SECURE KEY AGREEMENT USING CANCELABLE AND NONINVERTIBLE BIOMETRICS BASED ON PERIODIC TRANSFORMATION

by

Laleh Eskandarian

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

July 2017

SECURE KEY AGREEMENT
USING CANCELABLE AND NONINVERTIBLE BIOMETRICS
BASED ON PERIODIC TRANSFORMATION

APPROVED BY:

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

Prof. Berrin Yanıkoğlu .....

Assist. Prof. Cengiz Örencik .....

DATE OF APPROVAL: .....

ABSTRACT

SECURE KEY AGREEMENT

USING CANCELABLE AND NONINVERTIBLE BIOMETRICS

BASED ON PERIODIC TRANSFORMATION

Laleh Eskandarian

Master Thesis, July 2017

Supervisor: Prof. Albert Levi

Keywords: Biometrics, Bio-cryptography, Cancelable Biometrics, Noninvertible Biometrics, Periodic Transformation, Fingerprints, Key Agreement, Security Analysis.

Nowadays, many of the security-providing applications use biometric-based authentication, such as electronic banking, health and social services, commercial applications and law enforcement. However, since each person's biometrics is unique and not replaceable, once it is compromised, it will be compromised forever. Therefore, it is indeed hard for the users to trust biometrics.

To overcome this problem, in this thesis, we propose a novel protocol SKA-CaNPT: Secure Key Agreement Protocol using Cancelable and Noninvertible Biometrics based on Periodic Transformation. In this research, we use a periodic transformation function to make our biometrics cancelable and noninvertible. At the very end of our SKA-CaNPT protocol, the user and the server make an agreement on a symmetric shared key that is

based on the feature points of the biometrics of the user. As a proof of concept, we apply our SKA-CaNPT protocol on fingerprints. In our protocol, after extracting minutiae from the fingerprints, we first employ a periodic transformation function and then we categorize our minutiae points in a predefined neighborhood by using a threshold-based quantization mechanism. Our SKA-CaNPT protocol runs in a round-manner and in each round, the server decides about the acceptance or rejection of the user according to the similarity score of the common minutiae. In addition, if the transformed data is compromised, it can be renewed just by changing one of the inputs of our transformation function.

Besides, we apply different security analyses on our protocol. First of all, we use Shannon's entropy to analyze the randomness of the agreed keys, and it shows that the generated keys have enough randomness. Secondly, to analyze the distinctiveness of the agreed keys, we use the Hamming distance metric, results of which show that the keys of different people are distinguishable from each other. Moreover, according to the low IKGR (Incorrect Key Generation Rate), high CKGR (Correct Key Generation Rate) and high attack complexity possessed by our SKA-CaNPT protocol, we can conclude that our scheme is secure against brute-force, replay and impersonation attacks.

ÖZET

PERİYODİK DÖNÜŞÜM İLE OLUŞTURULAN İPTAL EDİLEBİLİR VE GERİ DÖNÜŞTÜRÜLEMEZ BİYOMETRİK VERİLERİN KULLANILDIĞI GÜVENLİ ANAHTAR ANLAŞMASI

Laleh Eskandarian
Yüksek Lisans Tezi, Temmuz 2017
Danışman: Prof. Dr. Albert Levi

Anahtar Sözcükler: Biyometrik, Biyo-kriptografi, İptal Edilebilir Biyometrik, Geri Dönüştürülemez Biyometrik, Periyodik Dönüşüm, Parmak İzi, Anahtar Anlaşması, Güvenlik Analizi.

Günümüzde, elektronik bankacılık, sağlık ve sosyal hizmetler, ticari uygulamalar ve hukuki uygulamalar gibi güvenlik sağlayan birçok uygulama biyometrik tabanlı kimlik doğrulama kullanmaktadır. Fakat, her bir kişinin biyometrik verisi benzersiz ve değiştirilemez olduğundan, söz konusu biyometrik verinin gizliliği bir kez ifşa edildiğinde, aslında bu biyometrik veri sonsuza dek kullanılamaz hale gelecektir. Bu nedenle, kullanıcıların biyometrik verilere güvenmesi aslında zordur.

Bu tez ile, yukarıda bahsi geçen sorunun üstesinden gelmek adına, yeni bir protokol olan SKA-CaNPT protokolünü öneriyoruz: Periyodik Dönüşüm ile Oluşturulan İptal Edilebilir ve Geri Dönüştürülemez Biyometrik Verilerin Kullanıldığı Güvenli Anahtar Anlaşması.

Bu çalışmada, biyometrik verileri iptal edilebilir ve geri dönüştürülemez kılmak için periyodik bir dönüşüm fonksiyonu kullanmaktayız. SKA-CANPT protokolünün sonunda, kullanıcı ve sunucu, kullanıcının biyometrik verisinin özellik noktalarına dayanan bir simetrik paylaşılan anahtar üzerinde anlaşırlar. SKA-CANPT protokolünün kavram kanıtı için parmak izlerini kullandık. Protokolde, parmak izlerinin özellik noktaları çıkarıldıktan sonra, öncelikle periyodik dönüşüm fonksiyonu kullanıyoruz ve sonrasında ise eşik tabanlı bir niceleme yöntemiyle daha önceden belirlenmiş komşuluk ilişkilerine göre bu özellik noktalarını kategorize ediyoruz. SKA-CANPT protokolü turlu bir düzende çalışır ve her turda sunucu, parmak izlerinden çıkarılan özellik noktalarının benzerlik puanına bakarak kullanıcının kabul veya reddine karar verir. Buna ek olarak, dönüştürülmüş veriler bir şekilde ifşa edilirse, dönüşüm fonksiyonunun girdilerinden biri değiştirerek yeni bir iptal edilebilir ve geri dönüştürülemez biyometrik veri oluşturulabilir.

Ayrıca, protokolümüze farklı güvenlik analizleri uyguladık. İlk olarak, üzerinde anlaşılan anahtarların rasgeleliğini analiz etmek için Shannonun entropi analizini kullandık ve sonuçlar ilgili anahtarların yeterli ölçüde rasgele olduklarını gösterdi. İkinci olarak, üzerinde anlaşılan anahtarların değişkenliğini analiz etmek için Hamming uzaklığı analizini kullandık ve sonuçlar farklı insanların biyometrik verilerinin kullanımı ile oluşturulan anahtarların birbirlerinden farklı olduklarını gösterdi. Dahası, düşük IKGR (Yanlış Anahtar Üretimi Oranı), yüksek CKGR (Doğru Anahtar Oluşturma Oranı) ve SKA-CANPT protokolünün sahip olduğu yüksek saldırı karmaşıklığına bakarak söyleyebiliriz ki önerdiğimiz protokol kaba kuvvet, yeniden oynatma ve kimliğe bürünme saldırılarına karşı güvenlidir.

to my beloved family

ACKNOWLEDGMENTS

I would first like to express my sincere thanks to my thesis supervisor Prof. Albert Levi for the continuous support of my master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. Besides my supervisor, I would like to thank the rest of my thesis committee: Prof. Berrin Yanıkoğlu and Assist. Prof. Cengiz Örencik, for their insightful comments.

My special thanks also goes to Dr. Duygu Karaoğlan Altop for her insightful comments and encouragement. Without her guidance and persistent help and her precious support it would not be possible to conduct this research. I also want to thank all my labmates for providing such a nice environment and specially, my project partner Dilara Akdoğan. She always helped and supported me.

Finally, I must express my very profound gratitude to my parents, to my grand mother, to my sister Ladan and my beloved Ehsan for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

I would like to thank TÜBİTAK for supporting this thesis under grant 114E557.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 *Introduction*

In everyday life, people use traditional authentication schemes to verify their identity, in which the secret key is either something they have (e.g. smart card, SIM card in mobile phone), or something they know (e.g. password, PIN code). Despite the wide use of these techniques, there are a lot of limitations. For instance, a personal device like a mobile phone or smart card could be stolen, lost or borrowed, or a password could be guessed. The problem of the above-mentioned methods is that the difference between the authorized person and the imposter is indistinguishable. One way of solving such a problem is to choose a complex password; however, because of being hard to be kept in mind, people tend to write their passwords down, which causes a security threat. In order to overcome these limitations, authentication using something we are (e.g. biometrics: fingerprint, iris, face, etc) could be useful [3]. There are a lot of advantages of using biometric-based authentication, such as, users don't need to keep their passwords in a secure place or remember them, or from another point of veiw, users cannot forget their passwords or loose them. In addition, it is very difficult to forge someone's biometrics in comparison to forging documents.

We have to protect templates in order to avoid some problematic consequences. First of all, if biometrics is stolen, it is lost forever; it means that in the case that someone's biometrics stolen or lost, (s)he could not replace it for his/her whole life. Secondly, because of the fact that the users can apply their biometric traits in many different applications, each one of these applications will be confronted with a risk when the corresponding biometrics is compromised once. This is known as the cross matching problem of biometrics. It means that if the users apply the same biometrics in different applications, they

could be potentially tracked. Finally, biometrics are not renewable. Users can renew their passwords or PINs by resetting them, but not their biometric traits.

The technique that can overcome the aforementioned problems is called *cancelable biometrics* [4] [5]. The main idea of cancelable biometrics is mapping original biometric templates into a new data prior to the matching procces, and storing new biometric template in the database [3]. Though, if the transformed biometric data is compromised, by only changing the matching characteristics, the biometric template could be renewed. The noninvertibility property can be introduced by applying a one-way transformation function while generating the cancelable biometrics. In that case, the template's protection strength relies on the transformation function.

The combination of biometrics and cryptography, which provides higher security and privacy, is referred to as $crypto - biometry$ or $bio - cryptography$. In bio-cryptographic applications, the common encryption and decryption is applied over the data, but the secret keys are driven from the biometric data. These secret keys should be (i) long enough to avoid them being guessed by an attacker, (ii) random enough to contain sufficient entropy, and (iii) distinctive enough to avoid them being forged. By this means, the security of the system can be maintained. Though, satisfying the secure key requirements due to the invariant characteristic of biometrics is really hard and complex. Moreover, the main obstacle of the above-mentioned combination is that biometrics are noisy. It means that we can only expect an approximate match of the stored template. On the other hand, cryptography needs the exactly right key to pass, because otherwise the protocol will fail [6]. Thus, when designing bio-cryptographic systems, this variation should be well analysed. For instance, error correction codes or fuzzy key binding methods, like *fuzzy commitment* or *fuzzy vault*, can be used to handle the above-mentioned problem.

1.1 Our Contribution

In this thesis, we present a novel secure key agreement protocol, SKA-CaNPT: Secure Key Agreement Protocol using Cancelable Noninvertible Biometrics based on Periodic Transformation. Our proposed approach is based on the SKA-PB protocol, introduced by Akdoğan et al. [7], and the idea of using a periodic transformation function to provide cancelability and noninvertibility properties, suggested by Dang et al. [8], which are described in detail in Section 3.1. SKA-CaNPT uses cancelable and noninvertible biometrics while generating the cryptographic keys that will be agreed upon. The reason behind this is to keep the original biometrics safe and secure so that the shared symmetric key can be extracted directly from the stored biometric template. The properties of cancelability and noninvertibility are fulfilled through a periodic transformation function that is applied on the captured biometrics. As a proof of fact, we apply our SKA-CaNPT protocol on fingerprints, which are known to be unordered set of biometrics. In other words, we apply our protocol on unordered set of minutiae points. In order to hide the genuine minutiae points, and thus to decrease the risk of information leakage to the attacker, a number of fake minutiae points are generated randomly. It is important to note here that, in order for the fake minutiae points to be indistinguishable from the genuine minutiae points, the same periodic transformation function is also applied on the fake minutiae points as well.

Our SKA-CaNPT protocol works in round manner and in each round, the server and the user try to agree on a symmetric shared key. First of all, they try to find a set of common minutiae points, and then, depending on the defined threshold value and the calculated similarity score, the user will either be accepted or (s)he will be rejected and a new round will start. In addition, the keys generated using our SKA-CaNPT protocol can be renewed and/or revoked if a transformed data is compromised, only by changing the input of the transformation algorithm and applying it on the original biometric data.

The security performance of our SKA-CaNPT protocol is analyzed from different aspects. From biometrics aspect, our method presents a high verification performance

proved by the achieved low IKGR (Incorrect Key Generation Rate) values and high CKGR (Correct Key Generation Rate) values. Additionally, we show that our SKA-CaNPT protocol is resistant against brute-force, replay and impersonation attacks. On the top of it, the quality of the agreed symmetric shared keys are high, demonstrated through randomness and distinctiveness evaluations. Therfore, we can use the generated keys as cryptographic keys since each key is different from the other keys, and they have enough randomness. In addition, our protocol is resistant to any attack that compromises the original biometrics since we use a one-way periodic transformation function to make our protocol cancelable and noninvertible. Furthermore, we analyze the communication, computation and storage requirements, along with the complexity of our protocol. Finally, we compare our protocol with SKA-PB protocol, which is introduced by Akdoğan et al. [7], and cancelable fuzzy vault approach that is presented by Dang et al. [8].

1.2 Outline

The thesis is organized into six sections. Section 2 gives background information and Section 3 reviews the related works. In Section 4, we introduce our proposed SKA-CaNPT protocol (Secure Key Agreement Protocol using Cancelable AND Noninvertible Biometrics based on Periodic Transformation). In Section 5, we discuss about the performance and the security of our SKA-CaNPT protocol, along with analyzing its complexity and memory requirements, and we compare it with the existing other works. Finally, Section 6 concludes our work and discusses about other possible future works.

2 *Background Information*

In this section, we describe the definitions and the notations that are utilized in this thesis. First of all, we discuss briefly about biometrics, and then we continue with cryptography. Thereafter, we explain about biometric cryptosystems, and finally, we present cancelable and noninvertible biometrics.

2.1 Biometrics

If we look back to many many years ago, we can find the images of handprints and footprints in prehistoric caves all around the world. Later, in Babylonia, researchers found fingerprints over the clay tablets. They also found that Chinese used thumbprints on their clay seals and in the 14^{th} century in Persia, on official documents, fingerprints were used. Starting from the late 1960s, in which fingerprints were very popular, the idea of using biometrics for user identification and user authentication has been formed [9] [10].

Authenticating people by checking their personal characteristics is known as biometric authentication. The considered biometrics can either be a behavioral characteristic, such as typing rhythm, keyboard tapping, voice, signature or the way a person walks, or a physical characteristics, such as fingerprint, hand geometry, iris, face or DNA. Therefore, we can divide the biometrics into two main categories: (i) something we are (physical biometric traits) and (ii) something we do (behavioral biometric traits).

Health problems or stress could have a direct impact on behavioral characteristics of a user; thus, they are less stable in comparison with physical characteristics. Therefore,

physical characteristics are supposed to be more useful and more popular for biometric authentication.

On the other hand, biometric authentication systems have two phases: (i) enrollment phase and (ii) verification phase. In the enrollment phase, users register their biometric traits to the system, using which the biometric template is constructed through feature extraction. In the verification phase, another feature vector is extracted from the newly captured biometric trait of the same user, and it is compared with the pre-registered one. Because of the fact that the biometrics are not exactly the same of each other in different acquisitions, which is known as intra-person distinction, the correlation between the newly constructed biometric template and the one from the database that has been captured in the enrolment phase is of interest. Depending on the correlation among these two templates, a decision is made about the acceptance or rejection of the user. In this comparison process, some authorized people may get rejected erroneously (false rejection) and some unauthorized people may get accepted by mistake (false accept). In this regard, biometric systems are mostly analyzed according to their false rejection rates (FRR) and false acceptance rates (FAR). Also, error equal rate (EER) is another way for analyzing a biometric security system. When the FRR becomes equal to FAR, the common value indicates EER. The biometric system becomes more secure as the EER decreases.

As mentioned above, physical characteristics of biometrics are more stable than that of the behavioral ones. Herewith, below, we discuss briefly about fingerprint, palm, iris and face. For fingerprint recognition, first the image of the fingertip is captured and then the minutiae are extracted using the valleys and/or the ridges, as shown in 2.1 [1]. In the authentication/verification phase, the images or templates that are collected at that moment are compared with that of the ones from the enrollment phase. Identification based on fingerprints is the oldest method among the other biometrics. It has been used in different applications for more than a century. It is more popular due to the facts that (i) its acquisition is easy, (ii) it includes 10 different resources that is more than the other biometrics, and (iii) the government uses it for law enforcement and immigration purposes.

Besides, hand geometry recognition is a method in which the shapes of the users'hands identify them. Hand geometry readers measure and record different dimensions of hands, such as length, width and thickness. They record both top surface image and side image of a hand, as given in Figure 2.2 [1]. Only 9 bytes of data is stored as a template, which in comparison to the other biometric templates is extremely low [11]. The main shortcoming of hand geometry is that it is not completely unique.

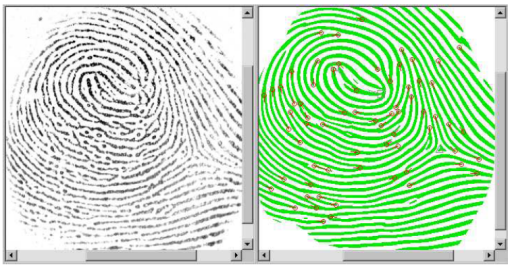

Figure 2.1: Fingerprint Recognition [1]

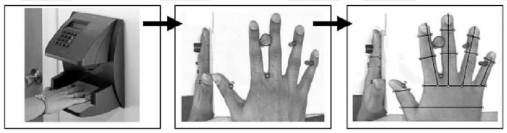

Figure 2.2: Hand Geometry Recognition [1]

Moreover, face recognition is based on recording the image of the users'whole face so that the key features could be extracted from it. These key features consist of relative distances between eyes, nose, mouth, jaw and cheekbones, as shown in Figure 2.3 [2]. A unique template is created according to the mentioned information just after dimension reduction like the PCA technique, which is applied to eliminate the unnecessary components that won't be used during image reconstruction [12]. In face recognition systems, features are extracted from the lines and the segmented regions based on region segmentation and line of interest, as shown in Figure 2.3 [2]. After that, the ordered set of face features are constructed according to the classification of these feature vectors. Indeed, people recognize each other by using this method.

On the other hand, there are two different categories in eye-based authentication: (i) iris recognition, and (ii) retina recognition, among which iris recognition is approved

to be more accurate than retina recognition. In iris recognition, data are extracted from the external colorful ring around the pupil, as shown in Figure 2.4 [13]. As the biometric template of the iris, the iris-code, which is the binary string extracted from the eye image, along with a bit mask are stored in the database. Iris-based authentication could easily be done with checking the matching/mismatching score of two iris-codes.

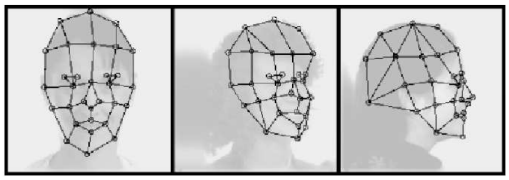

Figure 2.3: Face Recognition [2]

Figure 2.4: iris Recognition

As discussed above, biometrics have physical and behavioral characteristic. Recording the behavioral characteristics is much more easier than capturing the physical characteristics. For instance, signatures are being collected in many different applications for authentication and identification purposes. Also, voice could be recorded easily with a cheap and simple device. The only problem with these methods is that they are not as stable as physical characteristics-based methods. For instance, a user's signature may not be consistent, meaning that it could change over time, and thus, it could cause problems for the corresponding signature verification system. As another example, in a voice verification system, if there exists a little noise in the background, then the spoken phrase could not be recorded accurately, which becomes problematic for the voice verification system. In brief, physical characteristics are utmost persistent during the users' lifetime, unless an injury happens.

In this thesis, biometrics is used for key generation purposes, indeed fingerprints are utilized in our protocol evaluations. Since, our protocol is not a pure biometric authentication/verification protocol, but rather it is a key generation/agreement protocol, we use correct key generation rates (CKGR) and incorrect key generation rates (IKGR). The key agreement protocol becomes more secure as the IKGR decreases and CKGR increases.

2.2 Cryptography

Cryptography is a Greek word, within which *crypto* means hidden or secret, and *graphein* means writing. It is called to a study that makes the communication between two parties secure in the presence of adversaries. Cryptography is not just a new knowledge, we can see it in prehistoric Egyptian hieroglyphs: specific communication tools are used by the high-casts to protect their community. Also, in ancient Greece, the main reason of winning/loosing a battle was secret communication. These types of cryptography is called conventional cryptography. The modern cryptography started from the middle of the last century. Mr. Horst Feistel, who worked at IBM in 1977, designed Data Encryption Standard (DES), which was the starting point of cryptography in information technology. Nowadays, most of the communication systems, such as banking networks, internet, cell phones and etc., are using this kind of cryptography.

Cryptography is used to provide a number of goals: confidentiality, integrity, authentication and identification. In every cryptographic model, the following terms are utilized [14] [15]:

(i) *Plaintext* is the original message,

(ii) *Ciphertext* is the output of the encryption algorithm,

(iii) *Secret Key* is the key that is used through the encryption process (it is selected independently),

(iv) *Encryption* is the process or algorithm that encodes the plaintext by using the secret key, output of which is called the ciphertext, and

(v) *Decryption* is the process or algorithm that decodes the ciphertext into the plaintext using the secret key.

The process of secure communication among two parties is shown in Figure 2.5. Additionally, it is important to note here that encrypting a plaintext with two different keys will yield to two different ciphertexts, and likewise, encrpyting two different ciphertexts with the same key will yield to two different ciphertexts.

In the below subsections, we describe the cryptographic primitives that we use in our constructions.

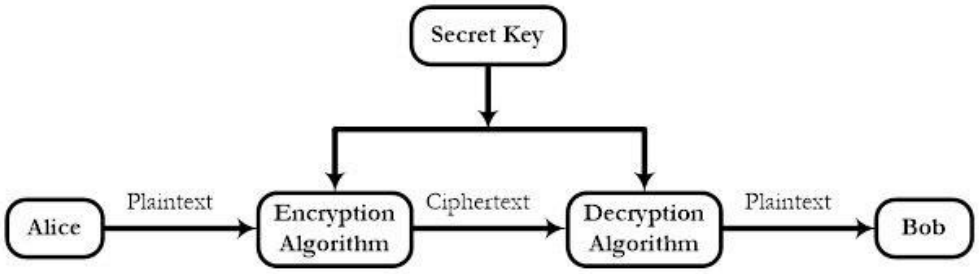

Figure 2.5: General methodology of cryptographic secure communication

2.2.1 Symmetric Key

Cryptographic systems are divided into two different categories depending on the key distribution process: (i) symmetric key cryptography, and (ii) asymmetric key cryptography. In the former, which is visualized in Figure 2.6, the same secret key is used for all communications between the two parties, while in the latter, public keys (which are known to everyone) and private keys (which are known only for the owner) are required for encryption and decryption operations, respectively. Since our protocol uses symmetric key, cryptography, only it will be described in detail in the following paragraphs.

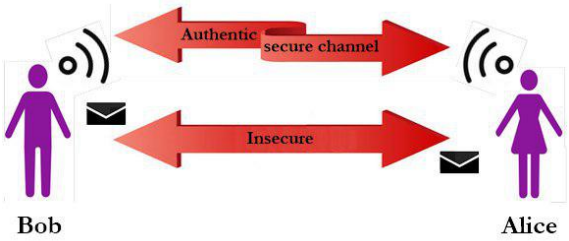

Figure 2.6: Secret Key Cryptography

In the symmetric key cryptography, in all cryptographic operations, like encryption and decryption, the same key is utilized, and that's why sometimes it is called as secret key cryptography [16]. As defined in Kerckhoff's Principle [17], the secrecy of a cryptographic system depends on the strength of its secret key, since every other detail should be public. Therefore, in symmetric key cryptography, the only thing that is important for its secrecy is the strength of its secret encryption key. It means that an attacker should not be able to figure out what the secret key is using a number of ciphertexts and/or plaintexts.

The most important advantage of symmetric key cryptography is its efficiency when implementing the primitive operations [18], while the disadvantages of symmetric key cryptography include the key exchange problem and proliferation of the secret keys. To communicate privately, each pair of the users need unique and specific secret keys for their communication sessions, so this may lead to proliferation of the keys. Therefore, to establish a secure communication, a key agreement protocol should run between each pair of the users to exchange the secret key: it is the process that should be done before sending a message. Thus, key exchange is neither easy nor trivial, and it makes the key management process difficult.

Key management establishes methods for generating, exchanging, using and storing the secret keys. Key establishment could only be *key distribution* or *key agreement* between two or more parties [19] [14]. There are lots of different *key distribution* schemes, but in general, it is known to be a process in which one of the users creates a secret key and sends it to the other user through a secure channel, because otherwise, an adversary could generate the key and send it to both of the communicating entities. On the other hand, in *key agreement* process, a shared secret key is derived by two or more communicating

parties. Different from key distribution, in key agreement, none of the communicating parties has any information about the secret key prior to the process. It is important to note here that the key agreement protocol should not leak any information about the secret key to be agreed upon to the adversary.

In this thesis, we propose a novel secure key agreement protocol, in which the server and the user agree on a secure key that is generated using the fingerprint features, as mentioned in Section 2.1.

2.2.2 Hash Functions

Hash function is a mathematical function which converts a numerical value of an arbitrary length to a numerical value of a fixed length, as shown in Figure 2.7. The output value of a hash function is called a message digest or a hash value. There are two strict requirements that a hash function should satisfy: (i) it should provide enough randomness for the outputs, and (ii) the output should change completely even if a bit changes in the input. Moreover, a cryptographic hash function should contain the following properties as well: (i) it should has pre-image resistance, which means that reversing a hash function should be computationally hard, and (ii) it should has collision resistance, which means that for a particular hash function output, we should not be able to find two different inputs, even with different lengths. There are two common applications for hash function utilization, regarding the aforementioned properties. The first one is password storage, in which instead of storing the password in clear, the hash value of it is stored in the system. The second one is data integrity check, through which users can make sure about data correctness. In addition, hash functions can also be used in digital signatures and message authentication [20].

In the following paragraphs, SHA and HMAC are explained as they are utilized in this thesis.

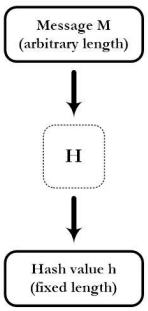

Figure 2.7: Hash Function

Secure Hash Algorithm (SHA)

NIST (National Institute of Standards and Technology) and NSA (National Security Agency) have developed a set of algorithms in 1993, which are a family of cryptographic hash functions that referred as Secure Hash Algorithm (SHA). The most commonly known versions of SHA are SHA-1, SHA-256, SHA-384 and SHA-512, the details of which are as given in Table 2.1. SHA-1 was proposed in 1995 to overcome the weaknesses of SHA-0, which is the oldest in the family [21]. On the other hand, the others, i.e., SHA-256, SHA-384 and SHA-512, are known to be the subsets of SHA-2, which was proposed in [22]. Since 2005, NIST have decided to use SHA-2 as the standard hash algorithm.

In our key agreement protocol that we propose in Section 4, we use SHA-256, which has a message digest of size 256, as given in Table 2.1.

Table 2.1: Various SHA versions

Parameter	***SHA-1***	***SHA-256***	***SHA-384***	***SHA-512***
Message digest size (in bits)	*160*	*256*	*384*	*512*
Parameter	$< 2^{64}$	2^{64}	2^{128}	2^{128}
Message size (in bits)	512	512	1024	1024
Block size (in bits)	32	32	64	64
Word size (in bits)	80	64	80	80

Keyed-Hash Message Authentication Code (HMAC)

A keyed-hash message authentication code (HMAC) is a particular kind of MAC (Message Authentication Code) that consists of a secret key and a hash function [23]. The most important advantages of HMAC are that it is not only much more faster but also much more efficient than symmetric encryption [14]. As mentioned in Section 2.2.2, although the length of a hash function's input can be an arbitrary value, its output is a fixed length message. Herewith, the efficiency of the cryptosystem can be maximized by decreasing the length of a long message with the use of HMAC. In other words, HMAC is a MAC that is used to authenticate data by applying a hash function like SHA-256 over the data with a secret key. It is mostly analogous to using digital signatures, but just HMAC uses symmetric key cryptography, while digital signature uses asymmetric key cryptography.

In our key agreement protocol that we propose in Section 4, we use SHA-256 in our HMAC algorithm, as also mentioned above.

2.3 Biometric Cryptosystems

We need cryptography in order to carry out a trustful communication. In traditional cryptosystems, users are authenticated based on secret keys, in which if the key cannot be kept secret, the authentication fails. Indeed, using the biometric characteristics of a user could solve such problems. Therefore, researchers proposed a combination of cryptography and biometrics to provide an effective solution for data security, and they named this combination *bio-cryptography* or *crypto-biometry*.

Bio-cryptographic systems are analogous to password-based key generation systems, just instead of traditional passwords, biometric features are used to generate a cryptographic key, or to secure the cryptographic key [24]. Thus, the main purpose of developing biometric cryptosystems is either to generate a symmetric key by using biometric features or to secure a cryptographic key with the help of biometric features [25]. We

can categorize biometric cryptosystems to (i) key binding, and (ii) key generation. *Key binding* is referred to a system in which a secure sketch is obtained by combining the biometric template with a cryptographic key. On the other hand, *key generation* is a method using which a secure sketch is derived directly from the biometric template, and it is used as the cryptographic key itself.

Since cryptographic systems require the exact same key to accept a user, in key binding and key generation based biometric cryptosystems, the intra-user diferences of biometric templates are problematic. To solve this type of problems, *fuzzy key binding* methods such as *fuzzy commitment* [26] and *fuzzy vault* [27] are presented. Among the above techniques, the *fuzzy vault* scheme became popular for biometric template protection, which has been implemented for iris [28], face [29] and fingerprint [30–34].

The most interesting context in biometric cryptosystems is key management; but, generating the key from biometric features is difficult due to the biometrics being noisy and including intra-user variations. Chang et al. [35] and Veilhauer et al. [36] propose a user specific quantization scheme, for generating secret keys from biometric data, in which helper data is the intra-user variations. Besides, *fuzzy extractor* and *secure sketch* are proposed by Dodies et al. [37] in the field of biometric key generation. The fuzzy extractor generates the cryptographic keys based on the biometric features, and the secure sketch is a helper data, which has less information leakage. The two famous methods of fuzzy extractors are *fuzzy commitment* and *fuzzy vault*, which are described in detail in the below paragraphs.

Fuzzy Commitment

Fuzzy commitment [26] is used to secure the biometric features of the individuals, which are in the form of binary vectors such as the iris code. The method is composed of two phases: (i) enrollment phase, and (ii) authentication phase, details of which are visualized in Figure 2.8. In the enrollment phase, first a key k that is in the form of a codeword is selected and a hash function is applied on it, $H(k)$. Then, XOR of the biometric template

and *H(k)* is calculated, which actually is the encrypted message of the fuzzy commitment scheme. In the authentication phase, XOR of the biometric query template's feature vector and the encrypted message is calculated, and then the hash function is applied on the result to check its correctness. It has been shown that, with the fuzzy commitment scheme, not only the key but also the biometrics of the user could be reconstructed by the attacker, with the use of error correction codes and statistical attacks [26].

Fuzzy Vault

The fuzzy vault scheme [27] is one of most popular biometric template protection schemes that in contrast with fuzzy commitment, is applicable on unordered set of biometric data such as fingerprints. The method is also composed of two phases: (i) enrollment phase, and (ii) authentication phase. In the enrollment phase, first, a secret key *k* is selected. After that, a polynomial *P(x)* is indexed with the selected secret key *k*. For each point of the biometric template's feature, the polynomial is evaluated, *(x,p(x))*. In order to increase the security of these calculated values, a large number of fake points are generated, which should not lie on the polynomial *P(x)*. At the end, the collection of genuine and fake points are named as the *vault*. In the authentication phase, assuming that the query of unordered biometrics is *x'*, with the use of Lagrange interpolation, *P(x)* is reconstructed using the genuine points are that determined from the query template *x'*. If *P(x)* could be reconstructed, its coefficient, forming the secret key *k'*, is corrected with the help of error correction codes, which will yield to the secret key *k*.

There are many researches in the literature about the fuzzy vault scheme that are based on fingerprint [31] [33] [38], iris [28] [39] [40] and face [29] [41]. Nevertheless, various attacks against the fuzzy vault scheme are discovered, such as brute-force attack [42], blended substitution attack [43], stolen key inversion attack [43] and correlation based attacks [44]. For instance, in [45], despite the fact that the authors have developed a fingerprint based fuzzy vault scheme, the proposed method includes information leakage. The problem happens when the authors add chaff points to the vault. These chaff points

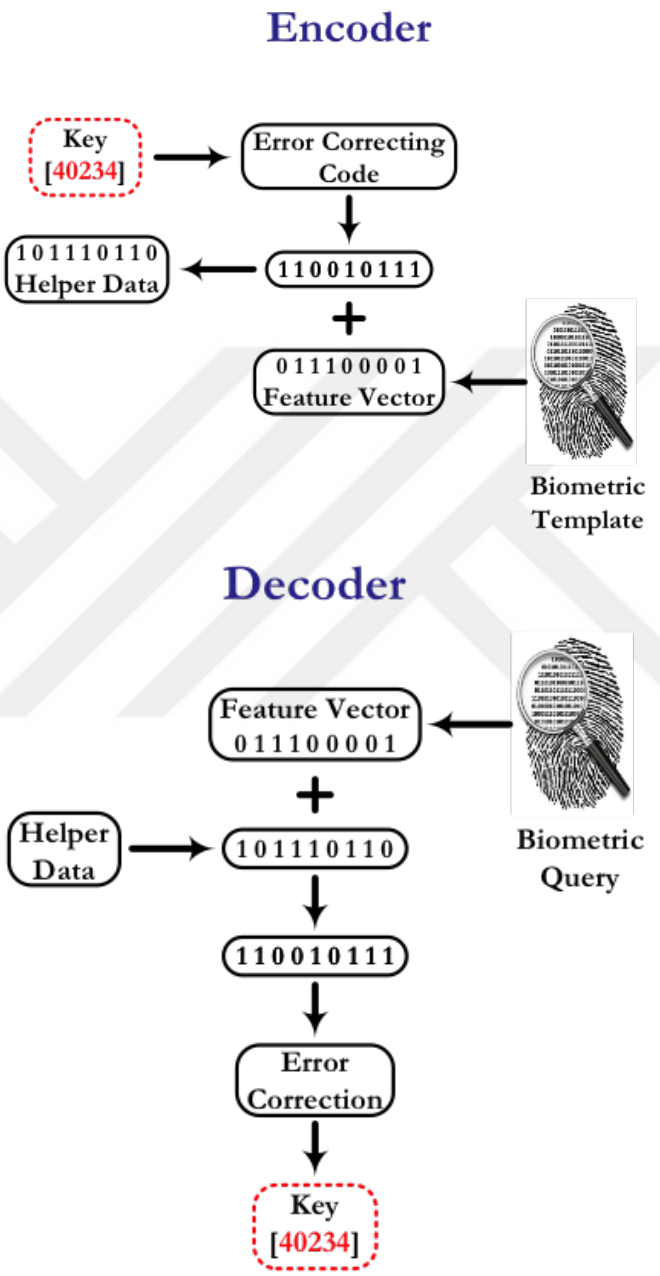

Figure 2.8: Fuzzy Commitment Scheme

are close to each other and far from the genuine points. Therefore, when the attacker finds two points that are close to each other and one of them is a chaff point, (s)he could make sure that the other one is a chaff point as well.

2.4 Cancelable and Noninvertible Biometrics

Nowadays, using the PINs, ID cards and passwords are no longer safe and adequate for identity verification since they could be easily stolen or shared, as discussed in Section 2. One of the solutions to this problem is using biometric characteristics in the authentication process. As mentioned in Section 2.1, physical characteristics are unique for each person, such as, fingerprint, iris and face; thus using them could be a reliable solution for the aforementioned problem. Nevertheless, biometrics based models could also cause some serious concerns [3]: (i) biometrics are not secret, (ii) if an attacker captures a biometric trait, it cannot be replaced easily and it will be lost forever, (iii) biometrics renewal is impossible, and (iv) the possibility of cross matching in biometrics database is high. To overcome these problems, *cancelable biometrics* [5] have been presented.

Cancelable biometrics is a concept in which the biometric template is protected through the combination of biometric feature replacement and security. Transforming or changing the biometric features or the biometric images prior to the matching process is the main idea of the cancelable biometric systems. The only thing that should be taken into account is that, during the cancelability process, the natural characteristics of biometrics should be preserved. Additionally, there are 3 important standards that a cancelable biometric system should include: (i) being specific and reusable, (ii) using a unidirectional transformation function, and (iii) possessing comparable performance [46].

In cancelable biometrics, instead of storing the original biometric template, first a transformation function is applied on the captured biometrics and then the result is stored in the database [3]. If this transformed version of the biometric template is compromised, then a new transformation function can be applied on the original biometrics. Therefore,

with this mentioned solution, the first three concerns about biometrics will be resolved. Moreover, for solving the cross matching problem, different transformation functions can be applied for different applications.

Cancelable biometrics can be classified into being (i) *invertible* and (ii) *noninvertible*. In invertible cancelable biometrics, if the transformed template is compromised, then the attacker can find the original biometric template of the user [25]. In noninvertible cancelable biometrics, a one way transformation function is used, which means that even if either of the transformation function and its parameters or the transformed template is compromised, obtaining the original biomterics is computationally infeasible. Therefore, we can make sure that the original biometrics could not be captured when we used noninvertible cancelable biometrics.

Cancelable noninvertible biometrics presents a solution to provide privacy for the user, in a way that, in the authentication phase, the user would not identify at all. The data that is produced by using noninvertible transformation guarantees that the original biometrics template will be protected [47]. As mentioned above, renewing someone's biometric is difficult and it is not like renewing PINs or passwords. However, cancelable biometrics allow a user to re-issue her/his biometrics. First of all, prior to the matching process, the original biometrics is mixed up and the new data is stored in the database. Secondly, if the data is compromised, then the mixing parameters can be changed and a new data can be produced instead of the compromised one. Figure 2.9 demonstrates the re-issuing of the original biometric data on cancelable biometric systems [48].

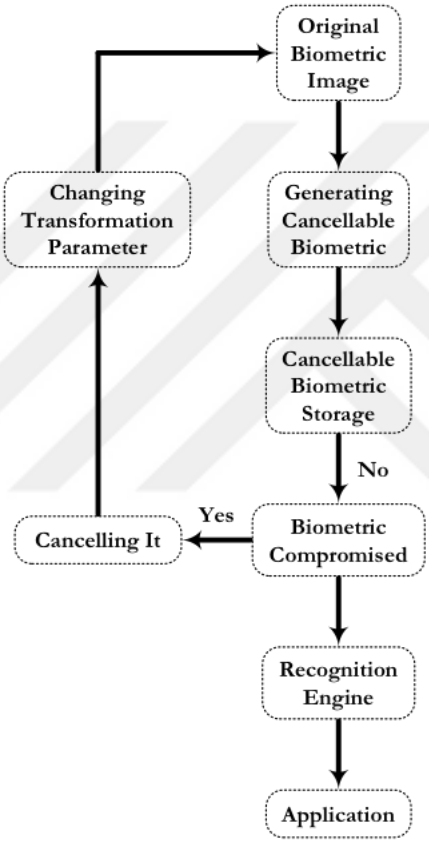

Figure 2.9: Re-issuing in Cancelable Biometrics Systems

3 *Related Work and Problem Statement*

In this section, we discuss the literature of the related works and we mention about the problem statement of this thesis.

3.1 Related Work

In a biometric system, biometric template protection is of great importance. Once a biometric template of a user is compromised, it could not be replaced. To overcome the above-mentioned problem, Jein et al. propose two approaches: (i) biometric cryptosystems, and (ii) feature transformation [49]. In biometric cryptosystems, the cryptographic key of the security protocol could be generated from the biometrics itself, or it could be binded to the biometrics. In both cases, only the helper data is stored in the database, while the key and the biometrics template are discarded. Therefore, the attacker can not find any information about the key and the biometrics from the stored helper data; so to reproduce a key from another biometric, it is really useful. Besides, Dodis et al. [37] present *fuzzy extractor* and *secure sketch*, which are two well-known approaches in biometric cryptosystem. *Fuzzy commitment* [26] and *fuzzy vault* [27] are examples of fuzzy extractor and key binding method, as mentioned in Section 2.3. On the other hand, in feature transformation, the data is transformed prior to the matching process and then it is stored in the database. The transformation function is a one-way function that makes recovering the original template from the transformed one hard.Thus, the original template stays safe.

Ratha et al. [3] recommend 3 types of noninvertible cancelable biometric transformation methods for fingerprint templates: (i) *Cartesian* transformation, (ii) *polar* transformation, and (iii) *functional* transformation, as shown in Figure 3.1. In *Cartesian* transformation, firstly, minutiae positions are normalized according to the singular point and then each fingerprint is partitioned into fixed sized rectangular cells. Finally, the positions of these generated rectangular cells are rearranged based on a mapping matrix, which is public information. In *polar* transformation, the polar coordinates of the minutiae are calculated based on the position of the core point, and then these calculated polar coordinates are divided into sectors. Finally, based on the translation key, and according to the changes of sector positions, positions of the minutiae are changed. At the same time, the minutiae angles are changed based on the variation of the sector positions before and after the transformation. As a last method, in the *functional* transformation, the authors propose to apply a local value smoothing function on the captured biometric template using a random key. Although the authors mentioned that their functional transformation is noninvertible, it can be shown that if an attacker uses a correlation attack, (s)he can find the original biometric templates.

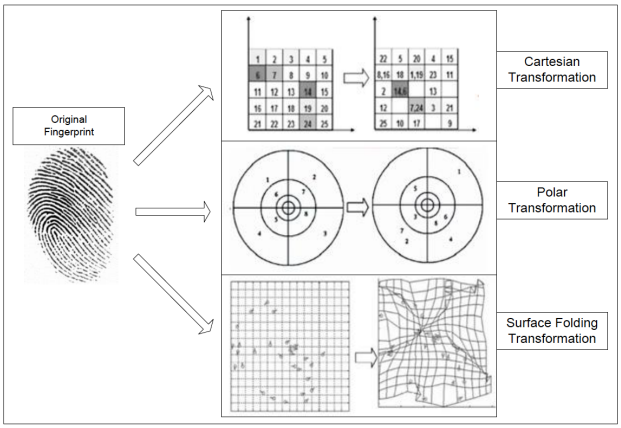

Figure 3.1: Cancelable Noninvertible Transformation Functions [3]

Several research have showed that the proposed noninvertible cancelable methods in [3] are not resistant against correlation and brute-force attacks [50] [51]. Alternatively, Jin et al. [52] propose BioHash, which is a cancelable and noninvertible biometrics approach. In this method, first, the authors extract the most specific characteristics of the

biometric features, and then, based on a user specific random matrix, these characteristics are placed on a set of randomly generated orthogonal directions. Besides, Farooq et al. [53] propose an approach which could be useful for implementing BioHash. In their model, the authors convert fingerprint minutiae into a binary string area. After that, these binary number representations are transformed to anonymous representations according to a unique personal key. The authors claim that the proposed transformation method is noninvertible, and also the template is cancelable. They mention that the template could be canceled just by entering a various new key. Notwithstanding, Nandakumar et al. [49] discuss that BioHash is indeed invertible. On the other hand, Chikkerur et al. [54] propose another method to generate cancelable fingerprint templates. In this method, from the fingerprint image, small pieces of minutiae are extracted and then without changing the distance between each of these small pieces of minutiae, the authors transform them to projection matrices. Unfortunately, the accuracy of this method is shown to be poor [48].

In the below paragraphs, we describe the two valuable related works that our proposed key agreement protocol is based upon,a biometric cryptosystem and a feature transformation method.

Secure key agreement using pure biometrics (SKA-PB)

Akdoğan et al. [7] propose SKA-PB, secure key agreement protocol using pure biometrics, in which the keys are generated using fingerprint's minutiae points that are directly extracted from the fingerprint images, without any other helper data. SKA-PB has two phases: (i) enrollment phase, and (ii) verification phase.

In the enrollment phase of the SKA-PB protocol, first of all, the user provides three fingerprint images of the same finger. Minutiae values, *(x,y,type)*, are extracted from these fingerprint images. The authors produce the fingerprint template based on the minutiae set, according to the following method. Firstly, a distance threshold, T_{dist} is set up and some representative minutiae points are found according to that value. The representative is a minutia point which is utmost T_{dist}-away of any other minutiae that has the smallest

y- coordinate value. To generate the final minutiae template, the server puts these three minutiae templates on top of each other and if a minutia point shows up at least in two out of three templates, it is added to the final minutiae template. There is also a neighborhood relation defined in the SKA-PB protocol, which is for the coordinates x_i and y_i of a particular minutia, all the points that are in the range of [x_i-T_{dist},x_i+T_{dist}] and [y_i-T_{dist},y_i+T_{dist}] are considered as a neighborhood minutiae. After each points' neighborhoods are found, the minutia values x_i, y_i and $type$ of each particular point and its neighbors are concatenated separately, and hashed as follows, $H^1(x_i||y_i||type)$. Then, the authors store these hashed values as each user's template in the server side.

In the verification phase of the SKA-PB protocol, three different fingerprints of the same finger are captured and fingerprint's minutiae are extracted like in the enrollment phase. As mentioned above, according to the predefined distance threshold T_{dist}, some representative minutiae are selected and the other minutiae are mapped to their representatives. Likewise, the most reliable minutiae, are calculated as in the enrollment phase. In the following, to keep the genuine points safe and secure, the authors add some chaff points according to a predefined strategy: fake points should also preserve T_{dist}-neighborhood relation; in other words they should be T_{dist}-away from all other points, both genuine and fake points. They developed this method to decrease the risk of information leakage to the attacker. Therefore, like in the enrollment phase, the concatenation of the minutiae values x_i, y_i and $type$, for both genuine and fake points, are computed. Then, the authors calculate the single and double hash values of these minutiae values, $(x_i||y_i||type)$. In the following, they transmit the double hashes and the ID's of each user to the server. At this point, the server and the user try to find a common set of minutiae points. Then, a similarity score is calculated and if the number of common minutiae points is above the predefined similarity score, then the user will be accepted; otherwise, the server will just reject the user. Since the SKA-PB protocol runs in a round manner, if in any round the user gets rejected, the protocol will start a new round.

Cancelable fuzzy vault with periodic transformation

Dang et al. [8] present a cancelable and noninvertible fuzzy vault model that uses a periodic transformation function on the face features of the users. The proposed scheme includes enrollment and authentication phases. In the enrollment phase, first the feature vector X is extracted from the user's face images. After that, the periods P of the feature vector X are calculated. Then, after these periods are concatenated, they are hashed to be kept safe and stored in the database. In addition, for the proper recovery of the periods P, in the authentication part, the authors calculate Reed-Solomon error correction code for those P values. Then, a periodic transformation function, which is a sine transformation, is applied on the extracted face feature vectors, which is called Y. In the following, the authors construct the fuzzy vault polynomial using a randomly generated key as the coefficient of the mentioned polynomial. The hash value of this randomly generated key is stored in the database to be used during the matching process later in the authentication phase. Next, to produce the genuine points for the fuzzy vault, the authors apply the above-mentioned polynomial on the transformed feature vector Y. In addition, they introduce a set of fake points to the fuzzy vault. All of these points, i.e., both the genuine and the fake points, are stored in the fuzzy vault database.

In the authentication phase, the user's face image is first recorded. Then, the face related feature vector X' and the corresponding period vector P' are extracted. Therefore, using the Read-Solomon error correction code, the calculated period vector P' is corrected. Then, the hash of the period vector P' is compared to the hash of the period vector P, and if they match, a periodic transformation function, the sine transformation, is applied on the feature vector X' to find the feature vector Y'. Otherwise, authentication fails. If the transformed feature vectors Y' and Y have enough overlap with each other when the fuzzy vault's decoding step is performed, then the key will be recovered correctly. After that, the hashed value of the recovered key is compared with the randomly generated key. If they match, the user is verified; otherwise, the authentication fails.

3.2 Problem Statement

In most of the key binding or key generation based existing methods proposed for key agreement, helper data is utilized along with pure biometrics, which complicates the overall system. Although Akdoğan et al. [7] proposed a method that generates the key directly from the captured biometrics without the use of a helper data, it suffers from being lack of cancellability and noninvertibility properties. On the other hand, Dang et al. [8] proposed a method to overcome this shortage, which makes the fuzzy vault scheme cancelable and noninvertible. The authors apply a periodic transformation function on the face features, which is a one dimensional vector, and they use the Reed-Solomon error correction code in their proposed method. As discussed by Akdoğan et al. [7], the SKA-PB key agreement protocol has larger attack complexity in comparison with the fuzzy vault scheme. Also, the SKA-PB protocol is really stronger against brute-force attacks than the fuzzy vault approach. Therefore, to overcome the above-mentioned problems and to provide a more secure environment, in this thesis, we combine the idea of cancelable noninvertible transformation with the proposed SKA-PB protocol, and we present SKA-CaNPT, a secure key agreement protocol that uses cancelable and noninvertible biometrics.

4 *Secure Key Agreement Protocol using Cancelable and Noninvertible Biometrics based on Periodic Transformation*

In this section, we provide a detailed explanation of our proposed protocol SKA-CaNPT: Secure Key Agreement Protocol using Cancelable and Noninvertible Biometrics based on Periodic Transformation, which is given in Algorithm 1 and visualized in Figure 4.1. As a proof of fact, we apply our SKA-CaNPT protocol on fingerprints. Our proposed approach is composed of two phases: (i) enrollment phase, and (ii) verification phase, each of which is explained in the below subsections one by one as it is shown in Figure 4.2. In the enrollment phase, minutiae are extracted from the captured biometric data and they are transformed with our periodic transformation algorithm. After that, the biometric template of the user is calculated through the process of finding the most reliable minutiae and the constructed templates are stored in the server database so that it can be used in the matching process later. Moreover, in the verification phase, the most reliable minutiae is calculated with the user's newly registered fingerprints, after feature extraction and periodic transformation operations are performed. In this phase, the template is composed of genuine minutiae and the generated chaff points. In the following, the user and the server try to find a common set of minutiae and a similarity score is calculated according to the number of the found common minutiae. Depending on this the calculated similarity score, our SKA-CaNPT protocol decides whether to reject or to accept the corresponding user. If the user is accepted, then the server and the user make an agreement on a symmetric key. Otherwise, the user is rejected and the protocol will srart from beginning.

Algorithm 1 Template Generation Algorithm

INPUT: FP_1, FP_2, FP_3, r
OUTPUT: G_s^T

G_s^{init} = *ExtractMinutiae*(FP_1, FP_2, FP_3)
for $i = 1 : |G_s^{init}| - 1$ **do**
    $m_1 = G_s^{init}(i) = (x_i, y_i)$
    $m_x = x_i \div 2\pi$
    $m_y = y_i \div 2\pi$
    $\alpha_i = x_i - m_x 2\pi$
    $\beta_i = y_i - m_y 2\pi$
    ${x'}_i = arcsin(r) - \alpha_i$
    ${y'}_i = arcsin(r) - \beta_i$
    $m_1 = (x'_i, y'_i)$
    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

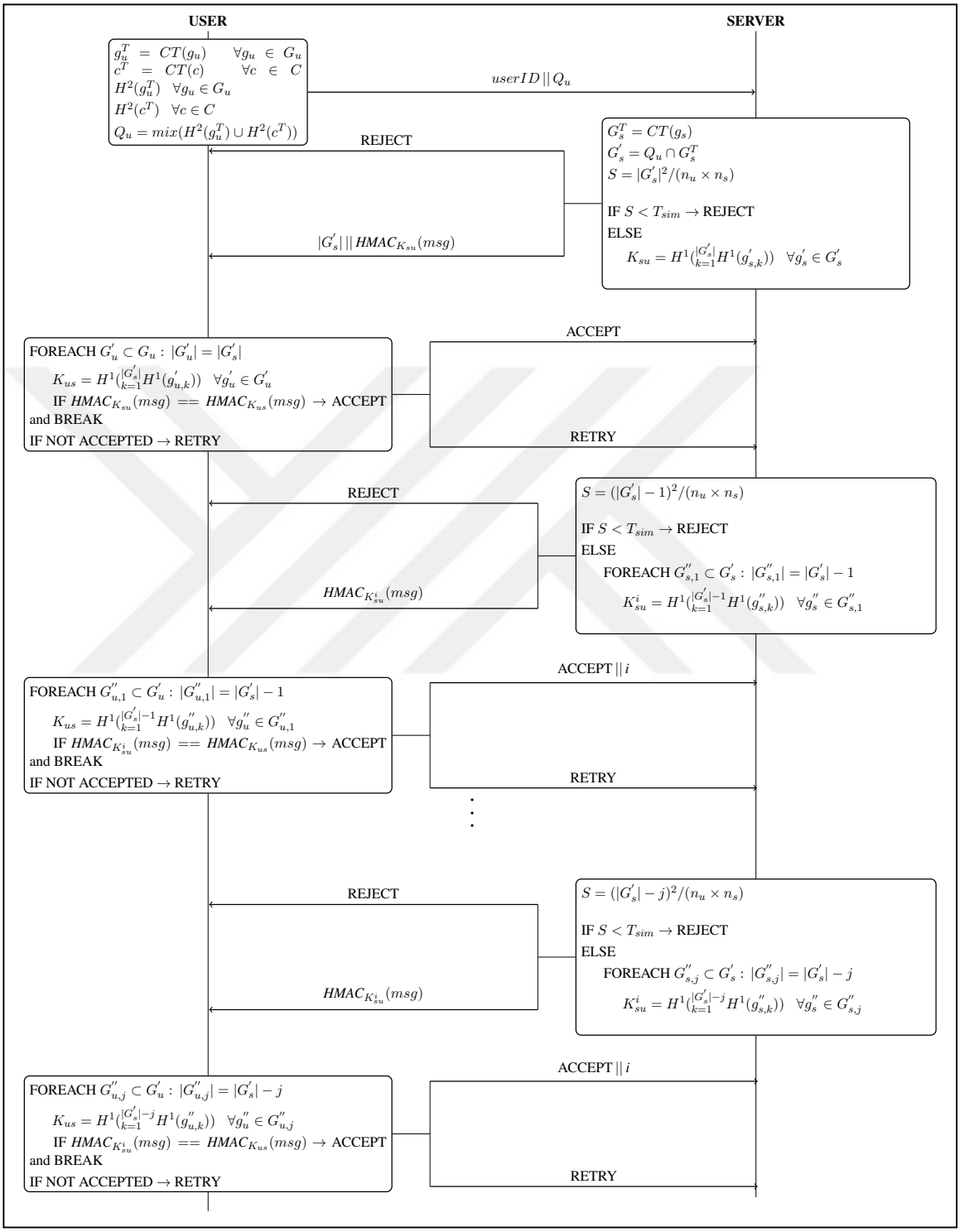

Figure 4.1: Our proposed cancelable and noninvertible secure key agreement protocol

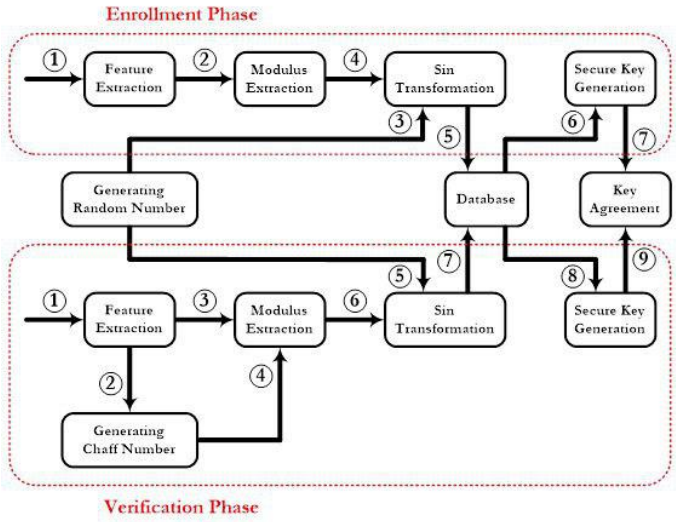

Figure 4.2: Our proposed cancelable and noninvertible secure key agreement protocol

Our approach basically combines the SKA-PB protocol that is proposed by Akdoğan et al. [7] and the idea of using a transformation function that is proposed by Dang et. al [8]. The periodic transformation function that we utilize has two important properties. The first one is similarity reservation, which means that the similarity of distances between the original templates and the transformed templates should be preserved. This property ensures that the two transformed fingerprint feature vectors will be similar to those of the two original fingerprint feature vectors. The second important property is the noninvertibility, which means that the original fingerprint templates should be protected against compromise forever. Table 4.1 gives the definition of symbols that are used in our SKA-CaNPT protocol.

Table 4.1: Symbols used in SKA-CaNPT Protocol Definition

Symbol	***Description***
FP	Fingerprint
x	Location of minutia on x-coordinate
y	Location of minutia on y-coordinate
type	Type of a minutia
x'	Location of minutia after transformation on x-coordinate
y'	Location of minutia after transformation on y-coordinate
r	A random number between $[-1, 1]$
m_x	Modulus number of x-coordinate
m_y	Modulus number of y-coordinate
$CT()$	Cancelable and noninvertible transformation function
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 ($i \geq 0$)
G_s	Set of genuine minutiae on the server side
G_s^T	Transformed set of genuine minutiae on the server side
G_u	Set of genuine minutiae on the user side
g_u^T	Transformed set of genuine minutiae on the user side
C	Set of fake minutiae on the user side
C^T	Transformed set of fake minutiae on the user side
Q_u	Set of shuffled $\left(H^2(g_u^T) \cup H^2(c^T)\right)$ s.t. $g_u^T \in G_u^T$ & $c^T \in C^T$
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 ($i \geq 0$)
$K^i_{(us,su)}$ — *us*	by the user to communicate with the server
$K^i_{(us,su)}$ — *su*	by the server to communicate with the user
HMAC$(\cdot)$	Keyed-Hashing for Message Authentication
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

4.1 Enrolment Phase

In the enrolment phase, the user is asked to provide 3 fingerprint images of her/his same finger, FP_1, FP_2, FP_3, using which the minutia are extracted, as it is performed in SKA-PB [7]. The stored information per minutiae are composed of 3 attributes: locations of the *x*-coordinate and the *y*-coordinate, and the *type* of the minutia. Each minutia could categorize to two different types: *end* and *bifurcation*. If the ridge in a point splits into two ridges, then it is called *bifurcation*, and if it terminates, then it is called *end*. In other respects, for adding the *cancelability* and *noninvertibility* properties, we are following the below defined steps [8]

(i) Modulus Extraction: We use the sine function as our transformation function. Due to the fact that the sine function is a periodic function with a period of 2π, we find the modules of each minutia coordinate separately, i.e. we calculate the modulus for the *x*-coordinate and the *y*-coordinate of each minutiae. Therefore, for each minutiae coordinate x_i and y_i, we have a definition as given in Equation 4.1, in which m is the modulus number of x and y. Hence, the modules of these coordinates m_{x_i} and m_{y_i} are calculated as in Equation 4.2.

$$\begin{aligned} x_i &= \alpha_i + m_{xi}2\pi \\ y_i &= \beta_i + m_{yi}2\pi \end{aligned} \tag{4.1}$$

$$\begin{aligned} m_{xi} &= [x_i \div 2\pi] \\ m_{yi} &= [y_i \div 2\pi] \end{aligned} \tag{4.2}$$

In the following you can find modulus extraction example:

$$\begin{aligned} x_i &= 272 \\ y_i &= 87 \\ m_{xi} &= [272 \div 2\pi] = 43 \\ m_{yi} &= [87 \div 2\pi] = 14 \end{aligned}$$

(ii) Sine Transformation: After the pmodulus extraction operations are completed, we transform the minutiae coordinates x_i and y_i as defined in Equation 4.4, using the analogy behind Equation 4.3. We are using the sine periodic transformation function as our cancelable noninvertible transformation function, which is a many-to-one function. It means that each one of the x_i and y_i coordinate values are transformed to some particular x_i' and y_i' coordinate values in the transformed domain, while for each specified x_i' and y_i' coordinate values in the original space, we could find a lot of x_i and y_i coordinate values. Having many different correspondent values with the particular x_i' and y_i' coordinates, the above-mentioned discourse can be accepted as a proof for the noninvertibility property of the sine transformation function.

$$\begin{aligned} sin(x_i + x_i') &= r_i \\ sin(y_i + y_i') &= r_i \end{aligned} \tag{4.3}$$

$$\begin{aligned} x_i' &= arcsin(r_i) - \alpha_i \\ y_i' &= arcsin(r_i) - \beta_i \end{aligned} \tag{4.4}$$

After the modulus extraction and sine transformation operations are completed, each fingerprint's template is produced according to its list of minutiae. For template generation, the groups of minutiae are categorized with respect to a given distance threshold, T_{dist}, and a representative minutia is selected for each group of minutiae, as described by Akdoğan et al. [7]. Then, the minutiae are quantized according to the selected representative: the minutiae having the same minutiae type and being utmost T_{dist}-away to any other minutiae are mapped to the one minutia having the smallest *y*-coordinate value. Thereafter in order to find the most reliable minutia, the server puts these fingerprint templates on top of each other. If a minutia shows up in at least two out of three fingerprint's templates, then as described in SKA-PB [7], it is counted as a reliable minutia, and it will be added to the final fingerprint's template. Finally, the concatenation of $(x_i||y_i||type)$ are calculated for each representative minutiae and its T_{dist}-neighborhood, as defined by

Akdoğan et al. [7], and their hashed values, $H^1(x_i||y_i||type)$, are stored in the server side, separately for each user. This process is performed only in the server side.

4.2 Verification Phase

In the verification phase, like in the enrollment phase, 3 different fingerprints of the same finger are acquired from the user, as discussed by Akdoğan et al. [7]. The minutiae are extracted from these fingerprint images and for adding the cancelability and noninvertibility properties to the extracted minutiae, we apply our periodic transformation function, i.e. the sine function, on these minutiae, as it is done in the enrollment phase. After that, the quantization operation is performed to designate the representative minutiae, and the most reliable minutiae are found, again, as in the enrolment phase.

Differently, in the verification phase, in order to mask the genuine minutiae, we generate 10*G_u fake minutiae randomly and we transform those randomly generated fake minutiae with our cancelable noninvertible transformation algorithm. The fake minutiae points should preserve the T_{dist}-neighbourhood relation as well. In other words, since our genuine minutiae points are T_{dist}-away from other genuine minutiae points, the fake minutiae points should be T_{dist}-away from all the other minutiae points, in order to decrease the possibility of information leakage to the attacker and in order not to be distinguishable from the genuine minutiae points. Afterwards, as explained in SKA-PB [7], the features of these minutiae points, x-coordinate, y-coordinate and $type$, are concatenated and two different hash values are calculated as follows: $H^1(x_i||y_i||type)$ and $H^2(x_i||y_i||type)$. The single hashes are utilized for generating the key and the double hashes are utilized for verification purposes, as described below.

As discussed by Akdoğan et al. [7], for finding the common genuine minutiae, the following steps are carried out. First of all, in the user side, double hashes of the minutiae points are concatenated with each user's ID, and then they are transmitted to the server. Secondly, the server compares each of these double hashed values with the double hashes of its correspondent user's minutiae points. In addition, if the user's genuine minutiae are

in the T_{dist}-neighbourhood of the server's minutiae, then they will be counted as common genuine minutiae as well. For distinguishing the fake minutia from the genuine minutia, a comparison is performed between the minutiae of the user and the minutiae of the server, and a similarity score is computed as given in Equation 4.5, where n_{com} is the number of common minutiae, n_s is the number of genuine minutiae on the server side, and n_u is the number of genuine minutiae on the user side.

$$S = \frac{n_{com}^2}{n_u + n_s} \times 100 \tag{4.5}$$

The key agreement process starts if the calculated similarity score is above a predefined acceptance threshold, T_{sim}, as defined in SKA-PB [7]. When this is the case, firstly, the server calculates the single hash of all common minutiae, $H^1(G_s)$, and then it calculates the double hash of them to generate the key, which the server and the user will use to secure their communication, K_{su}. In order to make sure that both the server and the user are using the same key, *HMAC* of a predefined message, *msg*, is calculated with K_{su} and it is transmitted to the user together with the number of common minutiae that the server has found, $|G'_s|$.

After that, according to the number of common found minutiae, the user tries each of the possible subsets of genuine minutiae with the equal number to generate the key. The user sends a positive acknowledgement if (s)he can verify the *HMAC*. Otherwise, s(he) tries to generate another key with another subset, until (s)he could verify the *HMAC* or all the other subsets are tested. The user will send a *RETRY* message if (s)he cannot verify the *HMAC* even all the subsets are tried.

As discussed in the SKA-PB protocol [7], if the user transmits a *RETRY* message to the server, a new similarity score is calculated in the server side using $|G'_s| - 1$ as the common minutiae count, and if it is above T_{sim}, the server generates all possible keys of all subsets of the common minutiae that the server found, which has the size equal to $|G'_s|-1$, and then it computes all *HMAC*s of the predefined message with respect to these keys and transmits them to the user. Therefore, the user sends a positive acknowledgement if (s)he can verify any of the transmitted *HMAC* values with any of the keys which is generated

based on any subsets of the genuine minutiae with the same size of $|G'_s| - 1$. Along with the positive acknowledgement, the user also transmits the index of the verified *HMAC*, i, to the server. Otherwise, another *RETRY* message will be transmitted to the server. In this case, the aforementioned process will repeat with $|G'_s| - 2$. The overall process stops at the j^{th} iteration if any of the *HMAC* values cannot be verified by the user or the protocol finds out that the similarity score of $|G'_s| - j$ is under the acceptance threshold, T_{sim}. At the very end of the protocol, the server and the user can agree on a symmetric cryptographic key if the user could verify any of the *HMAC* values. On the other hand, upon a request, the server and the user could start from the scratch, if the protocol stops without any key agreement.

In our SKA-CaNPT protocol, we utilize a cancelable and noninvertible transformation function, i.e. the sine function, to transform the user's original biometric feature vector to a secure transformed template. By this means, if the key is compromised by an attacker, cancelling the stored biometric template and generating a new one would be easy. To construct a new fingerprint template, a newly generated random vector r' will be replaced with the previous random vector r in the cancelable and noninvertible transformation algorithm, and then the key agreement protocol can be run with this newly transformed biometric template from scratch.

5 *Performance Evaluation*

For performance evaluation purposes, we have tested our SKA-CaNPT protocol with two datasets consisting of 30 and 292 different subjects. The first one retrieved from Verifinger sample database [55], each of which contains 8 images per finger. These fingerprints are captured by using Cross Match Verifier 300 at 500 ppi [56]. The second dataset fingerprint images collected using Papilon DS22N at 500 ppi [57] by TÜBITAK 114E557 project team from volunteers in Sabanci University. In this dataset we collect 10 different image from a finger. In addition, since we expect users to provide fingerprints with good quality, our dataset is small because large datasets include bad quality fingerprints as well. We use pre-aligned fingerprint images to test our protocol. As explained by Akdoğan et al. [7], the fingerprint images are aligned with Matlab R2015a, using their intensity values, and each fingerprint's minutiae are extracted using the Neurotechnology Biometric SDK 5.0 Verifinger [55]. In the first dataset, we use the first 3 images to produce fingerprint templates of the server side and the other 5 are used to generate the user side's fingerprint templates. Hence, each subject is tested $\binom{5}{3}$=10 times. In the second dataset, we use 3 first image to generate a template for the server side and we use the next 7 fingerprint images to generate fingerprint template in the user side. It means that each subject tested $\binom{7}{3}$=35 times. As we did not utilize registration on the system and we just used alignment for our fingerprint images, it could be a reason for some of our fails in our key agreement protocol. Moreover, we have also carried out impostor tests, in which we check each subject's template against all other subject's templates. In our SKA-CaNPT protocol, we have used SHA-256 hash function [22], as mentioned in Section 2.2.2, so all of the generated keys are 256 bits long.

5.1 Performance Metrics and Parameters

The most important performance measure of a biometric cryptosystem is the Error Rate, which is computed through Incorrect Key Generation Rate (IKGR) and Correct Key Generation Rate (CKGR). IKGR is the rate of falsely acceptance of impostor users, it means that the percentage of the impostor users which make an agreement with server about a key like they are genuine users and CKGR is the rate of truly accept of the genuine users, it means that the percentage of the server correctly make a agreement on a key with the genuine users.

On the other hand, randomness of the generated cryptographic keys is one of the most important concerns for key agreement/generation protocols. Here, Shannon's entropy [58], which is introduced by Claude E. Shannon, is used to measure the randomness, as given in Equation 5.1, where K_i represents the key's i_{th} bit and $P(.)$ is the probability function. According to this evaluation, the entropy value can be within the range of $[0, 1]$, in which 1 means maximum randomness.

$$H = -\sum_{i} P(K_i) \log_2 P(K_i) \tag{5.1}$$

Besides, distinctiveness of the generated cryptographic keys is another important concern for key agreement/generation protocols. The reason behind this is that, after each round of the protocol, a different key is required to be generated. Here, the distinctiveness of the generated keys are measured by the Hamming distance metric [59], which is presented by the Rihard Hamming. The Hamming distance defines the difference of the element positions in two strings of equal length, as shown in Figure 5.1.

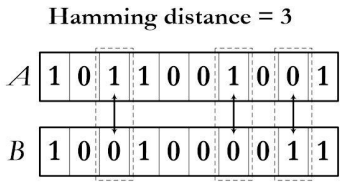

Figure 5.1: Hamming Distance Example

5.2 Verification Results

A similarity score is calculated for each test using Equation 4.5, which is defined in Section 4.2. For each subject, the maximum, minimum and average scores of the system are calculated. Each of these scores are computed using a different acceptance threshold value; thus the IKGR and CKGR of the system are calculated according to different thresholds. The results show that when the acceptance threshold is set to the maximum score of the system, the best result is obtained. We find the average number of attempts to make an agreement on the key which in the first dataset that is *3.068* and in the second dataset it is *12.44*. In addition, in 16 over 30 subjects the correct key is generating in the first attempt which means in 67.46% and in 197 of 292 subjects of the second dataset in the first attempt the correct key generated which means in 53.3%. Moreover, in the first dataset in one subject from 30 subject the server and the user could not make an agreement on a key. Therefore, the success rate according to the maximum score threshold is *96.67%* for the first dataset. In the second dataset any key agreement could not occure in 1 subject out of 292 sunjects. Thus, the second dataset' success rate is *99.65%*

Figure 5.2 illustrates the percentages of IKGR and CKGR of the first dataset when the threshold value is set to the maximum score of the system. As shown in this figure, the optimum error rate of our SKA-CaNPT protocol achieves when the threshold set to 24.762 with IKGR *0.67%* and CKGR *96.67%*. The IKGR and CKGR percentages of the second database is shown in the Figure 5.3. IKGR is *0.95%* and CKGR is *90.41%* when the acceptance threshold set to 22.22. For the biocryptographic authentication system mentioned values are absolutely sufficient.

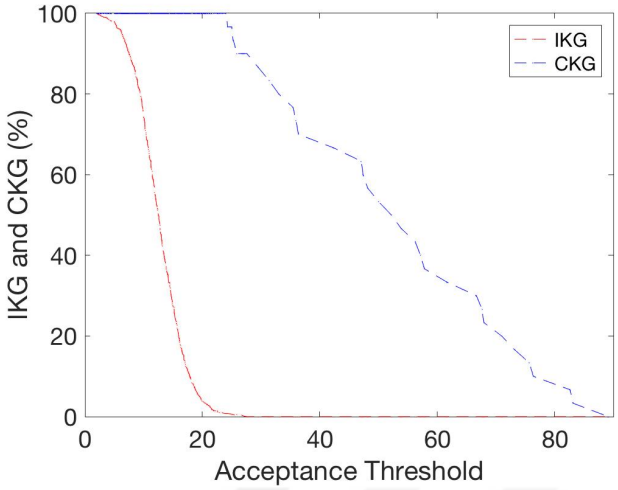

Figure 5.2: IKGR and CKGR of the first dataset when threshold is maximum score

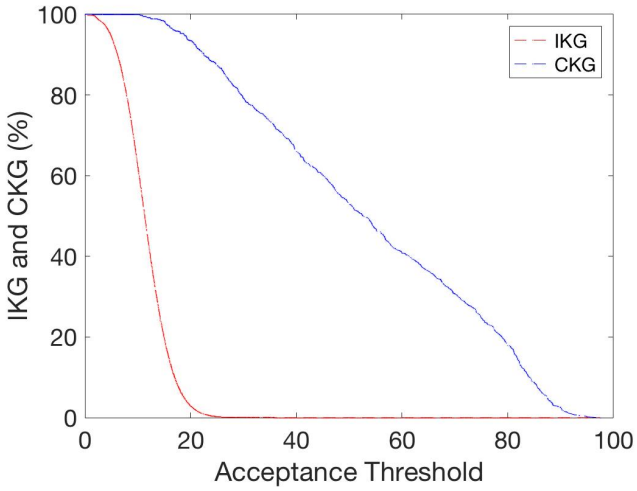

Figure 5.3: IKGR and CKGR of the second dataset when threshold is maximum score

5.3 Security Analysis

In this section, we first specify our threat model. Then, we analyze the resistance of our protocol against brute-force, impersonation and replay attacks. Finally, the qualification of the keys generated through our SKA-CaNPT protocol is evaluated according to the Hamming distance and entropy metrics.

Threat Model

The attacker's main purpose is as follows: (i) eavesdropping to find the key exchanged between the server the and users, (ii) impersonating a genuine user, and (iii) trying to find the original biometric template of a user. Since we are not assuming any secure channel, the attacker can access to all of the protocol messages, including the hash values and the HMACs which are transmitted between the user and the server. So, it is possible that the attacker can learn the number of minutiae which are used to generate the key. Therefore, using this value, the attacker can try the passive mode brute-force attack to guess the key. Also, in the active mode, the attacker can apply a replay attack to impersonate a genuine user. However, in the following paragraphs, we discuss how our protocol is resistant against these types of attacks.

Resistance Against Brute-force Attacks

A brute-force attack is always an option for an adversary; (s)he can try all possible combinations of the keys. In fact, this attack is infeasible against our SKA-CaNPT protocol since we are using 256 bits long keys in our protocol. Hence, below, we define a more intelligent brute-force attack by using protocol messages.

All possible minutiae locations and their types are generated by this attack. Although in our SKA-CaNPT protocol, we first transform the minutiae points prior to message exchange, Due to the similarity distance preservation property of our cancelable noninvertible transformation function, distances among the two minutiae points remains the same. Therefore, since the maximum distance between two minutiae points is 512 for the first dataset and 850 for the second dataset and there are 2 types of minutia, *end* or *bifurcation*, the attacker should generate $512 \times 512 \times 2 = 2^{19}$ points and $850 \times 850 \times 2 = 2^{21}$ respectively for the first and second database for a brute-force attack and apply the hash function on them once and twice. However, as mentioned before, we do not assume a secure channel and thus the attacker have access to Q_u,which is the list of transformed

fake and genuine minutiae. For this reason, the search space of the attacker reduces to $|Q_u|$. Nevertheless, as shown below, our SKA-CaNPT protocol is still secure against brute-force attack.

As mentioned above, Q_u and n_{com}^{key}, the number of minutia for generating the key, are the information that the attacker has. The attacker should try all the subsets of Q_u with the size of n_{com}^{key} for finding all the generated keys. So, the attack complexity can be calculated with Equation 5.2.

$$att_c = \binom{|Q_u|}{n_{com}^{key}} = \frac{|Q_u|!}{n_{com}^{key}!(|Q_u| - n_{com}^{key})!} \tag{5.2}$$

If we find the attack complexity after each key agreement by using Equation 5.2 and we compute the average of the attack complexity, we will have the overall attack complexity of the system. With this calculation, the attack complexity of our SKA-CaNPT protocol is 94 bits for the first dataset, which means that 2^{94} hash and HMAC verifications are required. Moreover, for the second database the attack complexity is 118 bits which HMAC verification and hash which is required is 2^{118}. Juliato et al. [60] discuss that one block of HMAC's computation takes 0.8977 microseconds even with a particular hardware implementation. Hence, according to the aforementioned complexity, it will take about 5.6×10^{14} years in the first dataset and 9.46×10^{21} years in the second dataset for an attacker to attack our SKA-CaNPT protocol. Therefore, we can claim that our protocol is sufficiently secure against intelligent brute-force attacks.

Resistance Against Replay and Impersonation Attacks

Impersonating a genuine user to find the correct key is referred to as replay attack. In replay attacks, the attacker replays the previously transmitted messages between the server and the victim user. In order to find the generated key effectively, the attacker needs to know the genuine minutiae; otherwise, (s)he must check all the combinations of Q_u,

which is the set of the mixed and hashed genuine and fake minutiae points. Since the transformed version of both genuine and fake minutiae points are listed in Q_u, genuine minutiae points are indistinguishable for the attacker, so (s)he should try all the possible combinations of Q_u using Equation 5.2. Therefore, the complexity of the replay attack is the same as that of the brute-force attack's complexity.

In addition, if the attacker tries to generate the key with applying his/her own figerprint, (s)he will be successful in 0.77% of the cases in the first dataset and ... of the cases in the second dataset since it is the IKGR performance of our protocol. It is important to note here that it is a general problem in all biometrics systems and it is not just specific to our method.

Randomness of the Agreed Keys

Our SKA-CaNPT protocol generates 300 keys from 30 different subjects in the first dataset. Figure 5.4 illustrates the entropy values for these keys. In this figure, each point indicates the entropy of each individual key value, which is referred with a different key ID. Randomness of a key has a direct relation with the entropy value. As much as the entropy value becomes close to 1, the key gets more random. This figure shows that the entropy of at least 84% of the keys are greater than 0.994 and the entropy of all of the keys are greater than 0.9745, which indicates a really good randomness. These keys are generated with the help of hash functions which are applied to the common minutiae. Since hash functions introduce some randomness to the input string, we were already expecting a high randomness value for these results. Hence, we also calculated the entropy of the concatenated common minutiae ($x||y||type$), which is shown in Figure5.5. In this figure, the entropy of the 90.66% of the keys are above 0.98, which is a bit less than the entropy of the generated keys; but still they provide enough randomness.

The second dataset includes 292 subjects which we generate 35 keys per subject, it means SKA-CaNPT generate 10220 keys. The entropy values of the keys shown in Figure5.6.In this dataset approximately 99.3% of the keys have entropy values more than 0.98

and 100% of the keys have entropy which is grater than 0.95, which shows a sufficient randomness. As we mentioned before, we are applying the hash function on the concatenated common minutiae ($x||y||type$) for generating the keys which hash functions add a high randomness to these keys. Therefore, in Figure5.7 the entropy of the concatenated common minutiae is shown. The entropy value is 0.96 for 92.25% of the concatenated common minutiae. Since we have a big dataset, having a value around 0.65 is normal. Therefore, concatenated common minutiae have a really good randomness.

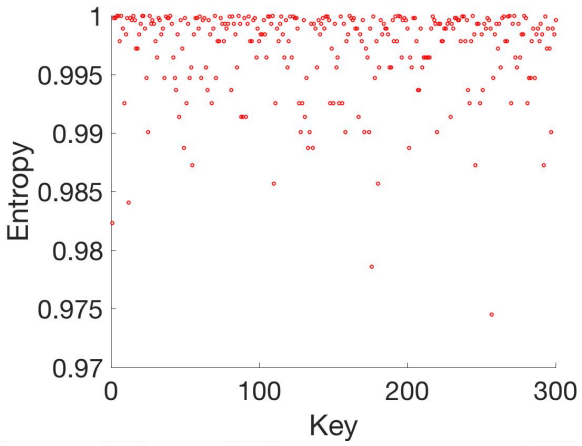

Figure 5.4: Entropy values of the generated keys for the first datasat

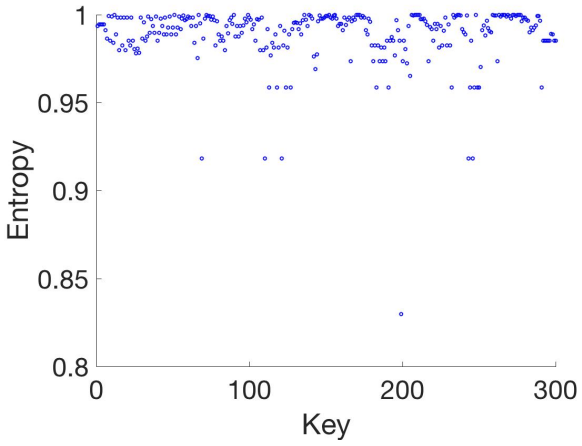

Figure 5.5: Entropy values of the concatenated of the transformed minutiae in the first datasat

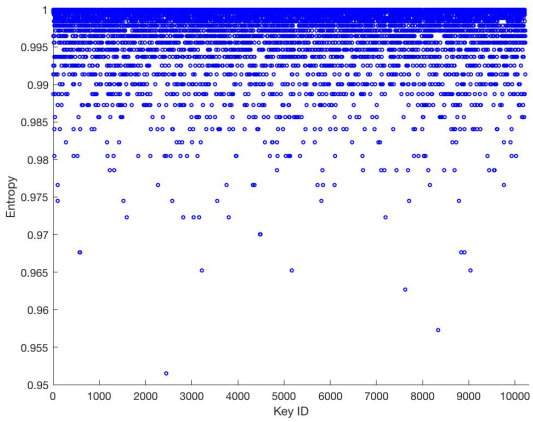

Figure 5.6: Entropy values of the generated keys for the second datasat

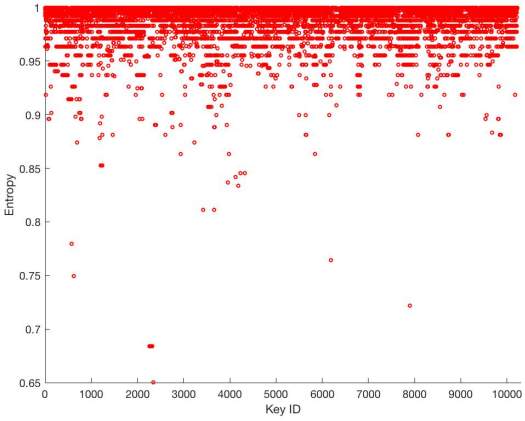

Figure 5.7: Entropy values of the concatenated of the transformed minutiae in the second datasat

Distinctiveness

Due to the intra-person distinction discussed in Section 2.3, we cannot expect to generate the same key after each key agreement run. Actually, generating the same key is unpleasant for us. The reason behind this is that if we generate the same key and if the key is compromised in one of the sessions, it will increase the risk of compromising messages from the other sessions as well, which is a concern for the confidentiality of the system. Indeed, generating different keys in each key agreement run is important for us. As discussed by Akdoğan et al. [7], according to the quality of the fingerprint scanner,

the finger's pressure on the scanner, acquisition enviornment and etc., the order of the minutiae and their quality can change. The above-mentioned situation can have either a positive or a negative impact on the key agreement process, depending on the application.

The Hamming distances are calculated to find the differences between the keys of the same user which are generated from the transformed minutiae, in different key agreement protocol runs. In the first dataset, 256 bit keys possess an average Hamming distance of approximately 120-130 bits for each different user. Moreover, as we can see in Figure 5.8, the minimum distance is equal to 103. Additionally, in the second dataset, the average Hamming distance of 256 bit keys for each different user possess approximately 120-130 bits, as shown in Figure 5.10, in which the minimum distance is 96 bits.

In addition to the above-mentioned calculations, we have also computed the average Hamming distance between the keys which are generated from the transformed minutiae of different users in order to show that different users' keys are also different from each other. Figure 5.9 illustrates that in the first dataset, the average Hamming distance changes between 120-135 bits. Besides, in the second dataset, the average Hamming distance changes from 120 to 135 bits, as shown in Figure 5.11. Hence, we can conclude that the average Hamming distance among different users' keys is close to the average Hamming distance among the same user's keys, which means that it is indistinguishable that which of the two keys belong to different users and which ones belong to the same user.

While generating the keys, we apply a hash function on the concatenation of the minutiae points. Due to the hash function's property, small change in the minutiae concatention causes a really huge difference on the hash values. Therefore, we need to calculate the Hamming distance of the minutiae concatenation to find out the level of unpredictability of our protocol. However, we compute the attack complexity of the *fingerprint similarity attack* over the minutiae cancatenations. The length of the minutiae concatenations vary since in each run of the protocol, the number of found common miutiae is different. In this attack, the imposter tries to guess the concatenation of a geniune minutia's length, before the hash function is applied on it, according to his/her fingerprint minutiae, by changing the bits of the minutiae concatenations. If we assume that the Hamming

distance of the vitims's minutiae concatenation and the imposter minutiae concatenation is clear for the imposter and (s)he knows the length of the minutiae concatenation, we have three different cases: (i) the imposter's minutiae concatenation is the same as the victim's minutiae concatenation, (ii) the imposter's minutiae concatenation is longer than the victim's minutiae concatenation, and (iii) the imposter's minutiae concatenation is shorter than the victim's minutiae concatenation. In each on these cases, the bit strings are considered equal-length. The quantity of the combination of the locations which are different shows the attack complexity. In addition, in the third case, the imposters's minutiae concatenation bits and the victims minutiae concatenations bits are important in the attack complexity, since the imposter should find them as well. The attack complexity is calculated for each protocol run. In addition, we find the median of each person's attack complexities to learn about the average case behaviour. As a consequence, the overall complexity of fingerprint similarity attack becomes 2^{251} for the first dataset, and 2^{252} for the second dataset. This amount of complexity is completely sufficient for preserving the SKA-CaNPT protocol from fingerprint similarity attack.

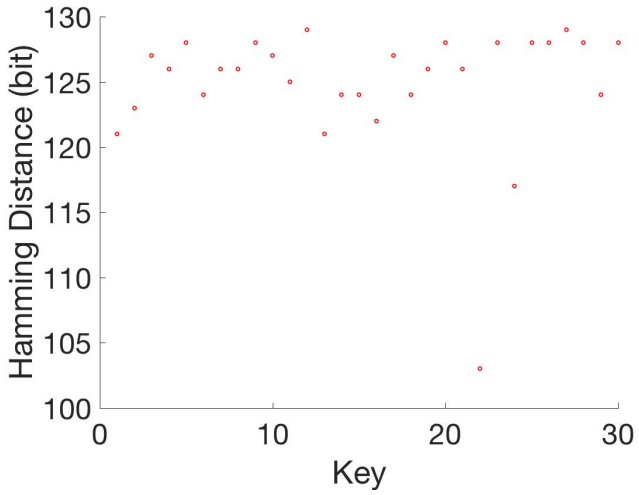

Figure 5.8: Average Hamming distances of the same user's keys of the 1^{st} Dataset

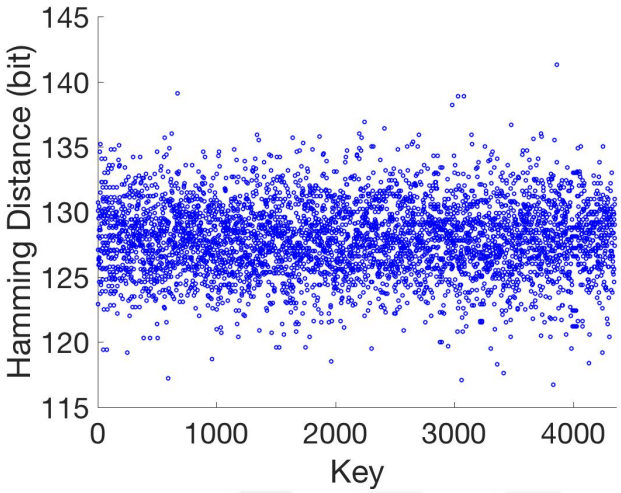

Figure 5.9: Average Hamming distances of the different user's keys of the 1^{st} Dataset

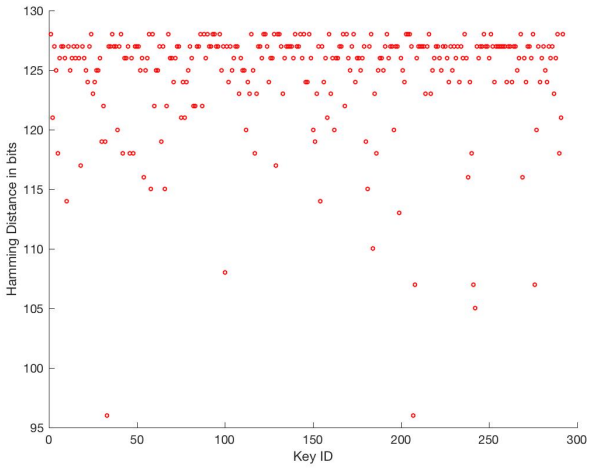

Figure 5.10: Average Hamming distances of the same user's keys of the 2^{nd} Dataset

Cancelability and Noninvertibility of Fingerprints' Template

In order to check the cancelability property of our protocol, we checked the Hamming distance between the keys which are generated from the transformed minutiae with different random numbers. As we can see in the Figure 5.12 and Figure 5.13, the average Hamming distance of the first and second datasets are varied between 120-135 bits. As

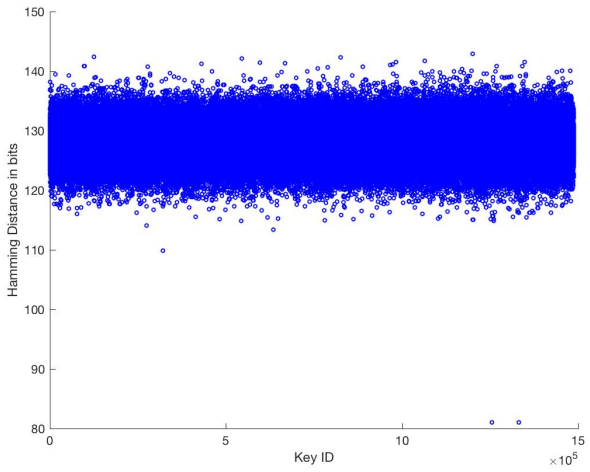

Figure 5.11: Average Hamming distances of the different user's keys of the 2^{nd} Dataset

it is observable, the Hamming distance among the keys of the same user that are transformed with different random keys are similar to the Hamming distance among the keys of the same user that are transformed with the same random keys. This shows a really good distinctiveness between these sets of keys. It means that whenever we need to cancel the fingerprints' template, we will obtain a new template which is totally different from the previous one and it is preserving the original template to be compromised. In addition, similar to the analysis described in Section 5.3, the fingerprint similarity attack complexity can be calculated by the Hamming distance among minutiae concatenations before applying the hash function on the same user's minutiae points that are transformed with different random numbers, whose result yields to 2^{252} in both of the utilized datasets. This amount of complexity shows that when we cancel the original fingerprints' template, the attacker can not substitute the original minutiae with his/her own minutiae. Therefore, whenever we need to cancel the fingerprints' template it will be easy and safe.

In order to justify the noninvertibility property, if we assume that an attacker finds (X', Y') which X' is the location of minutia after transformation on X-coordinate and Y' is the location of minutia after transformation on Y-coordinate, (s)he will try her/his best to find (X, Y) (X is the location of minutia on X-coordinate and Y is the location of minutia on Y-coordinate) using the knowledge of the transformation function and other

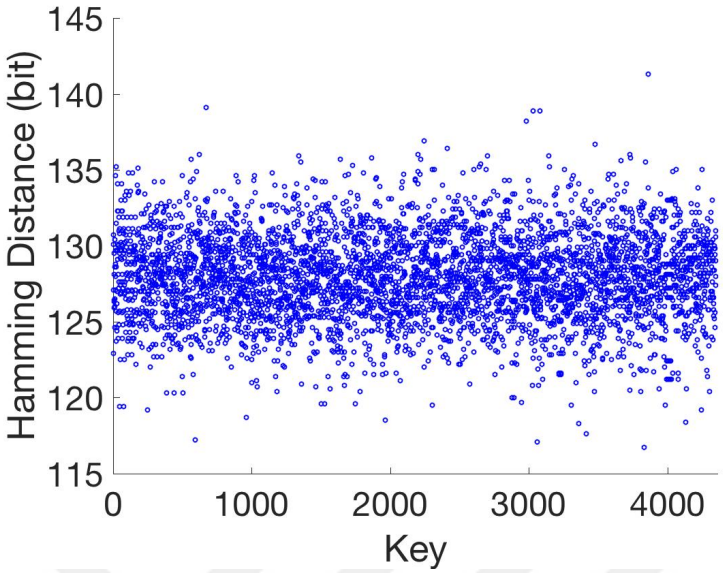

Figure 5.12: Average Hamming distances of the same user's keys generated with different random numbers for the 1^{st} Dataset

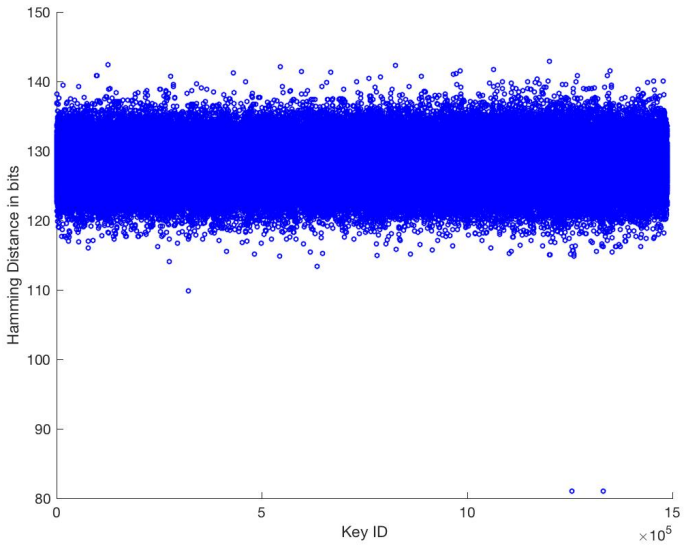

Figure 5.13: Average Hamming distances of the same user's keys generated with different random numbers for the 2^{nd} Dataset

public data. First of all, we should mention that our periodic transformation function has the many-to-one mapping property. It means that, based on Equation 4.3 and Equation 4.4 X can only be transformed into one X', and similarly Y can only be transformed into one Y', while X' and Y' can be transformed into many Xs and many Ys. Besides, if

the adversary has X', Y' and r_i, (s)he could find α_i and β_i, but since the module m_i is unknown, the adversary cannot infer either X or Y.

5.4 Computational Complexity Analysis

The computational complexity of our SKA-CaNPT protocol is computed for the server and the user separately by calculating the complexity of the transformation function and calculating the number of key generation attempts.

As we mention in Section 4, our cancelable and noninvertible transformation function is the sine periodic function. As discussed in [61, 62], the computational complexity of $sin(x)$ for fixed-point rational inputs are computable in polynomial space. Therefore, both in the server side and the user side, the transforming functions' computational complexity is $O(n)$, as n is the total number of the minutiae points. As discussed exclusively in Section 4 and as can be analyzed from Figure 4.1, the computational complexity of the server for key generation, can be calculated by Equation 5.3, where n_{com} is the number of common found minutiae. Therefore, we can calculate the server's average computational complexity based on the average of all key generation attempts against all subjects. It means that the average complexity of the server in the key generation step is 2^{17} in the first dataset and 2^9 in the second dataset. By this means, the total computational complexity of the server in our SKA-CaNPT protocol is equal to 2^{17} in the first dataset and 2^9 in the second dataset, since $O(n) < O(2^n)$

$$server_{complexity} = \sum_{i=n_{com}}^{n_{com}^{key}} \binom{n_{com}}{i} \tag{5.3}$$

In the user side, as we discuss in Section 4.2, after extracting the minutiae, we transform the original points by the sine transformation function. As we discussed before, the computational complexity of this operation is $O(n)$. Afterwards, in the key generation step, the number of key generation attempts can be calculated according to Equation 5.4,

in which n_u is the number of genuine minutiae in the user side. Therefore, after calculating all the tests' computational complexity of the key generation step, we can find the average of it, which is equal to 2^{39} for the first dataset and 2^{41} for the second dataset. Indeed, the total computational complexity of the user is equal to 2^{39} for the first dataset and 2^{41} for the second dataset, since $O(n) < O(2^n)$.

$$user_{complexity} = \sum_{i=n_{com}}^{n_{com}^{key}} \binom{n_u}{i} \tag{5.4}$$

5.5 Communication Complexity Analysis

The communication complexity of our SKA-CaNPT protocol can be computed according to the bits transmitted between the server and the user. First of all, the user sends a message to the server which is the starting point of our protocol. This message includes Q_u (set of genuine and fake minutiae) and the user's ID. Since we are using SHA-256, Q_u has 256 bits and the user *ID* is a 32 bit integer. In the dataset that we utilize, there are 440 and 550 minutiae points respectively for the first dataset and second dataset, in the average. Therefore, the communication cost of the first message is equal to $(440 \times 256) + 32 = 112672$ bits which is equal to 13.75 KB in the first dataset and $(550 \times 256) + 32 = 140832$ which is equal to 17.2 KB in the second dataset.

In the following, the second message transmitted from the server to the user can be either a rejection acknowledgment or n_{com} and *HMAC* values. Actually, rejection acknowledgement is just 1 bit, but if the user is accepted then the message will contain *HMAC*'s value, which is 256 bits, and the number of common minutia (n_{com}), which is a 32 bit integer. So, the communication cost of this message is equal to $256 + 32 = 288$ bits, in the worst case.

In the next step of our SKA-CaNPT protocol, the user responds to the server's message with *RETRY* or *ACCEPT*, both of which requires only 1 bit. If the user sends a *RETRY*

message, the server can transmit a negative acknowledgement or the list of the *HMAC* values. *HMAC* list contains $\binom{n_{com}}{n_{com}-1}$ number of minutiae points. The average of n_{com} is equal to 25 for the both datasets. Thus, the average communication cost of this kind of a message is equal to $256 \times \binom{22}{21} = 5632$ bits. If the user can not verify any of the *HMAC* values, it will send another *RETRY* message, which is again 1 bit. On the other hand, if the SKA-CaNPT protocol continues with an *ACCEPT* message of the user, then the positive acknowledgement is 1 bit, and the index of *HMAC* is a 32 bits integer. So, the total communication cost is equal to 33 bits.

In the following steps, the server's communication cost will change according to the following combinatorial value: $\binom{n_{com}}{x}$, where x = $n_{com} - 2$, $n_{com} - 3$, $n_{com} - 4$,...., n_{com}^{key} is the number minutiae which is used in the key generation process. Therefore, until the time that the key is generated, the communication cost of our SKA-CaNPT protocol changes in each round. However, the key is generated with an average number of 16 minutiae in the first dataset and 20 in the second dataset. Thereupon, the average communication cost in the server side is equal to $256 \times \binom{25}{16}$ = 523001600 bits = 65.37 MB for the first dataset and $256 \times \binom{25}{20}$ = 13601280 bits = 1.70016 MB for the second dataset. On the other hand, in the first dataset, the total size of the messages transmitted by the server is 65.37 MB and it is 13.75 KB for the user, while in the second dataset, the total size of the messages transmitted by the server is 1.70016 MB and it is 17.2 KB for the user. Hence, with today's Internet quality, the above-mentioned amounts are reasonable.

5.6 Memory Requirement Analysis

In our SKA-CaNPT protocol, the average number of transmitted minutiae is 42.4, which it is approximately 42 for the first dataset and it is 51 for the second dataset, in the server side. Moreover, in the server side, we are also storing each minutiae's neighborhood points. As we mentioned before, we have a neighborhood relation strategy, by which according to a predefined threshold value, we are storing representative minutiae's neighbor points in the server. The value of T_{dist} here is 10. Therefore, each representative minutiae

is represented with $21 \times 21 = 441$ points. It means that for each subject, the average number of minutiae that is stored in the server side is equal to $42 \times 441 = 18522$ for the first dataset, and $51 \times 441 = 22491$ for the second dataset. In addition, before storing these points on the server, first SHA-256 hash function is applied on them, which means that each point is 256 bits. The first dataset that we utilize consists of 30 subjects and the second dataset includes 292 subjects, as mentioned before. Therefore, the total storage size required by the server is equal to $30 \times 18522 \times 30 = 142,248,960$ bits for the first dataset, and for the second dataset it is $292 \times 22491 \times 292 = 1,917,672,624$ bits; in the other meaning, for the first dataset it is 16.9 MB and for the second dataset it is 239.7 MB. In the case that the server needs to store double hashes of the minutiae points and their neighbors, the storage amount will become $16.9 \times 2 = 33.8$ MB and $239.7 \times 2 = 479.4$ MB for each dataset respectively.

On the other hand, the average number of transmitted minutiae is 39.65, which is approximately equal to 40, in the first dataset, and it is 50 in the second dataset, in the user side. Here, the neighboring points are not stored, but $10 \times |G_u|$ chaff points are added to the original minutiae points. It means that, on the average, in the first dataset, we are adding $10 \times 40 = 400$, and in the second dataset $10 \times 50 = 500$ chaff points to the original minutiae points. Therefore, in the user side, the total number of minutiae points for the first dataset is equal to 440 points, and for the second dataset, it is 550 points. In the user side, first we calculate the single hashes of the minutiae points and then we calculate the double hash values. Hence, in total, our SKA-CaNPT protocol requires $440 \times 256 \times 2 = 28.16$ KB of storage for a user in the first dataset, and $550 \times 256 \times 2 = 35.2$ KB for the second dataset.

5.7 Comparative Analysis with the Related Work

In this section, we compare our SKA-CaNPT protocol with the SKA-PB protocol that is presented by Akdoğan et al. [7] and the cancelable fuzzy vault approach that is presented by Dang et al. [8]. These two protocols are compared with that of ours according to IKGR

and CKGR percentages and their brute-force attack complexities, as provided in Table 5.1, where NA means not applicable. We also compared our SKA-CaNPT protocol with the SKA-PB protocol with respect to replay and impersonation attack complexities together with the randomness and the distinctiveness of the agreed symmetric keys. The reason behind not being able to compare our protocol with the cancelable fuzzy vault approach against these performance metrics is that the authors do not include such analysis in their corresponding proposal.

The IKGR and CKGR of our SKA-CaNPT protocol is 0.67% and 96.67% for the first dataset, and 0.95% and 90.41% for the second dataset. These values are 0.57% and 99.43% in the SKA-PB protocol; nevertheless, our results are still reasonable and acceptable for bio-cryptographic authentication systems. On the other hand, the complexity of the brute-force, replay and impersonation attacks are the same of each other in both our SKA-CaNPT protocol and the base SKA-PB protocol, which is 2^{94} for the 1first dataset, and 2^{118} for the second dataset. In addition, the randomness of the agreed keys is 0.9745 and 0.95 in our SKA-CaNPT protocol, which is 0.98 and 0.94 in the SKA-PB protocol, respectively for the first and second datasets; hence, they are approximately similar, as well. Besides, the distinctiveness of the agreed keys in both of these protocols are between 120-130 bits, which are the same. As it is shown in Table 5.1 and our SKA-CaNPT protocol successfully improves the SKA-PB protocol by means of adding the cancelability and noninvertibility properties; despite degrading the IKGR and CKGR performance a little. Our SKA-CaNPT protocol preserves the attack complexity, randomness and distinctiveness of the SKA-PB protocol.

In [8], Dang et al. apply their proposed scheme on face features with 40 genuine minutiae points and on a polynomial of order 9 fuzzy vault. We applied this proposed method to both of our datasets. We can represent each of the x and y coordinates of the fingerprint values with 14 bits in both of the datasets; thus, when we concatenate them together we have 28 bits. According to our SKA-CaNPT protocol, the agreed cryptographic keys are 256 bits. So, $[256/28]$ is equal to 9, which means that we should compare our protocol with the Deng et al. proposed protocol with fuzzy vault of order 9. The comparison of

our SKA-CaNPT protocol with the cancelable fuzzy vault scheme shows that the IKGR and CKGR of our SKA-CaNPT protocol outperforms the IKGR and CKGR of the cancelable fuzzy vault scheme which are 1.8% and 54.4% for the first dataset, and 2.98% and 65.95% for the second dataset. The authors of this approach computed resistance against the brute-force attack for their approach when they used order 9 polynomial as 2^{57}. Our SKA-CaNPT protocol's brute-force attack complexity is 2^{94} for the first dataset and 2^{118} for the second one, which in comparison with the cancelable fuzzy vault scheme's attack complexity is really better. In conclusion, our SKA-CaNPT protocol is stronger against brute-force attack in comparison to cancelable fuzzy vault scheme. Since the cancelable fuzzy vault scheme is different from our protocol in the manner that the cryptographic keys are generated, comparing these two protocols in terms of randomness and distinctiveness of the agreed keys is not applicable.

Table 5.1: Comparison of SKA-CaNPT with SKA-PB protocol and the Cancelable Fuzzy Vault approach

Protocol	*SKA-CaNPT 1st Dataset*	*SKA-PB 1st Dataset*	*SKA-CaNPT 2nd Dataset*	*SKA-PB 2nd Dataset*	*Cancelable Fuzzy Vault 1st Dataset*	*Cancelable Fuzzy Vault 2nd Dataset*
IKGR	0.67%	0.57%	0.95%	0.57%	1.8%	2.98%
CKGR	96.67%	99.43%	90.41%	99.43%	54.4%	65.95%
Brute-force attack complexity	2^{94}	2^{94}	2^{118}	2^{118}	2^{57}	2^{57}
Replay attack complexity	2^{94}	2^{94}	2^{118}	2^{118}	NA	NA
Impersonation attack complexity	2^{94}	2^{94}	2^{118}	2^{118}	NA	NA
Original minutiae compromising	*No*	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Agreed key randomness	0.9745	0.98	0.95	0.94	NA	NA
Agreed key distinctiveness	120 – 130 bits	120 – 130 bits	120 – 130 bits	120 – 130 bits	NA	NA
Fingerprint similarity attack complexity	2^{251}	2^{205}	2^{252}	2^{205}	NA	NA

6 *Conclusions*

In this thesis, we present a secure key agreement protocol called SKA-CaNPT: Secure Key Agreement Protocol using Cancelable and Noninvertible Biometrics based on Periodic Transformation. As the name of our protocol implies, we use cancelable and noninvertible biometric features while generating the keys to be agreed, and we satisfy these properties through the use of periodic transformation. As a proof of concept, we apply our protocol on fingerprint images and we generate the secretly shared keys using minutiae points, which means that we are not using any random data while generating the key. Both the cryptographic keys that are agreed upon and the stored biometric templates that are generated using our SKA-CaNPT protocol are cancelable since, if we need to renew the biometric template, we can easily change one of the inputs of our sine transformation function to compute a brand new symmetric shared key. Besides, the stored biometric templates that are generated using our SKA-CaNPT protocol are noninvertible because of the utilized periodic transformation function. In the other words, the sine function is periodic with a period of a 2π, because of which we cannot infer the original biometric feature due to the sine function being many-to-one.

In our SKA-CaNPT protocol, we use chaff points in order to hide the genuine minutiae. We generate these chaff points according to a predefined strategy and then we apply our cancelable noninvertible transformation function on them to make them completely indistinguishable from the genuine minutiae. Since adding chaff points decreases the probability of information leakage to the attacker, it results in a higher verification perfor-

mance. I should mention that during the key generation process, our system uses the hash functions and threshold mechanisms as well.

We analyze the security of our proposed SKA-CaNPT protocol according to a number of different metrics. The *IKGR* values are 0.67% and 0.95%, while the *CKGR* values are 96.67% and 90.45%, respectively for the first and second datasets that we utilize, which shows the verification performance of our protocol. In addition, we analyze the strength of our protocol against brute-force, replay and impersonation attacks. The attack complexity of our protocol is 2^{94} in the first dataset, which means that the attacker needs 2^{93} trials to attack our system, while it is 2^{118} in the second dataset, which means that the attacker needs 2^{117} trials to attack our system, each of which implies a good complexity. We also analyze the randomness of our generated keys, which shows that the entropy of 84% of the generated keys are higher than 0.994 and the entropy of all the keys are greater than 0.9745 in the first dataset. Moreover, in the second dataset, the entropy of the 99.3% of the generated keys are higher than 0.98 and the entropy of all the keys are greater than 0.95. Hence, it can be concluded that the keys are random enough to be used as cryptographic keys. On the other hand, according to the Hamming distance metric, we evaluate the distinctiveness of the agreed keys, which shows that there is a quite large difference between the same person's keys and different persons' keys, making them completely indistinguishable from each other.

BIBLIOGRAPHY

[1] C. Barral, "Biometrics and security: Combining fingerprints, smard cards and cryptography," 2010.

[2] L. Wiskott1, J. Fellous, N. Kruger1and, and C. Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 775–779, 1997.

[3] N. Ratha, S. Chikkerur, J. Connell, and R. Bolle, "Generating cancelable fingerprint templates," *Pattern Analysis and Machine Intelligence of the IEEE Transactions*, vol. 29, no. 4, pp. 561–572, 2007.

[4] 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.

[5] R. Bolle, J. Connell, S. Pankanti, N. Ratha, and W. Andrew, "Senior, guide to biometrics," 2003.

[6] F. Hao, R. Anderson, and J. Daugman, "Combining cryptography with biometrics effectively," tech. rep., University of Cambridge, 2005.

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

[8] T. Dang, Q. Truong, T. Le1, and H. Truong, "Cancellable fuzzy vault with periodic transformation for biometric template protection," *IET Biometrics*, pp. 1–7, 2016.

[9] J. Wegstein, "A computer oriented single fingerprint identification system," tech. rep., National Bureau of Standards, NBS Technical note 443, Jan. 1968.

[10] J. Wegstein, "Matching fingerprints by computers," tech. rep., National Bureau of Standards, NBS Technical note 466, Jul. 1968.

[11] M. Baca, G. P., and F. T., *New Trends and Developments in Biometrics*. The Author(s), 2012.

[12] A. Moreno, A. Sanchez, J. Velez, and J. Diaz, "Face recognition using 3d local geometrical features: Pca vs. svm," *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pp. 185–190, 2005.

[13] "Samsung galaxy s5: delta scaner." https://eu.community.samsung.com/t5/Smartphone/Schermo-Samsung-galaxy-s5.

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

[15] W. Stallings, *Computer Networking with Internet Protocols and Technology*. NY, USA: Prentice Hall, 2003.

[16] D. Hein, "Elliptic curve cryptography asic for radio frequency authentication," 2008.

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

[18] M. Feldhofer and J. Wolkerstorfer, "Strong crypto for rfid tags a comparison of low-power hardware implementations," *IEEE International Symposium on Circuits and Systems ISCAS*, pp. 1839–1842, 2007.

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

[20] S. Halevi, "Cryptographic hash functions and their applications," tech. rep., IBM Research, Aug. 2009.

[21] D. Eastlake 3rd and P. Jones, "Secure hash algorithm 1 (sha1)," 1995.

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

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

[24] B. Jisha Nair and S. Ranjitha Kumari, "A review on biometric cryptosystems," *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, vol. 6, pp. 46–53, 2015.

[25] 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.

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

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

[28] Y. Lee, K. Bae, S. Lee, K. Park, and J. Kim, "Biometric key binding: Fuzzy vault based on iris images," *In Proceedings of Second International Conference on Biometrics*, vol. 38, pp. 800–808, 2007.

[29] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, and J. Gonzalez-Rodriguez, "Bayesian adaptation for user-dependent multimodal biometric authentication," *Pattern Recognition*, vol. 38, pp. 1317–1319, 2005.

[30] Y. Chung, D. Moon, S. Lee, S. Jung, T. Kim, and D. Ahn, "Automatic alignment of fingerprint features for fuzzy fingerprint vault," *In Proceedings of Conference on Information Security and Cryptology*, pp. 358–369, 2005.

[31] T. Clancy, D. Lin, and N. Kiyavash, "Secure smartcard-based fingerprint authentication," *In Proceedings of ACM SIGMM Workshop on Biometric Methods and Applications*, pp. 45–52, 2003.

[32] A. Nagar and S. Chaudhury, "Biometrics based asymmetric cryptosystem design using modified fuzzy vault scheme," *In Proceedings of IEEE International Conference Pattern Recognition*, vol. 4, pp. 537–540, 2006.

[33] U. Uludag and A. Jain, "Securing fingerprint template: Fuzzy vault with helper data," *In Proceedings of CVPR Workshop on Privacy Research In Vision*, p. 163, 2006.

[34] S. Yang and I. Verbauwhede, "Automatic secure fingerprint verification system based on fuzzy vault scheme," *In Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, p. 609612, 2005.

[35] Y.-J. Chang, W. Zhang, and T. Chen, "Biometrics based cryptographic key generation," *In Proceedings of IEEE Conference on Multimedia and Expo*, vol. 3, pp. 2203–2206, 2004.

[36] C. Vielhauer, R. Steinmetz, and A. Mayerhofer, "Biometric hash based on statistical features of online signatures," *In Proceedings of 16th International Conference on Pattern Recognition*, vol. 1, pp. 123–126, 2002.

[37] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data.," tech. rep., Technical Report 235, Cryptology ePrint Archive, February 2006.

[38] K. Nandakumar, A. Jain, and S. Pankanti, "Fingerprint-based fuzzy vault: implementation and performance," *IEEE Transactions on Information Forensics and Security*, vol. 2, pp. 744–757, 2007.

[39] F. Hao, R. Anderson, and J. Daugman, "Combining crypto with biometrics effectively," *IEEE Transactions on Computers*, vol. 55, pp. 1081–1088, 2006.

[40] T. Linh Vo, T. Dang, and J. Küng, "A hash-index method for securing fuzzy vaults," *International Conference on Trust, Privacy and Security in Digital Business*, vol. 8647, pp. 60–71, 2014.

[41] Y. Wu and B. Qiu, "Transforming a pattern identifier into biometric key generators," *IEEE International Conference on Multimedia and Expo (ICME)*, p. 7882, 2010.

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

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

[44] A. Kholmatov and B. Yanikoglu, "Realization of correlation attack against the fuzzy vault scheme," *BInternational Society for Optics and Photonics*, vol. 6819, pp. 1–6, 2008.

[45] C. Orencik, T. Pedersen, E. Savas, and M. Keskinoz, "Improved fuzzy vault scheme for

ngerprint verification," *International Conference on Security and Cryptography*, pp. 37–43, 2008.

[46] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. New York: Springer-Verlag, 2003.

[47] A. Jin and L. Hui, "Cancellable biometrics," *Scholarpedia*, vol. 5, no. 1, 2010.

[48] R. Mukhaiyar, "Cancellable biometric using matrix approaches," 2015.

[49] A. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP Journal on Advances in Signal Processing*, no. 113, p. 117, 2008.

[50] S. Shin, M.-K. Lee, D. Moon, and K. Moon, "Dictionary attack on functional transform-based cancelable fingerprint templates," *ETRI journal*, vol. 31, no. 5, pp. 628–630, 2009.

[51] F. Quan, S. Fei, C. Anni, and Z. Feifei, "Cracking cancelable fingerprint template of ratha," *In Proceedings of the International Symposium on Computer Science and Computational Technology. Washington, DC, USA: IEEE Computer Society*, pp. 572–575, 2008.

[52] A. Jin, D. Ling, and A. Goh, "Biohashing: two factor authentication featuring fingerprint data and tokenised random number," *Pattern recognition*, vol. 37, no. 11, pp. 2245–2255, 2004.

[53] F. Farooq, R. Bolle, T. Jea, and N. Ratha, "Anonymous and revocable fingerprint recognition," *In Proceedings of Computer Vision and Pattern Recognition, Minneapolis*, 2007.

[54] S. Chikkerur, N. Ratha, H. Connell, and R. Bolle, "Generating registration-free cancellable fingerprint templates," *In BTAS08*, pp. 1–6, 2008.

[55] "Neurotechnology Verifinger sample db." http://www.neurotechnology.com/. Accessed: 2015-05-01.

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

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

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

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

[60] M. uliato and C. Gebotys, "Fpga implementation of an hmac processor based on the sha-2 family of hash functions," tech. rep., Technical Report, University of Waterloo, 2011.

[61] J. Reif and S. Tatey, “On threshold circuits and polynomial computation,” tech. rep., Department of Computer Science, Duke University, Durham, North Carolina 27706.

[62] J. Reif, “Logarithmic depth circuits for algebraic functions,” *SIAM journal of computing*, vol. 15, no. 1, pp. 1231–242, 1986.