the different users' keys are between 120-135 bits for the second dataset. These values are very close to the average Hamming distances of the same users' keys. It means that we cannot decide if any two keys belong to the same user or different users by looking at their similarities or differences.

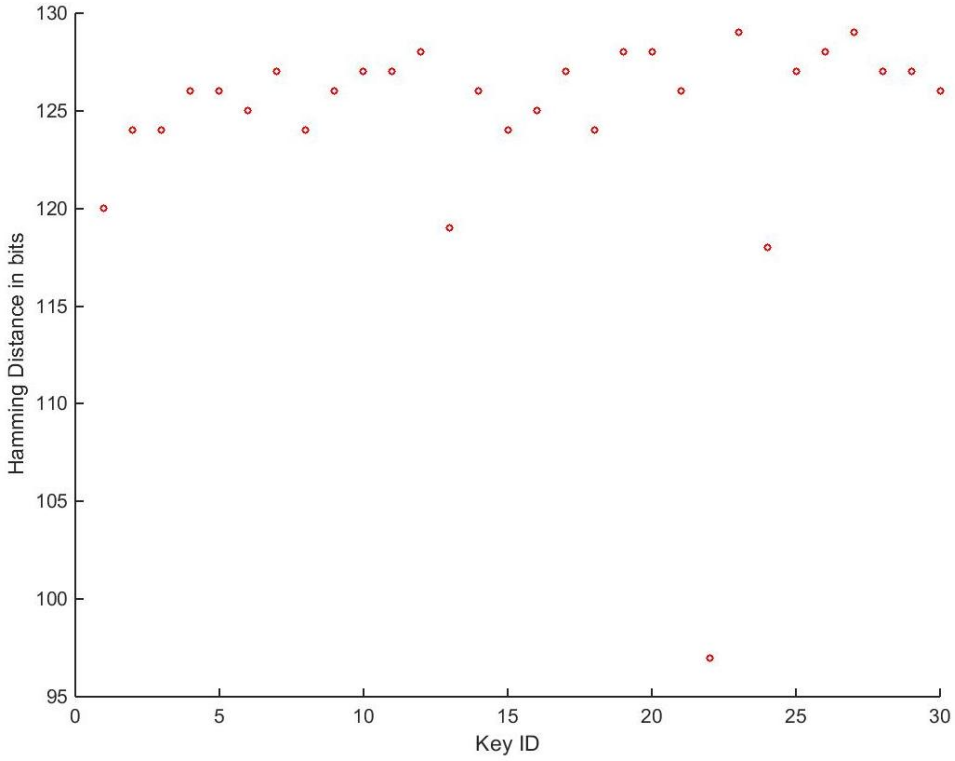

Figure 16: Average Hamming distances of the same users' keys for the 1^{st} dataset

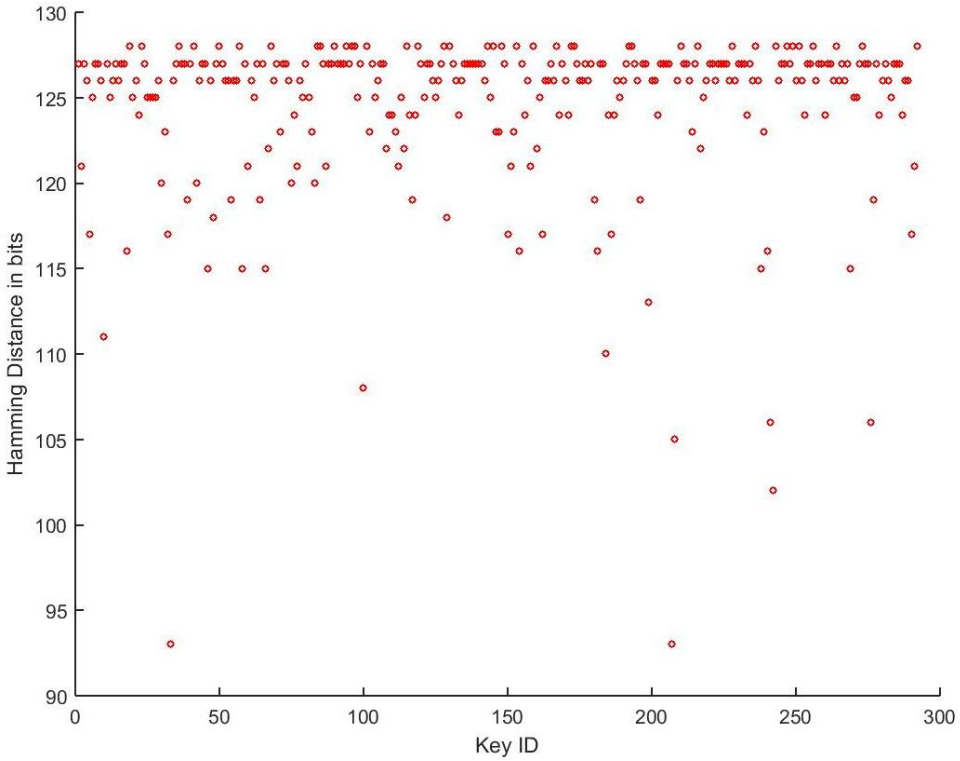

Figure 17: Average Hamming distances of the same users' keys for the 2^{nd} dataset

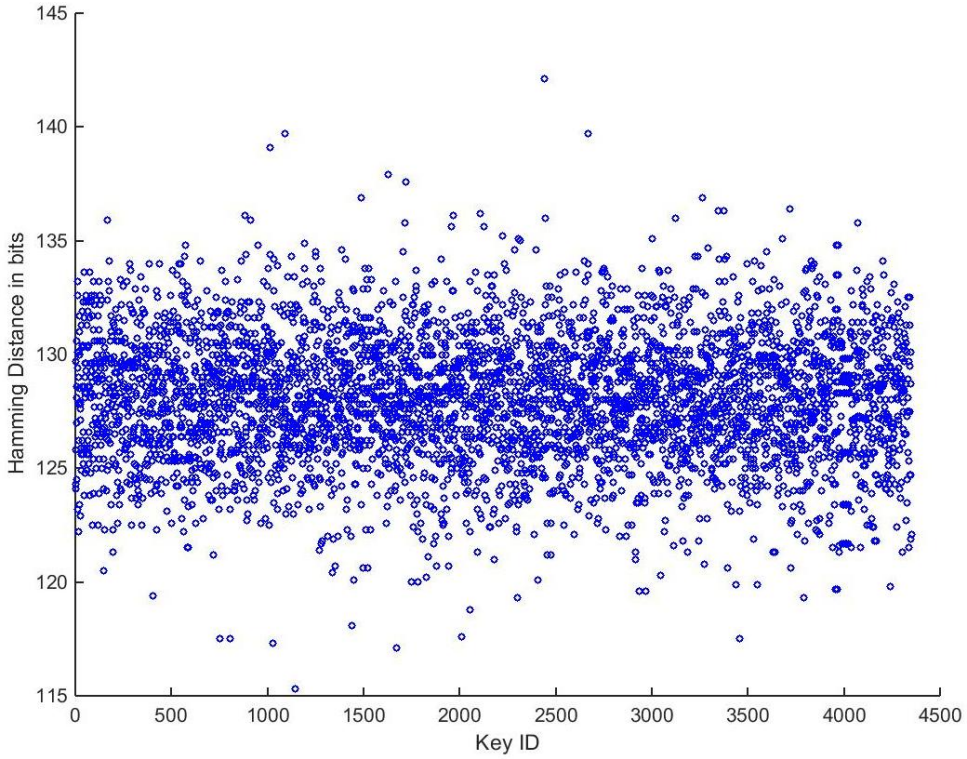

Figure 18: Average Hamming distances of the different users' keys for the 1^{st} dataset

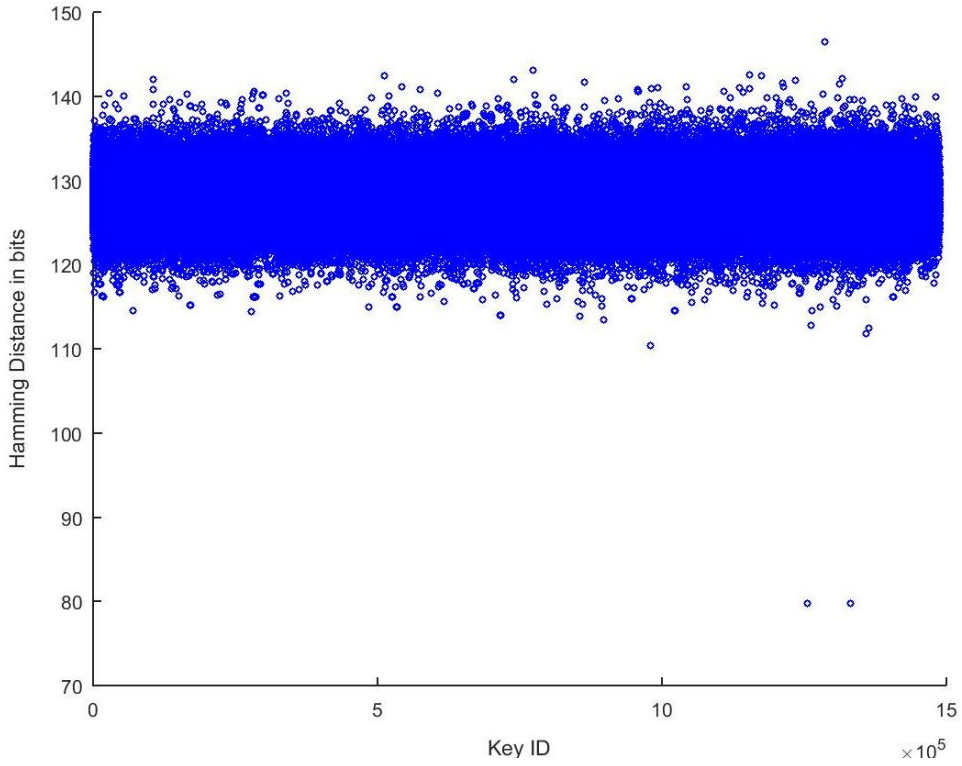

Figure 19: Average Hamming distances of the different users' keys for the 2^{nd} dataset

5.4 Computational Complexity Analysis

Computational complexity of our protocol is calculated by adding up the number of key generation attempts within the protocol run. This complexity is calculated from user and server point of views separately.

The server generates only one key in the first round of the protocol. For the second round of a given protocol run, the amount of key generations is the value of combination of the number of common found minutiae minus one, $n_{com} - 1$, out of the number of common found minutiae in the first round, n_{com}; i.e., $\binom{n_{com}}{n_{com}-1}$. In the upcoming rounds, the server makes computations by selecting one less minutia each time out of n_{com}. Thus, iterations stop when the correct key is generated, which contains n_{com}^{key} minutiae points. The total server complexity, which is formalised in Equation 5, is the sum of the number of all key generation attempts.

$$server_{complexity} = \sum_{i=n_{com}}^{n_{com}^{key}} \binom{n_{com}}{i} \tag{5}$$

The average server complexity, on the other hand, is the average of all attempts taken against all subjects. This average case complexity analysis gives the server complexity as 2^{17} with the first dataset and 2^9 with the second dataset. These values are quite sufficient for a real time application with small response time.

The user complexity for a protocol run is calculated in the following way. Due to the fact that the server sends the number of common found minutiae, n_{com}, to the user in the first round of the protocol, the user tries all possible subsets, whose size is equal to n_{com}, of the genuine minutiae. Therefore, while calculating the complexity, firstly the complexity of generating keys from all possible subsets is analysed. This particular complexity is determined by the number of combinations of n_{com} out of the number of genuine minutiae on the user side, n_u; i.e., $\binom{n_u}{n_{com}}$. If the key cannot be generated in the first round, the user continues the protocol by selecting subsets of genuine minutiae with $n_{com} - 1$ elements out of n_u. If the key is still not generated, the user tries subsets of one less minutia points at each round, until the key is generated or the user is rejected. According to this analysis, the total user complexity is the sum of all key generation attempts from all subsets until the protocol stops. Equation 6 formalises the user complexity.

$$user_{complexity} = \sum_{i=n_{com}}^{n_{com}^{key}} \binom{n_u}{i} \tag{6}$$

The user complexity is calculated for all user tests and then they are averaged. The analysis shows that the average user complexity is 2^{39} in the first dataset and 2^{41} in the second dataset. Although these values seem to be high, it is the cost of not using any helper data. If we ever used any helper data, it might be easier to agree on the common parts of the biometric. However, this would introduce some other problems, such as the distribution of this helper data, risk of compromise and information leakage to the attacker.

5.5 Communication Complexity Analysis

Communication complexity of the protocol is measured according to the number of bits exchanged between the user and the server. The protocol starts with the message sent by the user to the server. This message contains the user ID and the genuine and fake minutiae set, Q_u. User ID is a 32-bit integer; whereas Q_u consists of 256-bit hash results. Average number of elements in Q_u is 440 for the first dataset, and 550 for the second dataset. Since all of them are 256-bit hash values, 440 points correspond to $256 \times 440 = 112640$ bits. Considering the second dataset, this number becomes $256 \times 550 = 140800$ bits. As a result, the total communication cost of the first message is $32 + 112640 = 112672$ bits = 13.75 KB for the first dataset. On the other hand, for the second dataset, the total communication cost of the first message is $32 + 140800 = 140832$ bits = 17.2 KB.

The second message of the protocol is sent by the server. This message is either a negative acknowledgement which indicates that the user is rejected, or the number of common found minutiae and an HMAC value. Negative acknowledgment is just 1-bit. If the user is not rejected, the message contains 32-bit integer which is the number of common found minutiae, and a 256-bit value which is the HMAC result. Hence, the total cost of communication for this message is $256 + 32 = 288$ bits = 36 bytes.

The protocol continues with the message sent by the user. This message is either *ACCEPT* or *RETRY*. Either one of them can be represented by 1-bit; *ACCEPT* with *1*, *RETRY* with *0*. If the protocol continues with *RETRY* message, the server sends either a negative acknowledgment or a list of HMAC values. Negative acknowledgment is 1-bit. The number of elements in the HMAC values list is equal to $\binom{n_{com}}{n_{com}-1}$, where n_{com} is the total number of common found minutiae. In both datasets, the average number of common minutiae found by the server, n_{com}, is 24. Hence, the average cost of the message which consist of HMAC values is $256 \times \binom{24}{23} = 6144$ bits = 768 bytes.

After that, if the user cannot verify any HMAC value, the user sends a *RETRY* message represented by 1-bit. Otherwise, the user sends a positive acknowledgment with the index of the HMAC value which is verified. Positive acknowledgment is 1-bit, and the index is a 32-bit integer. Thus, the cost of this message is $1 + 32 = 33$ bits.

In the next steps of the protocol, the total communication cost of the messages sent by the user does not change. However, the cost of the messages sent by the server changes as $\binom{n_{com}}{x}$, where $x = n_{com} - 2,\ n_{com} - 3,\ \ldots,\ n_{com}^{key}$, and n_{com}^{key} is the number of minutiae used in the correct key generation. Thus, the communication cost changes at each round until the key is generated. The average number of minutiae with which the key is generated is 16 for the first dataset. Hence, the maximum cost of an HMAC list message is $256 \times \binom{24}{16} = 188280576$ bits = 22.4 MB. For the second dataset, the average number of minutiae in the key generation case is 20. Thus, the maximum cost of a message sent by the server is $256 \times \binom{24}{20} = 2720256$ bits = 332.1 KB.

Total size of the messages sent by the server is approximately 22.4 MB for the first dataset and 332.8 KB for the second dataset. On the other hand, total communication cost of the messages sent by the users is approximately 13.75 KB for the first dataset and 17.2 KB for the second dataset. All of the communication costs given above are quite reasonable given today's Internet speeds.

5.6 Memory Requirements Analysis

Considering the first dataset, on the server side, the average minutiae count of the subjects is 42. In addition to the original minutiae, the points in the T_{dist}-neighborhood are also stored in the server. In the tests, we take T_{dist} as 10. As a result, each original minutiae is represented with $21 \times 21 = 441$ points. On the average, $42 \times 441 = 18522$ points are stored by the server for each subject. These points are hashed values and each is 256 bits long. In total, each subject's template is 578.8 KB. For 30 subjects, the server needs 16.9 MB of storage in total. These values correspond to the case that the server stores only the single hashes of the points. At the verification stage, the server may calculate the double hashes or it may store them as well. If the server stores the double hashes, the storage need is doubled.

On the user side, average minutiae count of the subjects is 40. The user does not calculate the neighboring points. Therefore, the user needs 1.25 KB of storage for hash values of the minutiae. Since the user calculates double hashes as well, the needed storage is 2.5 KB. In addition, the user calculates $10 \times |G_u|$ fake points. On the average,

each user calculates 400 fake points. They cost of 12.5 KB of storage. As a result, the total needed storage is $12.5 + 2.5 = 15$ KB for a user in the first dataset.

If we take the second dataset into account, the average number of minutiae on the server side is 51. Together with the points in the T_{dist}-neighborhood, the total number of points is $51 \times 441 = 22491$. This number of points corresponds to 702.8 KB of storage. Total storage needed for 292 subjects is approximately 200 MB. When the double hashes is calculated at the verification stage, the storage need doubles for a particular subject.

With regard to the memory requirement of the user, we calculate the average number of minutiae of the users in the second dataset as 50. Thus, the user needs 1.56 KB of memory to store hash values of these minutiae. However, the user calculates also the double hashes and the storage need becomes 3.12 KB. In addition, $10 \times |G_u|$ fake points are generated and these will cost 15.63 KB of storage. As a result, $3.12 + 15.63 = 18.75$ KB of storage is needed in total.

All of the memory requirements given above are in acceptable limits. They can easily fit into the memories of current technology devices.

5.7 Comparative Analysis with the Related Work

Since our protocol works with unordered biometrics, the best method to be compared with our protocol is the fuzzy vault. Fuzzy vault is applied to the unordered set of biometric features and it is a highly accepted method for key binding. To the best of our knowledge, almost all fuzzy vault works in the literature do some improvements on the quality of the biometric features, because fuzzy vault requires exactly the same feature points when reconstructing the polynomial. Therefore, we analyse the performance of fuzzy vault by utilizing our quantization and most reliable minutiae selection methods. By doing so, we aim to strike a balance between our protocol and the fuzzy vault.

In Table 3, FAR and FRR percentages of the fuzzy vault method, when it is applied to our datasets, can be seen. Different polynomial degrees imply different key lengths. Since we can represent x and y coordinates of the fingerprints with 10 bit values in both

datasets, we have 20 bit abscissa values when we concatenate them for the fuzzy vault. As a result, a key of length 160 bits can be hidden into a degree 7 polynomial with 8 (= 160/20) coefficients. Similarly, 180 bits of secret correspond to a polynomial with degree 8 and 9 (= 180/20) coefficients. Same rule can be applied for degree 9, 10, 11 and 12 polynomials as well. Since our protocol generates 256 bit keys, it is appropriate to compare our results with a fuzzy vault of degree 11 and degree 12 polynomials.

The fuzzy vault with a polynomial with degree 11 requires at least 12 common minutiae to reconstruct the polynomial successfully. If a genuine user and the server agree on less than 12 points, the user is rejected and this is counted as false reject. On the other hand, if an impostor user and the server can find more than 11 minutiae in common, the impostor subject is accepted and this is counted as false accept. With this method of calculation, a fuzzy vault with degree 11 polynomial has 0.06% FAR and 40.33% FRR for the first dataset. Our method achieves 0.57% EER with the same dataset. In terms of FAR, the fuzzy vault is better than our protocol; however, FRR of the fuzzy vault makes it impractical to be used in real world applications. For the second dataset, a fuzzy vault with a degree 11 polynomial yields 0.80% FAR and 31.33% FRR. Both of the percentages are worse than our protocol's 0.48% EER. The results are similar with degree 12 polynomial as well. Although, the fuzzy vault obtains 0.02% FAR for the first dataset, its FRR is 46.33%. In other words, almost the half of the genuine users are rejected with this method. Similarly, the fuzzy vault, applied to the second dataset, results in 0.38% FAR and 34.53% FRR. Despite the better FAR, our protocol outperforms fuzzy vault in terms of FRR, because 34.53% FRR is unacceptable in any application.

Table 3: FAR and FRR Values of Fuzzy Vault

Dataset	***Polynomial degree***	***Key length (bits)***	***FAR*** *(%)*	***FRR*** *(%)*
1^{st} dataset	7	160	4.55	19.67
	8	180	1.83	28.33
	9	200	0.65	34.33
	10	220	0.19	37.67
	11	*240*	*0.06*	*40.33*
	12	*260*	*0.02*	*46.33*
2^{nd} dataset	7	160	10.64	17.31
	8	180	5.91	20.81
	9	200	3.14	24.60
	10	220	1.62	28.20
	11	*240*	*0.8*	*31.33*
	12	*260*	*0.38*	*34.53*

We also compare our protocol with the fuzzy vault, in terms of brute-force attack complexity. We already mentioned how we measure our protocol's attack complexity. We follow a similar approach while calculating the attack complexity of the fuzzy vault. In order to reconstruct the polynomial with degree d, the attacker must try all possible subsets with size $d+1$ of the whole set including genuine and fake minutiae, Q_u. It is known that in the first dataset the average number of minutiae on the user side is 40. Together with the fake minutiae, the total number of points becomes 440. Thus, the attacker must try to interpolate $\binom{440}{12}$ polynomials, which is yielding a complexity of 2^{77}. Moreover, with a degree 12 polynomial, the attacker must reconstruct $\binom{440}{13}$ polynomials. Hence, in this case, the attack complexity is 2^{82}. As we mentioned before, our protocol has the attack complexity of 2^{94} hash and HMAC verifications for the first dataset. Our protocol's attack complexity is larger than both of them. In addition, the total number of genuine and fake points in the second dataset is 550 and our protocol's attack complexity is 2^{118} hash and HMAC verifications. However, with a degree 11 polynomial, the attack complexity of the fuzzy vault is $\binom{550}{12} = 2^{81}$ polynomial

reconstructions for the second dataset. Moreover, with a degree 12 polynomial, the fuzzy vault attack complexity becomes $\binom{550}{13} = 2^{86}$. None of the fuzzy vault attack complexities is better than ours. As a result, we can conclude that our protocol's resistance against brute-force attacks is stronger than that of the fuzzy vault despite the user side computational complexity overhead of the proposed protocol.

6 Conclusions

In this thesis, we proposed a novel secure key agreement protocol using unordered feature sets of biometric traits. This protocol is exemplified using the fingerprint biometrics. The key is generated by making use of minutiae points in the fingerprint; no random data is used while generating the key. This way, the user is strictly bound to the cryptographic key. Moreover, there is no need to store any helper or random data other than the biometric template of the user at the server side.

Our system uses hash functions and threshold mechanisms while generating the keys. In addition, we carefully designed a fake minutiae generation strategy such that the fake minutiae hide the genuine minutiae without being confused with the genuine minutiae. For this purpose, we defined the concept of *neighborhood relation*. With the help of the neighborhood relation, the fake minutiae increases the verification performance of the system while not leaking any information to the attacker.

We analysed the security performance of our protocol in different aspects and with two different datasets. Our results showed verification performance of 0.57% EER for the first dataset and 0.48% EER for the second dataset. The resistance of our protocol against intelligent brute-force, replay and impersonation attacks was also analysed. Such attacks require 2^{94} and 2^{118} trials on the average for first and second datasets, respectively, which was shown to provide good computational security. In addition, we employed entropy-based randomness analyses of the agreed keys. Our analyses showed that approximately 84% of the keys' entropy values are above 0.994 and all of the keys' entropy values are above 0.98 in the first dataset. Similarly, in the second dataset, 99% of the keys have entropy values that are greater than 0.98, and almost all of the keys has entropy values that are greater than 0.96. Both of the results

imply that the keys are random enough to be used as cryptographic keys. Besides, the distinctiveness of the generated keys was measured using the Hamming distance metric. Our Hamming distance-based analyses showed that the same users' and different users' keys are quite indistinguishable from each other, regardless of the dataset. In terms of computational complexity, the server side operations took approximately 2^9 and 2^{17} hash and HMAC calculations, and the user side operations took approximately 2^{39} and 2^{41} hash and HMAC calculations for first and second datasets, respectively. Although the user side complexity seem to be high, it is actually a trade-off between using and not using any helper data to ease the agreement on the feature points in the key generation process. Using a helper data leads to the necessity of new precautions to avoid compromise of this helper data. Besides, the communication complexity and the memory requirements of the system are acceptable for implementing the protocol according to today's technology.

Apart from the performance and security evaluation of the proposed protocol, a comparative analysis with an existing work (fuzzy vault) was also performed. Our analyses showed that our protocol outperforms the fuzzy vault in terms of both verification performance and attack complexity point of views. However, the computational complexity of the user side operations in our protocol is high and this fact should be considered.

As a future work, template renewal process can be designed on the server side. In other words, templates can be arranged as cancelable biometrics. Moreover, our protocol can be adopted to other biometrics with ordered set of features, such as the iris biometrics.

References

[1] A. Juels and M. Sudan, "A fuzzy vault scheme," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.

[2] D. Akdogan, D. Karaoglan Altop, and A. Levi, "Secure key agreement using pure biometrics," in *IEEE Conference on Communications and Network Security*, 2015, pp. 191–199.

[3] B. Miller, "Vital signs of identity [biometrics]," *IEEE Spectrum*, vol. 31, no. 2, pp. 22–30, 1994.

[4] A. Jain, A. Ross, and S. Pankanti, "Biometrics: a tool for information security," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 125–143, 2006.

[5] U. Uludag, S. Pankanti, S. Prabhakar, and A. Jain, "Biometric cryptosystems: issues and challenges," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 948–960, 2004.

[6] A. Jain, S. Pankanti, S. Prabhakar, L. Hong, and A. Ross, "Biometrics: a grand challenge," in *International Conference on Pattern Recognition*, vol. 2, 2004, pp. 935–942.

[7] S. Pankanti, R. Bolle, and A. Jain, "Biometrics: The future of identification," *Computer Journal*, vol. 33, no. 2, pp. 46–49, 2000.

[8] S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. NY, USA: Knopf Doubleday Publishing Group, 2011.

[9] D. Kahn, *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. NY, USA: Simon and Schuster, 1996.

[10] W. Trappe and L. Washington, *Introduction to Cryptography with Coding Theory*. NJ, USA: Pearson Prentice Hall, 2006.

[11] D. E. Robling Denning, *Cryptography and Data Security*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1982.

[12] W. Stallings, *Cryptography and Network Security*. NY, USA: Prentice Hall, 2011.

[13] ——, *Computer Networking with Internet Protocols and Technology*. NJ, USA: Pearson Prentice Hall, 2004.

[14] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. USA: CRC press, 1996.

[15] D. Eastlake 3rd and P. Jones, "Secure hash algorithm 1 (SHA1)," RFC-3174, US, 1995.

[16] National Institute of Standards and Technology, *FIPS PUB 180-2: Secure Hash Standard*. pub-NIST, 2002.

[17] H. Gilbert and H. Handschuh, "Security analysis of SHA-256 and sisters," in *Selected areas in cryptography. Springer Berlin Heidelberg*, 2004, pp. 175–193.

[18] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," RFC-2104, US, 1997.

[19] N. Ratha, J. Connell, and R. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, pp. 614–634, 2001.

[20] J. Dong and T. Tan, "Security enhancement of biometrics, cryptography and data hiding by their combinations," *IET Conference*, pp. 239–244(5), 2008.

[21] F. Hao, R. Anderson, and J. Daugman, "Combining crypto with biometrics effectively," *IEEE Transactions on Computers*, vol. 55, no. 9, pp. 1081–1088, 2006.

[22] S. Kanade, D. Camara, D. Petrovska-Delacrtaz, and B. Dorizzi, "Application of biometrics to obtain high entropy cryptographic keys," in *World Acad. Sci. Eng. Tech*, 2009, p. 330.

[23] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy, and B. Vijaya Kumar, "Biometric encryption: enrollment and verification procedures," in *International Society for Optics and Photonics*, vol. 3386, 1998, pp. 24–35.

[24] F. Monrose, M. K. Reiter, and S. Wetzel, "Password hardening based on keystroke dynamics," *International Journal of Information Security*, vol. 1, no. 2, pp. 69–83, 2002.

[25] F. Monrose, M. Reiter, Q. Li, and S. Wetzel, "Cryptographic key generation from voice," in *IEEE Symposium on Security and Privacy*, 2001, pp. 202–213.

[26] D. Karaoğlan and A. Levi, "A survey on the development of security mechanisms for body area networks," *The Computer Journal*, vol. 57, no. 1, pp. 1484–1512, 2014.

[27] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *ACM Conference on Computer and Communications Security*, 1999, pp. 28–36.

[28] S. Kanade, D. Camara, E. Krichen, D. Petrovska-Delacretaz, and B. Dorizzi, "Three factor scheme for biometric-based cryptographic key regeneration using iris," in *Biometrics Symposium*, 2008, pp. 59–64.

[29] A. Stoianov, "Security of error correcting code for biometric encryption," in *International Conference on Privacy Security and Trust*, 2010, pp. 231–235.

[30] C. Rathgeb and A. Uhl, "Statistical attack against iris-biometric fuzzy commitment schemes," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2011, pp. 23–30.

[31] K. Nandakumar, A. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: Implementation and performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 4, pp. 744–757, 2007.

[32] U. Uludag and A. Jain, "Securing fingerprint template: Fuzzy vault with helper data," in *Computer Vision and Pattern Recognition Workshop*, 2006, pp. 163–163.

[33] P. Mihailescu, "The fuzzy vault for fingerprints is vulnerable to brute force attack," *Computing Research Repository*, 2007.

[34] W. Scheirer and T. Boult, "Cracking fuzzy vaults and biometric encryption," in *Biometrics Symposium*, 2007, pp. 1–6.

[35] A. Kholmatov and B. Yanikoglu, "Realization of correlation attack against the fuzzy vault scheme," *International Society for Optics and Photonics*, vol. 6819, 2008.

[36] C. Orencik, T. B. Pedersen, E. Savas, and M. Keskinoz, "Improved fuzzy vault scheme for fingerprint verification," *International Conference on Security and Cryptography*, pp. 37–43, 2008.

[37] A. M. Bazen and S. H. Gerez, "Fingerprint matching by thin-plate spline modelling of elastic deformations," *Pattern Recognition*, vol. 36, no. 8, pp. 1859–1867, 2003.

[38] "Neurotechnology Verifinger sample db," http://www.neurotechnology.com/, accessed: 2015-05-01.

[39] "Cross Match Verifier 300 lc," http://www.crossmatch.com/verifier-300-lc/, accessed: 2015-05-01.

[40] "Papilon DS22N," http://www.papilonint.com/ds-22n-overview/cevg, accessed: 2015-12-13.

[41] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, 1948.

[42] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[43] M. Juliato and C. Gebotys, "FPGA implementation of an HMAC processor based on the SHA-2 family of hash functions," University of Waterloo, Tech. Rep., 2011.