2012

Remote user authentication in distributed networks and systems

Jiangshan Yu

University of Wollongong

Recommended Citation
UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Remote User Authentication in Distributed Networks and Systems

A thesis submitted in fulfillment of the requirements for the award of the degree

Master of Computer Science by Research

from

UNIVERSITY OF WOLLONGONG

by

Jiangshan Yu

School of Computer Science and Software Engineering
September 2012
© Copyright 2012

by

Jiangshan Yu

All Rights Reserved
Dedicated to  
My Parents & Grandparents
Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

________________________________________
Jiangshan Yu
September 20, 2012
Abstract

Entity authentication is becoming more and more important. With widespread use of distributed computer networks, for example, cellular networks, virtual reality communities, World Wide Web, peer-to-peer networks and multiplayer online games, there is a need to be more vigilant about the security and privacy of users. One way to address the security and privacy concerns is remote user authentication and this is widely used in distributed systems for identifying users and servers. Remote user authentication is a means of identifying a user and verifying whether this user has permission to access the network services and resources. However, an attacker may impersonate a server to communicate with a user and then, the attacker is able to steal the user’s information. Thereafter, the attacker can pass authentication with the real server by using the stolen information of the user. Therefore, mutual authentication is needed in order to prevent bogus server attacks. Other requirements of user authentication include ensuring the confidentiality of further exchanging messages, protecting user privacy, providing user anonymity and achieving unlinkability. In the complex environments of computer networks, however, it is a challenge to design efficient and secure mutual authentication protocols under such security requirements.

The research reported here aims to provide efficient and secure identification services with further security requirements for users in distributed systems and networks. In general, the identification services may require three factors, i.e., password, smart card and biometric characteristics. The authentication which based on password is called password-based authentication. Password-based authentication together with another factor, smart card, is called two-factor authentication. In which, a successful user authentication can be achieved if the user has a correct password together with a corresponding smart card. The biometric-based authentication mainly based on the biometric characteristics, for example, finger print, iris scan and a face, and it may also require a smart card. The three-factor authentication consists
all of these three factors, i.e., password, smart card and biometric characteristics. There is another concept which belongs to two-factor authentication, called single sign-on (SSO). It enables a user to use a unitary secure credential (or token) to access multiple computers and systems where he/she has access permissions.

The contributions of this thesis are research on both single sign-on and three-factor authentication. In particular, this research will analyze the recent, supposed secure single sign-on scheme proposed in 2012 by Chang and Lee [CL12]. However, their scheme is actually not secure as we show that it fails to meet credential privacy and soundness of authentication. Based on this analysis, this research will suggest repairs to the scheme by employing the efficient verifiable encryption of RSA signature (RSA-VES) proposed by Ateniese [Ate99] for realizing fair exchange of RSA signatures. In addition, this research will formalize the security model of single sign-on schemes with authenticated key exchange, and based on the model, a provably secure single sign-on scheme will be proposed. This scheme satisfies soundness, preserves credential privacy, meets user anonymity and supports session key exchange. For users who have higher security requirements, this research also proposes an improved generic framework, which is an efficiently systematic approach which upgrades two-factor authentication schemes to three-factor authentication schemes. This research also provides a provably secure concrete instantiation of the framework with comparison, practicability analysis, privacy discussion and formal security proof.
I would like to thank Dr. Guilin Wang and Prof. Yi Mu, my supervisors, for their patience and kind guidance during my study. Many thanks for your supervision which was vital to me in achieving my goals. I also would like to thank all my friends for their friendship. Finally, I greatly appreciate my family for their love, support and understanding.
Publications and Draft


2. Guilin Wang, Jiangshan Yu, and Qi Xie, “Security Analysis of A Single Sign-On Mechanism for Distributed Computer Networks”, IEEE Transactions on Industrial Informatics, July, 2012. (Provisionally accepted after second round review; the draft can be found in IACR Cryptology ePrint Archive 2012: 107)

# Contents

Abstract \hspace{1cm} v

Acknowledgement \hspace{1cm} vii

Publications and Draft \hspace{1cm} viii

1 Introduction \hspace{1cm} 1
   1.1 Overview of Cryptography \hspace{1cm} 1
   1.2 User Authentication \hspace{1cm} 2
   1.3 Related Work \hspace{1cm} 3
      1.3.1 Password-Based Authentication \hspace{1cm} 3
      1.3.2 Two-Factor Authentication \hspace{1cm} 3
      1.3.3 Biometric Authentication \hspace{1cm} 5
      1.3.4 Three-Factor Authentication \hspace{1cm} 6
   1.4 Challenges \hspace{1cm} 8
   1.5 Aims and Objectives \hspace{1cm} 8
   1.6 Organisation of The Thesis \hspace{1cm} 9

2 Background \hspace{1cm} 11
   2.1 Intractable Problems \hspace{1cm} 11
      2.1.1 Discrete Logarithm Problem \hspace{1cm} 11
      2.1.2 Diffie-Hellman Problem \hspace{1cm} 11
   2.2 Cryptographic Tools \hspace{1cm} 12
      2.2.1 Cryptographic Hash Functions \hspace{1cm} 12
      2.2.2 Time Stamp \hspace{1cm} 13
      2.2.3 Diffie-Hellman Key Exchange \hspace{1cm} 13
   2.3 Encryption Techniques \hspace{1cm} 14
5 A Generic Framework of Three-Factor Authentication 56

5.1 Introduction ....................................................... 56

5.2 Biometric Identification Mechanisms ......................... 57
  5.2.1 Fuzzy Extractor ............................................ 58
  5.2.2 Fuzzy Vault .................................................. 59

5.3 A Generic Three-factor Authentication Framework .......... 61
  5.3.1 Review of Huang et al.’s Framework ....................... 61
  5.3.2 Improved Framework ....................................... 63

5.4 Concrete Instantiation ........................................... 65
  5.4.1 Concrete Protocol ........................................... 66
  5.4.2 Analysis of Implementation ................................. 67
  5.4.3 Formal Security Proof of Instantiation Protocol .......... 70
  5.4.4 Privacy Discussion ......................................... 80

5.5 Conclusion ......................................................... 81

6 Conclusion 82

6.1 Contributions .................................................... 82

6.2 Open Problems .................................................... 83

Bibliography 84
List of Tables

3.1 Notations in the Chang-Lee Scheme ............................................. 23
4.1 Notations in the Proposed SSO Scheme ............................................ 48
5.1 Notations in the Concrete Three-Factor Authentication ......................... 65
5.2 Parameters in Different Databases [NJP07] ........................................ 68
5.3 Comparison of Schemes ................................................................. 69
List of Figures

3.1 User Identification Phase of the Chang-Lee Scheme 25
3.2 The Proposed Improved Scheme 36
4.1 Participant Identification Phase of the Proposed SSO Scheme 50
5.1 GAR and FAR of the ‘Fuzzy Vault’ [NNJ08] 68
Chapter 1

Introduction

1.1 Overview of Cryptography

Classic cryptography is the techniques of hiding the meaning of a written text. It was first documented in the use of non-standard hieroglyphs by ancient Egyptians circa 1900 B.C., for secure communication in the presence of third parties, i.e. ‘adversaries’. Since World War I it has been growing and effectively became synonymous with ‘encryption’ until the advent of modern cryptography.

Modern cryptography intersects with a number of different disciplines, like mathematics, computer science and electrical engineering. The algorithms designed for modern cryptography normally rely on computational hardness assumptions, for example, the difficulty of integer factorization in number theory. Theoretically, it is, indeed, possible to break these algorithms (e.g. by brute force attack) but it is unfeasible using known techniques and computational devices. So, the security of cryptographic algorithms and their applications are called computationally secure. Today, in terms of information security, the field of cryptography has been expanded from confidentiality and integrity to various aspects such as authentication, non-repudiation, trust and privacy.

Public key cryptography, also known as asymmetric cryptography, was invented in the late 1970s. It enables building secret communication using a public channel without the establishment of a prior secret key. In 1976, Diffie and Hellman [DH76] were first proposed a solution to address the problem of key distribution using public-key cryptography. Their idea involved using two distinct keys, one for plaintext encryption that can be made public, and one for ciphertext decryption which is kept private. Key generation requires that deducing the secret key from the public key is computationally unfeasible. Public key cryptosystems make authentication easy to achieve and have inspired a lot of research. As a result, a number of schemes
1.2 User Authentication

User authentication is the process of individual identity confirmation, to ensure that an individual is really who he claims to be. Probably the earliest user authentication mechanism was based on passwords. This concept was first proposed by Lamport in 1981 [Lam81], and remains the most common mechanism for user authentication in computer systems and networks.

While such protocols have been widely used, a number of problems have appeared, for example, the poor selection of passwords, the shortcoming of capture by Trojans and the reuse of passwords. These can lead to attacks such as dictionary attacks. Dictionary attack is the method to break the password-based authentication scheme by systematically trying every likely word or the likely combination of words in a dictionary as a password. This attack works because that many users prefer to use some ordinary words as passwords. For example, the user’s first name or his/her telephone number. A good remedy is the use of hardware authentication tokens together with passwords to enhance security. This is called two factor authentication, which has become popular, consisting of a password together with a hardware token which is usually a smart card.

Since Chang and Wu [CW91] introduced the remote user authentication scheme using smart cards in 1991, there are many two-factor authentication schemes which have been proposed. The security, however, could remain compromised since the smart card may be stolen, the range of possible password could be small and users may frequently forget or lose their passwords. Due to such concerns, biometric identification, which exploits the biometric features of the user to authenticate him/herself, has been introduced.

Biometric identification overcomes the flaws of two-factor authentication because biometric features have high entropy, cannot be forgotten and are rarely lost [JR03]. The first biometric authentication scheme was ‘fuzzy commitment’, proposed by Juels and Wattenberg [JW99] in 1999. This has inspired many subsequent researchers. One problem is that biometric features are not completely private since they may easily be ‘stolen’; e.g. the fingerprint can be obtained from things the person has touched and the facial features may be obtained from a user’s photograph. A way to alleviate these problems is to combine all three of these factors in what is
1.3 Related Work

1.3.1 Password-Based Authentication

To thwart the compromise of password table, which is maintained by a server, many schemes [EKW74, LMM81] have been proposed using password hashing rather than the plain password in a directory table. This method protects passwords even when the directory table is disclosed. However, an adversary may impersonate a legal user to pass authentication by modifying the data in the directory table. Other schemes, such as [SKS+92, NSC+93, OR87, SY96, Syv93], attempt to ensure the authentication with the help of a trusted third party in networks, in which, the secret information (e.g. secret key) must be stored in a table on the server side. Thus, security is not reliable since leaking of the table could lead to system breakage.

1.3.2 Two-Factor Authentication

To eliminate the shortcomings of using directory tables, two-factor schemes which are based on both a password and a smart card have been proposed [CW91, CH93, OT89]. However, they all have drawbacks [YS99]. To resolve the problems in these schemes, Yang and Shieh [YS99] proposed two two-factor authentication schemes, one based on timestamp and the other based on random nonce. Both support easy password changing. Later, Chan and Cheng [CC02], and Fan et al. [FLZ02] found that the Yang-Schieh scheme is insecure against impersonation attack. To remedy this flaw, Shen, Lin and Hwang [SLH03], and Yang, Wang and Chang [YWC05] suggested improvements to the Yang-Schieh scheme. However, Yoon et al. [YKY05] identified attacks on the YWC-scheme [YWC05], and then improved the scheme. In 2006, however, Wang and Bao [WB06] pointed out that both the SLH-scheme [SLH03] and Yoon et al.’s scheme [YKY05] are vulnerable to impersonation attacks.
Single Sign-On

With the increasing usage of network services, a user may need to maintain more and more ID/password pairs for accessing different distributed service providers. This imposes a burden on users and service providers as well as the communication overhead of computer networks. To tackle this problem, a single sign-on (SSO) mechanism \cite{Gro} has been introduced so that after obtaining a credential from a trusted authority, each legal user can use this single credential to authenticate itself and then access multiple service providers.

Intuitively, an SSO scheme should meet at least three basic security requirements: completeness, soundness and credential privacy. Completeness of authentication \cite{BR93a} requires that: (a) both sides accept each other if they have matched the conversation; (b) the probability that one side accepts the other one who actually has not engaged in the matching conversation is negligible. Soundness means that an unregistered user without a credential should not be able to access the services offered by service providers. Credential privacy guarantees that colluding dishonest service providers should not be able to fully recover a user’s credential and then impersonate the user to log in to other service providers.

Formal security definitions of SSO schemes were given in \cite{HMSY10}. However, soundness of credential based authentication has not been formally studied yet despite its importance, and the preserving of both soundness and credential privacy is still a challenge in designing an SSO \cite{WYX12}.

In 2000, Lee and Chang \cite{LC00} first proposed a user identification and key distribution scheme, actually an SSO scheme, maintaining user anonymity in distributed computer networks. Later, Wu and Hsu \cite{WH04} pointed out that the Lee-Chang scheme is vulnerable to masquerading attacks and identity disclosure attacks. The former enable an adversary to impersonate a service provider to exchange the session key with users and then to obtain sensitive information in further communication. This is possible because of the one-way authentication in the Lee-Chang scheme. The second type of attack, which focuses on the user anonymity, can expose the identity of a user. Meanwhile, Yang et al. \cite{YWB+04} showed that the Wu-Hsu scheme cannot preserve credential privacy either, since a malicious service provider can recover users’ credentials, and they then proposed an improvement to overcome this limitation. In 2006, however, Mangipudi and Katti \cite{MK06} pointed out that
Yang et al.’s scheme is insecure against DoS (Deniable of Service) attack and presented a new scheme. In 2009, Hsu and Chuang [HC09] demonstrated that both the Yang et al. and the Mangipudi-Katti schemes do not provide user anonymity since their schemes are vulnerable to identity disclosure attacks. To prevent such attacks, Hsu and Chuang proposed an RSA-based user identification scheme.

In [HMSY10], Han et al. proposed a generic SSO construction which relies on broadcast encryption plus zero knowledge (ZK) proof [FFS88] showing that the prover knows the corresponding private key of a given public key. So, implicitly, each user is assumed to have been issued a public key in a public key infrastructure (PKI). In the setting of an RSA cryptosystem, such ZK proof is very inefficient due to the complexity of interactive communications between the prover (a user) and the verifier (a service provider).

Recently, Chang and Lee [CL12] pointed out that the Hsu-Chuang scheme is vulnerable to impersonation attacks and the scheme requires additional time-synchronized mechanisms which have unstable latency in distributed networks. Then, they proposed a user anonymity preserving improvement with high efficiency. The scheme used random nonce to replace an additional time-synchronized mechanism, does not need PKI (Public key infrastructure) for users, and is suitable for mobile device users. Compared with Han et al.’s generic scheme, the Chang-Lee scheme has several attractive features: less underlying primitives without using broadcast encryption, high efficiency without resort to ZK proof and no requirement of PKI for users. Unfortunately, the analysis in Chapter 3 shows that the Chang-Lee scheme fails to provide proper user authentication and to preserve credential privacy since the knowledge proof of user authentication guarantees neither soundness nor credential privacy.

1.3.3 Biometric Authentication

To prevent the inherent drawbacks of passwords, biometric authentication has been proposed. In 1999, Juels and Wattenberg [JW99] proposed the first fuzzy commitment scheme, using Hamming distance to tolerate errors. Later, in 2002, Juels and Sudan [JS02] introduced a provably secure fuzzy vault scheme, in which a user chooses a long-bit secret key (treated as a biometric key) and hides it using the user’s biometric template. The fuzzy vault scheme uses Euclidean distance measurement to tolerate errors. One year after, Clancy et. al [Cla03] proposed
a secure smart card-based fingerprint authentication scheme, which was based on Juels and Sudan’s fuzzy vault. In 2007, Nandakumar et al \cite{NJP07} proposed a fully automatic implementation by employing a fuzzy vault, using helper data to align unidentified fingerprints accurately. The improved scheme used both location and orientation attributes \((x, y, \theta)\) of a minutia point to record the biometric data, where \((x, y)\) is the row and column that indicate the location in the image, and \(\theta\) is the orientation in respect to the X-axis. The helper data are high curvature points extracted from the fingerprint orientation field, thus it neither affects the security nor leaks any information about the biometric template. Fuzzy vault has been widely accepted since the Euclidean distance measurement is suitable for the majority of biometric applications \cite{WQ10}.

Another famous scheme by Dodis et al. \cite{DRS04}, called ‘fuzzy extractor’, generates a pair including a secret key and a public key directly from the user’s biometric template and uses Hamming distance, set difference and edit distance to tolerate errors. Other interesting works are briefly reviewed as follows. In 2008, Teoh and Ong \cite{AT08} proposed a randomized dynamic quantization transformation (RDQT), which is based on fuzzy commitment, to binarize biometric data, satisfying randomness and uniqueness. Meanwhile, Sheng et al. \cite{SHFD08} presented a template-free biometric-key generation, which can also generate a key directly from biodata.

### 1.3.4 Three-Factor Authentication

To achieve stronger security requirements, three-factor authentication has been introduced since the biometric features may not be completely private. In 2003, Kim et al. \cite{KLY03} proposed two ID-based password authentication schemes, using smart card and fingerprints, without the use of public key directory tables. However, Scott \cite{Sco04} pointed out that a passive eavesdropper without access to any smart cards, passwords, or fingerprints, could impersonate any identity to log in to the server after successfully eavesdropping legitimate log-on only once.

In 2004, Uludag et al. \cite{UJP04} surveyed various types of biometric cryptosystems, and they recommended using digital rights management (DRM) systems \cite{JM03} to address the problems of biometric cryptosystems. In their methods, the cryptographic key is bound with biometric template then stored in a database. Thus, the key cannot be revealed without passing biometric authentication. However, the requirement of the biometric database has increased the cost and put
users’ privacy at risk. To protect users’ privacy, in 2006 Bhargav-Spantze et al. [BSSB06, BSSM+07] proposed a novel privacy preserving two-phase multi-factor authentication scheme with biometrics, based on zero knowledge proof (ZKP), in which, user privacy is preserved by using the Petersen commitments. However, the scheme is very costly because the modular exponentiations and the database of all users’ commitments are stored on the server side. In 2009, Fan and Lin [FL09] constructed an efficiency enhancing and privacy preserving three-factor authentication scheme, but it did not support free password changing and it had flaws in the formal proof. In their security proof, Theorem 2 defines that the protocol is a secure key exchange scheme if the public-key encryption scheme used in the protocol is secure against CCA2; however, in step 3 of the protocol, the session key material \( v \) is encrypted in a symmetric key scheme, and the session key \( h(v) \) is only decided by the server, where \( h(\cdot) \) is a hash function. Thus, if the symmetric key encryption scheme is insecure, then the protocol cannot provide secure key exchanging.

Recently, Li and Hwang [LH10] proposed an efficient biometric-based remote user authentication scheme using smart cards, without synchronized clocks. Later, Li et al. [LNM+11] pointed out that the Li-Huang scheme does not provide proper authentication since the scheme is vulnerable to man-in-the-middle attack. To address this shortcoming, they presented a further improvement. In 2011, however, Das et al. [Das11] found Li et al.’s improved scheme neither provided strong authentication nor supported easy password change. They then presented an improvement on Li et al.’s scheme. Our analysis, however, shows this scheme is vulnerable to the off-line guessing password attack. An adversary who has a smart card, can extract \( f_i, r_i, N \) from the smart card, where \( f_i = h(BioData_i), r_i = h(h(N||PW_i)||f_i) \). Then, the adversary can crack the user’s password by matching \( r_i = h(h(N||PW_i)||f_i) \) for every different \( PW_i \) in the password range.

In 2011, Huang et al. [HXC+11] proposed a generic framework for three-factor authentication, preserving security and privacy. The basic idea is to use fuzzy extractor to generate the biometric key from the biometric templates, and run twice a underlying two-factor authentication scheme. In the first time it runs the two-factor scheme as normal, and in the second time it uses the biometric key to replace the password and runs the underlying scheme again, thus achieving a three-factor scheme. This framework does not require any additional mechanism to enhance the underlying two-factor authentication protocol, and in the derived scheme, users need not show their biometric features to the server, and servers need not store any
user information on a database. Thus, privacy is preserved and cost is reduced.

1.4 Challenges

The need for authentication of individual identity is a fundamental requirement in our society. In this computer age, single sign-on is a highly desirable solution for user authentication, suiting most common users since it reduces requirements for multiple logins and for remembering multiple IDs/passwords. This also alleviates forgotten password problems. Unfortunately, there are some shortcomings in the existing schemes such as (a) the inability to preserve user anonymity properly; (b) vulnerability to possible attacks, e.g. impersonation attacks; (c) a seeming absence of formal study and proof on soundness of the single sign-on; (d) the requirement for additional time-synchronized mechanisms; (e) lower efficiency and higher cost. Thus, it is a challenge to design an efficient and provably secure single sign-on scheme in distributed networks.

For users who have higher security requirements, three factor authentication is an ideal solution since it incorporates all advantages of password-based authentication, two-factor authentication and biometric authentication. An ideal three-factor authentication protocol can greatly ensure information confidentiality in distributed systems. However, the existing research on three-factor authentication is far from satisfactory and has a number of problems. The literature shows, for example, that corrupting biometric data is not only a privacy issue but is also related to the security of protocols; most existing solutions, and even their improved versions, have flaws which can lead to protocol breaking. Thus, it is a challenge to design a provably secure three-factor authentication scheme which preserves privacy in complex network environments.

1.5 Aims and Objectives

This thesis provides research into remote user authentication and focuses in particular on single sign-on and three-factor authentication. The aims of this thesis are as follows:

1. In the literature, several single sign-on schemes have been proposed. However, most of them have security flaws, and even worse, their improvements are also
insecure against possible attacks. Thus, this thesis aims to give an insight into the most recent SSO schemes identifying their flaws, issues and challenges.

2. The second aim of this thesis is to formalize the single sign-on and its security model to formally resolve the issues identified. Also, an efficient and provably secure single sign-on authentication scheme without the identified drawbacks will be provided according to the formal model.

3. A generic framework for three-factor authentication is the third aim of this thesis. The framework, which is efficient and practical, can upgrade two-factor authentication schemes to three-factor authentication schemes without additional requirements on the underlying schemes, and can preserve user privacy even when interfacing with a malicious server. Also, a concrete three-factor authentication scheme with formal security proof is needed.

1.6 Organisation of The Thesis

This thesis considers the use of single sign-on and three-factor authentication in the context of distributed environments. This chapter has reviewed the literature and the importance of user authentication and discussed the challenges and aims of this research.

Chapter 2 introduces five areas of background information relevant to the current research. The first part introduces some intractable problems with special focus on the discrete logarithm problem and the Diffie-Hellman problem. The chapter then reviews some cryptographic tools, encryption mechanisms and digital signatures. Finally, zero-knowledge proof-of-knowledge is discussed in the last part.

Chapter 3 first reviews the recent Chang-Lee scheme [CL12]. Chang and Lee claimed high security but this chapter demonstrates that the scheme is actually insecure as it fails to meet credential privacy and soundness of authentication. In particular, this chapter presents two impersonation attacks which also apply to another SSO scheme proposed by Hsu and Chuang [HC09], which inspired the design of the Chang-Lee scheme. This chapter then identifies the flaws in their security arguments to explain why attacks are possible against their SSO scheme. Moreover, by employing the efficient verifiable encryption of RSA signatures (RSA-VES) as proposed by Ateniese [Ate99], this chapter proposes an improvement for repairing the
1.6. Organisation of The Thesis

Chang-Lee scheme. In addition, the formal study of the soundness of authentication has been identified as one open problem.

Chapter 4 formalizes the security model of the single sign-on schemes with authenticated key exchange. In particular, this chapter points out the difference between soundness and credential privacy, and combines them both into one definition. This part also proposes a provably secure single sign-on authentication scheme which satisfies soundness, preserves credential privacy, meets user anonymity and supports session key exchange. The proposed scheme is very efficient so that it is suitable for mobile devices in distributed systems and networks.

Chapter 5 reviews and improves a generic framework for three-factor authentication proposed by Huang et. al [HXC+11] and then proposes a provably secure concrete instantiation according to the improved framework. Before reviewing Huang’s framework, this part discusses two biometric identification schemes. Then, based on the discussion, this chapter suggests improvements to Huang’s framework by employing fuzzy vault as first proposed by Juels and Sudan [JS02]. In addition, a concrete scheme is given by incorporating fuzzy vault and Yang’s scheme [YWWD08] via the improved framework. This chapter also provides the practicability analysis of the derived scheme, then compares the scheme with other existing three-factor schemes and lastly, it also provides a formal security proof and a privacy discussion of the concrete instantiation.

Finally, Chapter 6 concludes this thesis with a summary of our proposed contributions, future work and new open problems for future research.
Chapter 2

Background

This chapter introduces five areas of fundamental background knowledge: intractable problems, cryptographic tools, encryption techniques, digital signatures and zero-knowledge proof of knowledge (ZKPK).

2.1 Intractable Problems

2.1.1 Discrete Logarithm Problem

The discrete logarithm problem \([\text{Mao04}]\) is a significant element in a number of theoretical problems and is the core problem at the root of many difficulties encountered in cryptographic security assumptions.

Definition 2.1. (Discrete Logarithm Problem (DLP)) In a cyclic group with generator \(g\), the DLP is defined as follows.
On input \((g, y) \in \mathbb{G}\), output \(a\) such that \(y = g^a\).

2.1.2 Diffie-Hellman Problem

The Diffie-Hellman problem (DHP) was proposed by Diffie and Hellman \([\text{DH76}]\). The DHP can be divided into two related problems: computational DHP and decisional DHP.

Computational Diffie-Hellman (CDH) Problem

Definition 2.2. In a cyclic group \(\mathbb{G}\) of order \(p\) with generator \(g\), pick integers \(a, b \in \{0, 1, ..., p - 1\}\) randomly and take \(g, g^a, g^b\) as input, the CDH problem is to compute \(g^{ab}\) without given the values of \(a\) and \(b\).
2.2. Cryptographic Tools

The CDH problem \cite{DH76} is closely related to the DLP due to the open question of whether the DLP problem can be solved if CDH has been solved in $\mathbb{G}$.

**Decisional Diffie-Hellman (DDH) Problem**

The DDH problem has been proposed as a decisional version of the CDH problem.

**Definition 2.3.** In a cyclic group $\mathbb{G}$ of order $p$ with generator $g$, pick integers $a, b, z \in \{0, 1, \ldots, p - 1\}$ randomly and given two distributions $(g, g^a, g^b, g^z)$ and $(g, g^a, g^b, g^{ab})$, the DDH problem is to distinguish these two distributions. In other words, the problem is to decide whether $g^z = g^{ab}$ without knowing $a$, $b$ and $z$.

## 2.2 Cryptographic Tools

### 2.2.1 Cryptographic Hash Functions

A cryptographic hash function \cite{Mao04}, $H: \{0, 1\}^* \rightarrow \{0, 1\}^k$, is an algorithm which outputs the fixed $k$-length string for any arbitrary length input and has been widely employed in cryptographic schemes. In this thesis, all hash functions refer to the ideal cryptographic hash function which meets three main properties.

- The hash value is easy to compute for any given input message.
- It is unfeasible to find two distinct messages with the same hash value.
- It is unfeasible to recover a message from its hash value.

In 1986, Fiat and Shamir \cite{FS86} first proposed the random oracle model (ROM), and later, it was formalized by Bellare and Rogaway \cite{BR93b}. In the ROM, a hash function is modelled as a random oracle which is a theoretical black box. This black box answers every query with a random number selected from its output domain. In other words, the output of the hash function is treated as a randomness in the cryptographic security proof. However, one concern is that no hash function can be realized as a truly random function in the real world. Thus, some researchers have tried to prove schemes without the use of a random oracle. Despite this argument, the ROM is still popularly used in cryptographic security proofs. For example, Optimal Asymmetric Encryption Padding (OAEP) \cite{BR94a} and one-mask Diffie-Hellman key exchange (OMDHKE) \cite{BCP04} are provably secure in the ROM.
2.2.2 Time Stamp

In this thesis, the ‘time stamp’ means a digital time stamp. It is a proof showing that a digital event existed at a certain time and the event has not been changed since that time. In cryptography, it is normally employed to prevent message replay attacks. Time stamp is usually in two procedures: one is the signing procedure which binds the local clock code together with message and signs a signature on it; the other is the verifying procedure which convinces the receiver that the received message is valid only if the time stamp is being received for the first time.

The drawback of using the time stamp is the requirement of time synchronizing. This imposes restrictions on the use of time stamp. Thus, there are many schemes interested in using random nonce to achieve the same goal.

2.2.3 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange scheme is the first public key system, proposed by Diffie and Hellman in 1976 [DH76]. The scheme is widely accepted in public key systems to establish a session key. The session key enables two entities to communicate with each other over a public network with data integrity and confidentiality. The security of the Diffie-Hellman key exchange scheme is based on the computational Diffie-Hellman problem. The processes are as follows.

- **Initialization.** Two entities Alice and Bob agree on a cyclic group $G$ with a generator $g$.

- **Key Exchange.** Alice and Bob choose secret random integers $a$ and $b$ respectively, calculate their own session key materials $g^a$ and $g^b$ respectively, and then send them to each other.

- **Key Agreement.** Alice and Bob calculate the session key by $(g^b)^a$ and $(g^a)^b$, respectively. Now, they share the same session key $g^{ab}$ for further communication.

To prevent man-in-the-middle attacks, normally the session key material is bound together with the entity identity using cryptographic techniques, e.g. digital signature scheme.
2.3 Encryption Techniques

2.3.1 Symmetric Key Encryption

Symmetric key encryption is used to encrypt plaintext and decrypt ciphertext with the same secret key. The secret key is a shared secret between the sender and receiver such as a simple word, a name, or a random number. For example, the secret key can be the letters in the first column of the second page in the Bible. Since symmetric key encryption is more efficient than public key encryption, it is the favorite for protocol designing, e.g. Transport Layer Security (TLS) and Secure/Multipurpose Internet Mail Extensions (S/MIME). However, key sharing is a vulnerability of symmetric key encryption since that it requires a truly secure channel to share a key privately.

Symmetric key encryption can be classified into stream ciphers and block ciphers. The former is exceptionally fast but has a high cost and normally operates one bit at a time; the latter operates on a block of bits and has been used more frequently. An example of a stream cipher is the one-time pad [Mil82] introduced by Frank Miller in 1882. It has been proven that it is impossible to crack if used correct. RC4 is another example of a stream cipher, which was proposed by Ron Rivest of RSA Security in 1987 [Wik12a] and adopted in Secure Sockets Layer (SSL). The most well-known block cipher schemes are Data Encryption Standard (DES) which was published as FIPS PUB 46 in 1977 [oS77], Advanced Encryption Standard (AES) which was designed by Daemen and Rijmen [DR00] and published as U.S. FIPS PUB 197 in 2001 to supersede DES.

2.3.2 Asymmetric Key Encryption

Public encryption was first publicly introduced in the paper ‘New Directions in Cryptography’ [DH76] by Diffie and Hellman in 1976. Public key cryptosystems require two separate keys, one for plaintext encryption and one for ciphertext decryption. The encryption key can be published and the decryption key is kept secret.

Many classic asymmetric key encryption schemes have been widely adopted, e.g. RSA encryption [RSA78], ElGmagal encryption [ElG85], optimal asymmetric encryption padding (OAEP) [BR94b] and OAEP+ [Sho02].
RSA Encryption Scheme

The RSA cryptosystem is the best known and one of the most widely used public key cryptosystems. It was invented in 1978 by Rivest, Shamir and Adleman [RSA78]. There are two algorithms in RSA cryptosystem with two different keys for encryption and decryption. Anyone can access the encryption algorithm with a public key offered by the person who receives the messages. Thereafter, anyone can send encrypted messages (ciphertext) to the receiver. However, it is impossible to decrypt the ciphertext if only the public key is known. Thus, only the receiver who knows the private key can decrypt the ciphertext. The RSA encryption scheme consists of three parts, namely initialization, encryption and decryption.

- **Initialization**
  1. Select two distinct large primes $p$ and $q$. Here, ‘large’ means from 1024 to 2048 bits or 308 to 616 decimal digits.
  2. Calculate $n = p \cdot q$ and $\phi(n) = (p - 1) \cdot (q - 1)$.
  3. Choose a random integer $e < \phi(n)$ such that $\gcd(e, \phi(n)) = 1$ and publish the public key $(n, e)$.
  4. Compute the integer $d$ such that $e \cdot d \equiv 1 \mod \phi(n)$ and store the private key $d$.

- **Encryption.** Given a (block of) message $0 \leq m < n$, the ciphertext $c$ is the least residue of $m^e \mod n$. That is $c \equiv m^e \mod n$.

- **Decryption.** To decrypt the ciphertext $c$, the plaintext can be recovered by calculating the least residue of $c^d \mod n$. That is $m \equiv c^d \mod n$, where $0 \leq m < n$.

The security of RSA closely related to the computationally unfeasible problem of large integer factorization. Informally, to decrypt $c$, we need private key $d$ which was calculated by using the Euclidean algorithm with public key $e$ and secret $\phi(n)$. Thus, we cannot get the private key without $\phi(n)$ as well as $(p, q)$. Hence, to decrypt $c$ without knowing $d$, we must factorize $n$. In the initialization, the $p$ and $q$ are large primes (1024-2048 bits) and thus, $n$ is about 2048-4096 bits. Therefore, it is computationally unfeasible under today’s knowledge to crack a well set-up RSA cryptosystem.
2.4 Digital Signatures

The digital signatures were introduced by Diffie and Hellman [DH76] and first formalized by Goldwasser, Micali and Rivest [GMR88]. It was invented to authenticate a signer of messages or documents, and to ensure that the messages have not been modified.

2.4.1 Formal Definition

**Definition 2.4.** (Digital signature) A digital signature scheme consists of three algorithms:

1. **KeyGen(λ)**. Takes security parameter λ as input, outputs verifying/signing key pair (PK, SK) for a signer.

2. **SGen(m, SK)**. Takes message m and signing key SK of a signer as input, and outputs a signature σ on message m.

3. **SVer(m, σ, PK)**. Takes signer’s public key PK, message m and signature σ as input, and outputs ‘valid’ iff the σ is signed by the signer on message m. Otherwise, it outputs ‘invalid’ for rejection.

Formally, a signature scheme is called existentially unforgeable if any PPT adversary A can only win the following game, called Game-UFCMA, with a negligible probability [GMR84, GMR88].

**Definition 2.5.** (Game-UFCMA) The Game-UFCMA has three phases which are defined as follows:

- **Initialization** (PK, SK) ← KeyGen(λ). Given a security parameter λ, a verifying/signing key pair is generated by the key generation algorithm and adversary A is given the verifying key PK.

- **Query** σᵢ ← SGen(SK, mᵢ). A runs up to qᵦ sign times to ask the signature signing oracle in an adaptive manner. Each time, the signing oracle will reply a signature σᵢ for each message mᵢ chosen by A, where 1 ≤ i ≤ qᵦ sign.

- **Forge** A outputs a new message and signature pair (m, σ). A wins if

  1. SVer(pk, m, σ) = 1, i.e., σ is a valid signature for message m under the public key PK.
2.4. Digital Signatures

2. $m \neq m_i$, for any $i \in \{1, \cdots, q_{\text{sign}}\}$.

2.4.2 RSA Signature Scheme

Since the concept of the digital signature was invented by Diffie and Hellman [DH76], many signature schemes have been proposed. The RSA signature scheme [RSA78] may be the earliest scheme, which comprises three algorithms defined as follows.

- $\text{KeyGen}(\lambda)$. Refer to the initialization phase of the RSA encryption scheme in 2.3.2.

- $\text{SGen}(m, d)$. To sign a message $m$, the signer generates the signature $\sigma$ by computing $\sigma = h(m)^d \mod n$.

- $\text{SVer}(m, \sigma, e)$. Given a message $m$ with a signature $\sigma$, the $\text{SVer}$ outputs valid iff $h(m) = \sigma^e \mod n$. Otherwise, it outputs invalid.

The primitive RSA signature scheme is not secure against certain attacks, e.g. common-modulus attacks against RSA [DK02]. The Common-Modulus Attack has been aimed at the case where two or more users of the RSA cryptosystem share the same RSA modulus $n$, which leads to (a) a user’s secret key being able to recovered by another user; (b) a user factoring $n$; and (c) an attacker recovering the plaintext.

2.4.3 Schnorr Signature Scheme

As one of most frequently used signature schemes, the Schnorr signature scheme [Sch89, Sch91] is provably secure in a random oracle model under the assumption that the discrete logarithm problem is intractable [BP02, PS96, PS00, Mao04]. We now review the Schnorr signature scheme as follows.

- $\text{KeyGen}(\lambda)$. The scheme is defined in a cyclic group $G$ of order $q$ with a generator $g \in \mathbb{Z}_p^*$, where $p$ and $q$ are primes such that $q|p - 1$, $q \geq 2^{160}$, and $p \geq 2^{1024}$. A secure hash function $h(\cdot)$ is also selected. The private key is $x$ choosing from $\mathbb{Z}_q^*$, and the public key is $y = g^x \mod p$.

- $\text{SGen}(m, x)$. To sign message $m$ with private key $x$, a signer picks a random integer $r \in \mathbb{Z}_q^*$ and outputs the signature $\sigma = (a, e, s)$ by computing $a = g^r \mod p$, $e = h(a, m)$ and $s = r + x \cdot e \mod q$. 

2.5 Zero-Knowledge Proof of Knowledge

Zero-knowledge proof was proposed by [GMR85] and discussed in detail in [GMR89]. It is an interactive protocol which enables a prover to convince a verifier the truth of an assertion, without revealing anything but the validity of proof. The zero-knowledge proof should satisfy three properties, namely completeness, soundness and zero-knowledge (ZK-ness). The completeness guarantees that the verifier will be convinced by a prover if the statement is true. The soundness ensures that the verifier will never be convinced by any prover if the statement is false. The ZK-ness requires that verifier can learn nothing but the fact.

Soon afterwards, a noninteractive zero-knowledge (NIZK) proof [CP92] was proposed and the proof of knowledge [BG92] was introduced. The proof of knowledge is an interactive proof, enables a prover to convince a verifier that he knows some secrets without showing the secrets to the verifier. If the proof of knowledge also satisfies the properties of zero knowledge proof, then it can be called zero-knowledge proof of knowledge (ZKPK).

2.5.1 RSA-based Verifiable Encryption of Signatures (RSA-VES)

Verifiable encryption of signatures (VES) was proposed in 1999 for fair exchange. VES comprises three parties, two users (namely Alice and Bob) and a trusted party. The basic idea of VES is that Alice who has a key pair of signature schemes signs a signature on a contract, encrypts it using the trusted party’s public key, and uses the noninteractive signature-based proof of knowledge protocol [CP92] to convince Bob that she has encrypted the signature in the ciphertext and the trusted party
can recover it from the proof materials. After validating the proofs, Bob sends his signature which is also on the contract to Alice and expects the signature from Alice. For the purpose of fair exchange, Alice should send her signature back to Bob after accepting Bob’s signature. If she does not do so, Bob can also get her signature by sending Alice’s proof materials together with his own signature to the trusted party who then recovers Alice’s signature and sends it to Bob, and in the meanwhile, forwards Bob’s signature to Alice. Thus, the fair exchange is achieved.

In this thesis, we consider the case of RSA-VES such that the signer is working over a cyclic subgroup with unknown order, but the length of this order is publicly known. The RSA-VES, which is reviewed as follows, will be used in the chapter 3.

**Initialization**

Alice selects two large safe primes \( p \) and \( q \) to set \( n = pq \). Namely, there are two primes \( p' \) and \( q' \) such that \( p = 2p' + 1 \) and \( q = 2q' + 1 \). Alice then computes her public key \((e, n)\) where \( e > 2 \) is a prime, and her private key \( d \) such that \( ed \equiv 1 \mod 2p'q' \). Alice also need to choose a cryptographic hash function \( h(\cdot) \) such that \( h : \{0, 1\}^* \rightarrow \{0, 1\}^k \), where \( 160 \leq k \leq n - 1 \). Now, Alice sends \((e, n)\) to the trusted party and publishes \((e, n, h(\cdot))\).

Let \( \mathbb{Q}_n \) be the subgroup of squares in \( \mathbb{Z}_N^* \), whose order \( \#G = p'q' \) is unknown to the public but its bit-length \( l_G = |N| - 2 \) is publicly known.

Upon receiving \((e, n)\) from Alice, the trusted party checks its validity and randomly selects a \( g \in \mathbb{Z}_N^* \) if \((e, n)\) is valid public key of Alice. To control the tightness of the ZK proof, a security parameter \( \epsilon > 1 \) is chosen. Then, the trusted party randomly selects a secret key \( x \), computes and sends public parameters \( g = g^2 \mod n \) and \( y = g^x \mod n \) to Alice. Finally, the trusted party publishes \((\epsilon, g, g, y)\).

**Proof Generation**

First, Alice need to sign an RSA-based signature on message \( m \) by computing \( \sigma = h(m)^{2d} \mod n \). To generate a proof of this signature, Alice first encrypts it as \( K_1 = \sigma \cdot y^r \mod n \) and \( K_2 = g^r \mod n \), where \( r \) is a random integer with binary length \( l_G \). Secondly, Alice computes two commitments \( a = (y^e)^{r_1} \mod n \) and \( b = g^{r_1} \mod n \), where \( r_1 \) is a also random number such that \( r_1 \in \pm\{0, 1\}^{\epsilon(l_G+k)} \). Then, Alice computes the last part of proof \((c, s)\) as \( c = h(m||y^{2r}||K_2||y^e||g||a||b) \) and \( s = r_1 - c \cdot r \) (in \( \mathbb{Z} \)). Finally, Alice sends the proof \( P = (K_1, K_2, a, b, c, s) \) as the
2.5. Zero-Knowledge Proof of Knowledge

whole proof to Bob.

Proof Verification

To verify the proof $P$, Bob calculates $W = \frac{K_i}{h(m)^2} \mod n$, $a' = (y^e)^s \cdot (W)^c \mod n$, $b' = g^s \cdot K_i^c \mod n$, and checks whether $(c, s) \in \{0, 1\}^k \times \pm \{0, 1\}^{(lG + k) + 1}$ and $c = h(m||W||K_2||y^e||g||a'||b')$ holds. If it does hold, then Bob signs a signature $\sigma'$ on hashed message $h(m)$ and sends it to Alice.

Fair Exchange

The ZKPK is achieved until the last step. However, the RSA-VES was proposed for fair exchange. Thus, after Bob sends his signature to Alice, a signature from Alice is expected. If he has not received it, Bob can also obtain the signature with the help of the trusted party by the following steps.

First, Bob sends message $m$ together with his signature $\sigma'$ and the encrypted signature $(K_1, K_2)$ to the trusted party. The trusted party verifies Bob’s signature first and if it is valid, then decrypts the signature by computing $\sigma = \frac{K_1}{K_2}$. Finally, the trusted party sends $\sigma$ to Bob and redirects $\sigma'$ to Alice. Thus, fair exchange is achieved.
Chapter 3

Cryptanalysis of A Secure Single Sign-On Scheme

3.1 Introduction

With the wide spread use of distributed computer networks, it has become common to allow users to access various network services offered by distributed service providers [BX11]. Consequently, user authentication (also called user identification) [Lam81, LC00] plays a crucial role in distributed computer networks to verify if a user is legal and can therefore be granted access to the services requested. To avoid bogus servers, users usually need to authenticate service providers. After mutual authentication, a session key may be negotiated to keep the confidentiality of the data exchanged between a user and a service provider [LC00, JW09]. In many scenarios, the anonymity of legal users must be protected as well [LC00]. However, practice has shown that it is a big challenge to design efficient and secure authentication protocols with these security properties in complex computer network environments [CPS11].

Single sign-on (SSO) is a new authentication mechanism that enables a legal user with a single credential to be authenticated by multiple service providers in distributed computer network. Chang and Lee [CL12] made a careful study of the SSO mechanism. First, they argued that the Hsu-Chuang user identification scheme, actually an SSO scheme, has two weaknesses: (a) An outsider can forge a valid credential by mounting a credential forging attack since the Hsu-Chang scheme employs naive RSA signature without any hash function to issue a credential for any random identity selected by a user (In fact, this feature based on [YWB+04].); and (b) the Hsu-Chuang scheme requires clock synchronization since it uses a time stamp. Then, Chang and Lee presented an interesting RSA-based SSO scheme, which is highly efficient in computation and communication (So it is suitable for mobile...
3.2 Review of the Chang-Lee Scheme

The Chang-Lee single sign-on scheme [CL12] is a remote user authentication scheme, supporting session key establishment and user anonymity. In their scheme, RSA cryptosystems are used to initialize a trusted authority, called an SCPC (smart devices), and does not rely on clock synchronization by using a nonce instead of a time stamp. Finally, they presented a well-organized security analysis to show that their SSO scheme supports secure mutual authentication, session key agreement, and user anonymity.

This chapter, however, will demonstrate that their scheme is actually insecure as it fails to meet credential privacy and soundness of authentication. Specifically, we show that the Chang-Lee scheme [CL12] is actually insecure by presenting two impersonation attacks, i.e., credential recovering attack and impersonation attack without credentials. In the first attack, a malicious service provider who has communicated with a legal user twice can successfully recover the user’s credential. Then, the malicious service provider can impersonate the user to access resources and services provided by other service providers. The other attack may enable an outside attacker without any valid credential to impersonate a legal user or even a nonexistent user to have free access to the services. These two attacks imply that the Chang-Lee SSO scheme fails to meet credential privacy and soundness, which are essential requirements for SSO schemes and authentication protocols. We also identify the flaws in their security arguments in order to explain why it is possible to mount our attacks against their scheme. Similar attacks can also be applied to the Hsu-Chuang scheme [HC09], on which the Chang-Lee scheme is based. Finally, to avoid these two impersonation attacks we propose an improved SSO scheme to enhance the user authentication phase of the Chang-Lee scheme. To this end, we employ the efficient RSA-based verifiable encryption of signatures (VES) proposed by Ateniese [Aten04] to verifiably and securely encrypt a user’s credential. In fact, Ateniese’s VES was originally introduced to realize fair exchange.

The rest of this chapter is organized as follows. The next section reviews the Chang-Lee scheme [CL12]. After that, we present two attacks against the Chang-Lee scheme in Section 3.3 and briefly analyse the Hsu-Chuang scheme [HC09] in Section 3.4. Then, the improved SSO scheme using VES is given in Section 3.5. Finally, Section 3.6 draws some conclusions.

3.2 Review of the Chang-Lee Scheme

The Chang-Lee single sign-on scheme [CL12] is a remote user authentication scheme, supporting session key establishment and user anonymity. In their scheme, RSA cryptosystems are used to initialize a trusted authority, called an SCPC (smart
3.2. Review of the Chang-Lee Scheme

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCPC</td>
<td>The trusted authority</td>
</tr>
<tr>
<td>$U_i, P_j$</td>
<td>User and Service provider, respectively</td>
</tr>
<tr>
<td>$ID_i, ID_j$</td>
<td>The unique identity of $U_i$ and $P_j$, respectively</td>
</tr>
<tr>
<td>$e_X, d_X$</td>
<td>The public/private RSA key pair of identity $X$</td>
</tr>
<tr>
<td>$S_i$</td>
<td>The credential of $U_i$ created by SCPC</td>
</tr>
<tr>
<td>$S_x$</td>
<td>The long term private key of SCPC</td>
</tr>
<tr>
<td>$S_y$</td>
<td>The public key of SCPC</td>
</tr>
<tr>
<td>$E_K(M)$</td>
<td>A symmetric key encryption of plaintext $M$ using a key $K$</td>
</tr>
<tr>
<td>$D_K(C)$</td>
<td>A symmetric key decryption of ciphertext $C$ using a key $K$</td>
</tr>
<tr>
<td>$\sigma_j(M, SK_j)$</td>
<td>The signature $\sigma_j$ on $M$ signed by $P_j$ with signing key $SK_j$ via algorithm $SGen(\cdot)$</td>
</tr>
<tr>
<td>$SVer(M, \sigma_j, PK_j)$</td>
<td>The verifying of signature $\sigma_j$ on $M$ with public key $PK_j$</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>A given one way hash function</td>
</tr>
<tr>
<td>$||$</td>
<td>The operation of concatenation</td>
</tr>
</tbody>
</table>

Table 3.1: Notations in the Chang-Lee Scheme

The trusted authority SCPC first selects two large safe primes $p$ and $q$, and then sets $N = pq$. After that, SCPC determines its RSA key pair $(e, d)$ such that $ed = 1$. The Diffie-Hellman key exchange technique is employed to establish session keys. In the Chang-Lee scheme, each user $U_i$ applies a credential from the trusted authority SCPC, who signs an RSA signature for the user’s hashed identity. After that, $U_i$ uses a kind of knowledge proof to show that he/she is in possession of the valid credential without revealing his/her identity to eavesdroppers. Actually, this is the core idea of user authentication in their scheme and also the reason why their scheme fails to achieve secure authentication as we shall show shortly. On the other side, each $P_j$ maintains its own RSA key pair for doing server authentication. The Chang-Lee SSO scheme consists of three phases: system initialization, registration, and user identification. Table 3.1 explains notations, and the details of the Chang-Lee scheme are reviewed as follows.

### 3.2.1 System Initialization Phase

The trusted authority SCPC first selects two large safe primes $p$ and $q$, and then sets $N = pq$. After that, SCPC determines its RSA key pair $(e, d)$ such that $ed = 1$.
3.2. Review of the Chang-Lee Scheme

mod $\phi(N)$, where $\phi(N) = (p-1)(q-1)$. SCPC chooses a generator $g \in \mathbb{Z}_n^*$, where $n$ is also a large prime number. Finally, SCPC publishes $(e, g, n, N)$, keeps $d$ as a secret, and erases $(p, q)$ immediately once this phase has been completed.

3.2.2 Registration Phase

In this phase, each user $U_i$ chooses a unique identity $ID_i$ with a fixed bit-length, and sends it to SCPC. After that, SCPC will return $U_i$ the credential $S_i = (ID_i || h(ID_i))^d \mod N$, where $||$ denotes a concatenation of two binary strings and $h(\cdot)$ is a collision-resistant cryptographic one-way hash function. Here, both $ID_i$ and $S_i$ must be transferred via a secure channel.

At the same time, each service provider $P_j$ with identity $ID_j$ must maintain its own RSA public parameters $(e_j, N_j)$ and private key $d_j$ as does by SCPC.

3.2.3 User Identification Phase

To access the resources of service provider $P_j$, user $U_i$ needs to go through the authentication protocol specified in Fig. 3.1. Here, $k$ and $t$ are random integers chosen by $P_j$ and $U_i$ respectively; $n_1$, $n_2$ and $n_3$ are three random nonces; and $E(\cdot)$ denotes a symmetric key encryption scheme which is used to protect the confidentiality of user $U_i$’s identity $ID_i$. We highlight this phase as follows.

- Upon receiving service request message $m_1$ from user $U_i$, service provider $P_j$ generates and returns user message $m_2$ which is made up primarily by its RSA signature on $(Z, ID_j, n_1)$. Once this signature is validated, it means that user $U_i$ has authenticated service provider $P_j$ successfully. Here, $Z = g^k \mod n$ is the temporal Diffie-Hellman (DH) key exchange material issued by $P_j$.

- After that, user $U_i$ correspondingly generates his/her temporal DH key exchange material $w = g^t \mod n$ and issues proof $x = S_i^{h(K_{ij}||w||n_2)}$, where $K_{ij} = h(ID_i||k_{ij})$ is the derived session key and $k_{ij} = Z^t \mod n = w^k \mod n$ is the raw key obtained by using the DH key exchange technique.

- Proof $x = S_i^{h(K_{ij}||w||n_2)}$ is used to convince $P_j$ that $U_i$ does hold valid credential $S_i$ without revealing the value of $S_i$. Namely, after receiving message $m_3$ service provider $P_j$ can confirm $x$’s validity by checking if $SID_i^{h(K_{ij}||w||n_2)} \mod N = x^e$
3.2. Review of the Chang-Lee Scheme

\[ Z = g^k \mod n \]
\[ u = h(Z \parallel ID_j \parallel n_1) \]
\[ v = (u \parallel h(u))^{d_j} \mod N_j \]

\[ k_{ij} = Z^j \mod n \]
\[ K_{ij} = h(ID_j \parallel k_{ij}) \]
\[ w = g^t \mod n \]
\[ x = g^{h(K_{ij} || v || n_2)} \mod n \]
\[ y = E_{K_{ij}}(ID_i \parallel n_3 \parallel n_2) \]

\[ m_4 = (w, x, y) \]

\[ k_{ij} = w^k \mod n \]
\[ K_{ij} = h(ID_j \parallel k_{ij}) \]
\[ (ID_i \parallel n_3 \parallel n_2) = D_{K_{ij}}(y) \]
\[ SID_i = (ID_i \parallel h(ID_i)) \]
\[ SID_i^{h(K_{ij} || v || n_2)} \mod n \]
\[ V = h(n_3) \]

\[ m_3 = (V) \]

\[ m_2 = (Z, v, n_2) \]

\[ m_1 = (\text{Req}, n_i) \]

Figure 3.1: User Identification Phase of the Chang-Lee Scheme

mod \( N \), where \( SID_i = (ID_i \parallel h(ID_i)) \). Once this quality holds, it means that user \( U_i \) has been authenticated successfully by service provider \( P_j \). It is worth noting that proof \( x \) is designed in a particular way so that except \( P_j \) and \( U_i \), no one else can verify it as both \( U_i \)'s identity \( ID_i \) and the newly established session key \( K_{ij} \) are used to produce \( x \). This aims to achieve user anonymity as no eavesdropper can learn the values of \( ID_i \) and \( K_{ij} \).

- Finally, message \( m_4 \) (i.e. \( h(n_3) \)) is employed to show that \( P_j \) has obtained message \( m_3 \) correctly, which implies the success of mutual authentication and session key establishment.
3.3 Attacks Against the Chang-Lee Scheme

As can be seen from the above, it seems that the Chang-Lee SSO scheme achieves secure mutual authentication since server authentication is done by using traditional RSA signature issued by service provider $P_j$ and without valid credential $S_i$ it looks impossible for an attacker to impersonate a legal user $U_i$ by going through the user authentication procedure.

It can be seen from the following, however, that the Chang-Lee scheme is actually not a secure SSO scheme because there are two potential effective and concrete impersonation attacks. The first attack, the ‘credential recovering attack’, compromises the credential privacy in the Chang-Lee scheme as a malicious service provider is able to recover the credential of a legal user. The other attack, an ‘impersonation attack without credentials’, demonstrates how an outside attacker may be able to freely make use of resources and services offered by service providers, since the attacker can successfully impersonate a legal user without holding a valid credential and thus violate the requirement of soundness for an SSO scheme. In real life, these attacks may put both users and service providers at high risk.

We now first describe our attacks together with the assumptions required, justify why these assumptions are reasonable, and finally discuss why the security analysis and proofs given in [CL12] are not enough to guarantee the security of the Chang-Lee SSO scheme.

3.3.1 Credential Recovering Attack

Intuitively, the Chang-Lee SSO scheme seems to satisfy the requirement of credential privacy since receiving credential proof $x = S_i^{h_2} \mod N$, where $h_2$ denotes $h(K_{ij}||w||n_2)$, does not allow service provider $P_j$ to recover user $U_i$’s credential $S_i$ by computing $S_i = x^{h_2^{-1}} \mod N$, where $h_2^{-1}$ refers to $h_2^{-1} \mod \phi(N)$. In fact, the difficulty of calculating $h_2^{-1}$ from the given $(e, N, x, h)$ is the exact rationale why the RSA cryptosystem is secure, i.e., it should be intractable for an attacker to derive the RSA private key from the public key (and a given ciphertext). This is because here we could treat $(h_2, h_2^{-1})$ as another RSA public/private key pair w.r.t the same RSA modulus $N$. Moreover, directly recovering $S_i$ from $x = S_i^{h_2} \mod N$ also looks impossible as this seems equivalent to decrypting the RSA ciphertext $x$ w.r.t. the (ephemeral) public key $h_2$. 
Nevertheless, there is a pitfall in the production of proof $x = S_i^{h_2} \mod N$ as here the same credential $S_i$ is encrypted multiple times under different (ephemeral) public keys $h_2 \text{ w.r.t.} \text{ the same RSA modulus } N$. Consequently, under the assumption that malicious service provider $P_j$ has run the Chang-Lee SSO scheme with the same user $U_i$ twice, $P_j$ will be able to recover $U_i$’s credential $S_i$ with high probability by using the extended Euclidean algorithm. Namely, $P_j$ can solve $S_i$ from two equations $x = S_i^{h_2} \mod N$ and $x' = S_i^{h_2'} \mod N$. The details of the attack, which share some features of common-modulus attacks against RSA \cite{DK02}, are given as follows:

1. After successfully running the Chang-Lee SSO scheme twice with the same user $U_i$, malicious service provider $P_j$ stores all messages exchanged in these two instances, denoted as $(ID_i, x, K_{ij}, w, n_2, \ldots)$ for the first instance, and $(ID_i, x', K_{ij}', w', n_2', \ldots)$ for the second instance.

2. By denoting $h_2 = h(K_{ij}||w||n_2)$ and $h_2' = h(K_{ij}'||w'||n_2')$, $P_j$ first checks if $h_2$ and $h_2'$ are co-prime, i.e. if $\gcd(h_2, h_2') = 1$. In the case that $\gcd(h_2, h_2') = 1$, $P_j$ then runs the extended Euclidean algorithm to compute two integers $a$ and $b$ such that $a \cdot h_2 + b \cdot h_2' = 1$ (in $\mathbb{Z}$). Finally, malicious $P_j$ can recover $U_i$’s credential $S_i$ by computing

$$S_i = x^a \cdot x'^b \mod N. \quad (1)$$

Eq. (1) is justified by the following equalities:

$$x^a \cdot x'^b \mod N = (S_i^{h_2})^a \cdot (S_i^{h_2'})^b \mod N$$

$$= S_i^{a \cdot h_2 + b \cdot h_2'} \mod N$$

$$= S_i^1 \mod N$$

$$= S_i.$$ 

3. If $\gcd(h_2, h_2') \neq 1$, $P_j$ needs to run more instances with $U_i$ so that it can get two instances such that $\gcd(h_2, h_2') = 1$.

There are a number of comments to be made regarding the above attacks. First, it has a success rate of about 60% due for two reasons: (a) for two randomly selected integers $u$ and $v$, the probability that $\gcd(u, v) = 1$ holds is $6/\pi^2 \approx 0.6$ \cite{Ten95, Wei}; and (b) as the outputs of hash function $h$, $h_2$ and $h_2'$ can be regarded as random numbers. This means that after executing the Chang-Lee SSO scheme with the
same user $U_i$ twice, malicious $P_j$ will be able to recover $U_i$’s credential $S_i$ with a probability of about 0.6. Consequently, it is easy to see that after running the scheme with $U_i$ a couple of times, $P_j$ can recover $S_i$ almost certainly. Second, it is not hard to see that the above attack could be mounted by two or multiple malicious service providers who collude together once they put the values of $h_2$ together. Finally, the attack will lead to serious consequences since after recovering the valid credential of a legal user, malicious $P_j$ can impersonate this user by running Chang-Lee SSO scheme in the same way as a legal user does to freely make use of the services offered by other service providers.

How could service provider $P_j$ be malicious and then mount the above attack? On the one hand, the Chang-Lee SSO scheme specifies that SCPC is the trusted party (refer to Section IV A [CL12]). So, this implies that service providers are not trusted parties and that they could be malicious. By agreeing with Yang et al. [YWB+04], when they said that “the Wu-Hsu’s modified version could not protect the user’s token against a malicious service provider, ...”, [CL12] also implicitly agrees that there is the potential for attacks from malicious service providers against SSO schemes. Moreover, if all service providers are assumed to be trusted, to identify him/herself user $U_i$ can simply encrypt his/her credential $S_i$ under the RSA public key of service provider $P_i$. Then, $P_i$ can easily decrypt this ciphertext to get $U_i$’s credential and verify its validity by checking if it is a correct signature issued by SCPC. In fact, such a straightforward scheme with strong assumption is much simpler, more efficient and has better security, at least against this type of attack.

On the other hand, according to the security models given in [YWB+04] and [HMSY10], malicious service providers could be attackers in SSO schemes. In fact, this is a traditional as well as prudential way to deal with trustworthiness, since we cannot simply assume that beside the trusted authority SCPC, all service providers are also trusted. The basic reason is that assuming the existence of a trusted party is the strongest supposition in cryptography but it is usually very costly to develop and maintain. In particular, Han et al. [HMSY10] defined collusion impersonation attacks as a way to capture the scenarios in which malicious service providers may recover a user’s credential and then impersonate the user to login to other service providers. It is easy to see that the above credential recovery attack is simply a special case of collusion impersonation attack where a single malicious service provider can recover a user’s credential.
3.3.2 Impersonation Attack Without Credentials

We now study the soundness of the Chang-Lee SSO scheme, which seems to satisfy this security requirement as well. The main reason is that to get valid proof $x$ satisfying $SID_i^{h_2} \mod N = x^e \mod N$ for a random hash output $h_2$, there seems no other way but to compute $x$ by $x = SID_i^{h_2e^{-1}} \mod N$, i.e., $x = (SID_i^e)^{h_2}$ or $x = (SID_i^{d_i})^{h_2} \mod N$. Therefore, an attacker should not be able to log in to any service provider if it does not have the knowledge of either SCPC’s RSA private key $d$ or user $U_i$’s credential $S_i$.

Again, however, such a plausible discussion simply explains the rationale of the Chang-Lee SSO scheme but cannot guarantee its security w.r.t. the soundness. This is also the essential reason why the current focus of research in information security is on formal proofs which rigorously show the security of cryptosystems. Indeed, no one can formally prove that without knowing either SCPC’s RSA private key $d$ or user $U_i$’s credential $S_i$, it is unfeasible to compute a proof $x$ that passes through authentication, as an outside attacker is able to get a shortcut if the SCPC’s RSA public key $e$ is a small integer so that $|e| < |h(\cdot)|$. The attack is explained in detail as follows:

1. To impersonate legal user $U_i$ with identity $ID_i$ for accessing service provider $P_j$, an attacker $E$ first sends $P_j$ request message $m_1$ normally, as $U_i$ does.

2. Upon receiving message $m_2$ from $P_j$, $E$ then checks $P_j$’s signature and chooses a random integer $t$ to compute $(k_{ij}, K_{ij}, w)$. Before moving on to the next step, attacker $E$ needs to check whether $h(K_{ij}||w||n_2)$ is divisible by $e$. If not, $E$ has to choose another $t$ or start a new session to satisfy this condition.

3. As $h(K_{ij}||w||n_2)$ is divisible by $e$, let $h(K_{ij}||w||n_2) = e \cdot b$ for some integer $b \in \mathbb{Z}$. Now, $E$ computes $x$ by $x = SID_i^b$, where $SID_i = ID_i||h(ID_i)$

4. Finally, $E$ can impersonate user $U_i$ to pass the authentication by sending $m_3 = (w, x, y)$ to $P_j$, since $P_j$ will notice that $SID_i^{h(K_{ij}||w||n_2)} \mod N = x^e \mod N$. This is because we have: $SID_i^{h(K_{ij}||w||n_2)} \mod N = SID_i^{e} \mod N = x^e \mod N$.

There are a number of things worth noting in regard to the above impersonation attack without credentials. First, the attack will succeed at a rate of about $1/e$ for one random number $t$ in a new session. The reason is that $e| h(K_{ij}||w||n_2)$ holds
with a probability of about $1/e$, since $|e| < |h(\cdot)|$ and the output of hash function $h$ can be treated as random numbers. Consequently, if $e = 3$ the above attack can succeed once by trying about three values of $t$ on average. Even if $e$ is as large as $65537(= 2^{16} + 1)$, trying 65537 times to get a successful impersonation may not be difficult for attacker $E$ as it may explore a machine, which can be much more powerful than a mobile device, to do the computations needed for each try, i.e., two modular exponentiations and two hash evaluations. Moreover, even when timeout is introduced into the Chang-Lee scheme it may be not a real obstacle for attacker $E$ as it can initialize new sessions (w.r.t. the same or different identities).

Second, in the above attack we assume that $e$ is a small integer and attacker $E$ may know the value of one legal user’s identity $ID_i$. This is reasonable as explained below. On the one hand, in the system initialization phase (Section IV-A) the Chang-Lee scheme only specifies that the trusted party $SCPC$ needs to set its RSA key pair $(e, d)$ but does not give any limitation on the length of public exponent $e$. So, $e$ could be a small integer with binary length less than the output length of hash function $h$, i.e., $|e| < |h(\cdot)|$. Moreover, in practice this is likely to happen because: (a) to speed up the RSA signature verification, some security standards (e.g. PKCS #1 [PKC]), academic papers (e.g. [Bon99]) and popular web sites ((e.g. wikipedia [Wik12b]) suggest that $e$ can be set as 3 or 65537; and (b) as the Chang-Lee scheme is claimed to be efficient even for mobile devices in distributed networks, using small exponent $e$ can provide further computational advantage for these devices as they usually have limited resources for computation and storage [XSK+05]. In addition, the security analysis given in [CL12] neither excludes the case of small $e$ nor relies on the concrete procedure of setting $SCPC$’s RSA key pair $(e, d)$.

On the other hand, in the Chang-Lee SSO scheme users’ identities are not as crucial as their credentials, though the identities are transferred in ciphertext to provide user anonymity. So, users’ identities could be known by an attacker due to reasons, such as users’ negligence. At least service providers know users’ identities. Moreover, even if users’ identities are well protected so that attacker $E$ cannot impersonate registered user $U_i$ as above, $E$ can freely forge an identity $ID$. This is possible because in the Chang-Lee scheme, each user selects his/her identity by following only one requirement: each identity is a string with fixed bit-length. Therefore, even an outside attacker $E$ can use an arbitrary such string as an identity to mount the above attack, since the service providers are not provided any additional mechanism to check whether identity $ID$ has been registered with $SCPC$. This also
3.3. Attacks Against the Chang-Lee Scheme

implies that if $e$ is a small integer, $E$ can even impersonate a nonexistent user to make use of the resources and services offered by service providers.

Finally, it must be emphasized that impersonation attacks without valid credentials seriously violate the security of SSO schemes as it allows attacker to be successfully authenticated without first obtaining a valid credential from the trusted authority after registration. In other words, it means that in an SSO scheme suffering these attacks there are alternatives which enable passing through authentication without credentials.

3.3.3 Discussion

In [CL12], Chang and Lee provided a well-organized security analysis to show that their SSO scheme is secure. However, the two impersonation attacks presented in the previous section mean that their SSO scheme is actually not secure. So, why is their analysis not enough to guarantee the security of their scheme? What is the security flaw in their scheme leading to the above attacks? And what could we learn from these attacks to prevent similar situations in the future design of SSO schemes? These are the topics of this section.

In [CL12], the security of the Chang-Lee SSO scheme has been analysed in three different ways: 1. BAN logic [BAN90] was used to show the correctness of the Chang-Lee scheme; 2. Informal security arguments were given to demonstrate that their scheme can resist some attacks, including impersonation attacks. 3. A formal security proof was given to prove that their scheme is a secure authenticated key exchange (AKE) protocol [BR93a]. However, these security analyses and proofs still do not guarantee the full security of the Chang-Lee scheme and there are a number of reasons for this. First, as early as the 1990s it was known that although BAN logic had been shown useful to identify some attacks, it could approve protocols which are actually unsound in practice because of some technical weaknesses in the logic [BM94]. Moreover, in [CL12] the authors did not give details to show how the BAN logic can be used to prove that their scheme guarantees mutual authentication. In fact, at the end of section V-A of [CL12], the authors claimed to be able to: “prove that $U_i$ and $P_j$ are able to authenticate each other using our protocol.” but they provided no argument to show why each party could not be impersonated by an attacker. Second, the authors did discuss informally why their scheme could
3.3. Attacks Against the Chang-Lee Scheme

withstand impersonation attacks by considering two scenarios, for example, an attacker re-uses previous nonce $n_2$ to forge message $m_3$ or selects random credential $S_i$ to compute $SID_i$ by $SID_i = S_i^x \mod N$. However, such informal arguments neither strongly confirm their scheme’s security against these two concrete attacks nor exclude the existence of other scenarios of impersonation attacks, such as those presented in previous sections. Finally, their formal proof about AKE only focuses on the session key security, i.e., an attacker with all reasonable resources is not able to know the session key established between the two parties under the computational Diffie-Hellman (CDH) assumption (refer to Theorem 1 in [CL12], not the security of mutual authentication. According to the definitions given by Bellare and Rogaway [BR93a], one fundamental requirement of a secure AKE protocol is that there be a secure mutual authentication in the first place.

From the above, we can see that it is the use of credential proof $x = S_i^{h_2} \mod N$ which leads to the above two attacks against the Chang-Lee SSO scheme. More specifically, $x = S_i^{h_2} \mod N$ is a kind of knowledge proof which shows that a prover (usually played by user $U_i$) knows credential $S_i$. However, this is not a secure proof as a malicious verifier (i.e., service provider $P_j$) can recover $S_i$ and an outside attacker may be able to get authenticated without a credential. Based on this observation, a natural improvement on the Chang-Lee scheme would be to replace non-interactive proof $x$ by a rigorous but interactive zero knowledge (ZK) proof [FFS88] that shows the prover’s knowledge of secret $S_i = SID_i^d \mod N$ without revealing any additional information about credential $S_i$. In other words, using the verifiably encrypted signature introduced in [CM00], user $U_i$ can encrypt his/her credential $S_i$ under the public key of a trusted party and verifiably convince service provider $P_j$ that the ciphertext does contain $S_i$ w.r.t. $U_i$’s identity $ID_i$ without allowing $P_j$ to get any additional information about credential $S_i$. Compared with two modular exponentiations used for generating and verifying proof $x$, however, ZK proofs for showing the possession of an RSA signature usually require hundreds of modulo exponentiations [ASW00, CM00] since these proofs rely on inefficient ‘cut and choose’ method, i.e., binary challenges.

From the two attacks presented above, we can learn that both credential privacy and soundness are crucial for SSO schemes. As mentioned in Section III-A, credential privacy has been studied in Yang et. al [YWB04] and Han et al. [HMSY10]. To the best of our knowledge, however, there is surprisingly, no existing research which has given a careful treatment of soundness. For example, Han et al. [HMSY10] did
3.4 Attacks on the Hsu-Chuang Scheme

not investigate soundness, though they did carefully study how to formally define
credential forgery and recovery attacks from outsiders, users, service providers and
their potential collusion. According to the most traditional form of authentication,
a user will be authenticated if he/she can provide a valid pair of user name and
password (i.e. credential), and soundness is obviously satisfied because a user is
not able to go through authentication without providing a valid credential which
is registered and maintained by a server. In complex scenarios, like the Chang-Lee
scheme, the situation may be less obvious and, in fact, quite challenging. For this
reason, the problem remains an open one for future study. The question of formally
defining the soundness of SSO/authentication schemes and rigorously proving them
for concrete solutions remains an interesting and important one.

Finally, it must be noted that the analysis above shows only that the Chang-Lee
SSO scheme fails to achieve secure authentication, without violating its security for
achieving user anonymity and session key privacy.

3.4 Attacks on the Hsu-Chuang Scheme

In this section, we briefly highlight the difference between the Chang-Lee scheme
[CL12] and the Hsu-Chuang scheme [HC09] to see why the above describe imperson-
ation attacks apply to this latter as well. The two schemes have similar structures
and use similar notations, but the technical details differ. In summary, the Hsu-
Chuang scheme is differs from the Chang-Lee scheme in three ways. First, in the
Hsu-Chuang scheme user $U_i$’s credential $S_i$ is a naive RSA signature signed by the
trusted party $SCPC$, i.e., $S_i = ID_i^d \mod N$, where $ID_i$ is $U_i$’s identity selected
by him/herself. Second, to authenticate itself, service provider $P_j$ sends signature
$u = g_j^{h(Z||T_1||ID_j)}d_j \mod N_j$, where $Z$ is the DH key material generated by $P_j$, $T_1$
is the current timestamp, and $ID_j$ is $P_j$’s identity. Finally, for user authentication
user $U_i$ issues and sends proof $x = S_i^{h(K_{ij}||Z||w||T_2)} \mod N$ to $P_j$, who validates $x$ by
checking if $ID_i^{h(K_{ij}||Z||w||T_2)} = x^e \mod N$. For more detail, see [HC09] or Section II
of [CL12].

As pointed out in [CL12], the Hsu-Chuang scheme is vulnerable to impersonation
attack as an attacker can forge a valid credential $S_i$ w.r.t. identity $ID_i$ by simply
selecting random $S_i \in Z^*_N$ and then computing $ID_i = S_i^e \mod N$. This attack can
be excluded if a specific encoding format is required for identities and the credential
is issued by using a secure hash $h$, i.e., $S_i = h(ID_i)^d \mod N$, as in the Chang-Lee scheme. According to the discussion in Section III, the Hsu-Chuang scheme is still not secure even with such a countermeasure. The reason is that our two attacks against the Chang-Lee scheme apply to the Hsu-Chuang scheme as well. This means that the Hsu-Chuang scheme also fails to satisfy both credential privacy and soundness of authentication. In addition, there is another flaw in the Hsu-Chuang scheme. Attacker $E$ can impersonate service provider $P_j$ to cheat legal users, as the service authentication is conducted by using a non-traditional RSA signature, 

$$u = g_j^{h(Z||T_1||ID_j)} \mod N_j.$$ 

By communicating with $P_j$ twice attacker $E$ can get messages $(Z, T_1, ID_j, u)$ and $(Z', T_1', ID_j, u')$ so that $u = g_j^{h(Z||T_1||ID_j)} \mod N_j$ and $u' = g_j^{h(Z'||T_1'||ID_j')} \mod N_j$. Once $\gcd(h(Z||T_1||ID_j), h(Z'||T_1'||ID_j')) = 1$ (this holds with probability about 0.6), $E$ can find two integers $a$ and $b$ such that $a \cdot h(Z||T_1||ID_J) + b \cdot h(Z'||T_1'||ID_J) = 1$. Hence, $E$ can recover $g_j^{d_j}$ mod $N_j$ by computing $g_j^{d_j} \mod N_j = u^a u'^b \mod N_j$. After that, $E$ can impersonate $P_j$ to any legal user by using the value of $g_j^{d_j} \mod N_j$ to issue signature $u = (g_j^{d_j} \mod N_j)^{h(Z||T_1||ID_j)}$, without knowing $P_j$’s RSA private key $d_j$.

### 3.5 Proposed Improvement

To overcome the flaws in the Chang-Lee scheme [CL12], an RSA-based verifiable encryption of signatures (RSA-VES) can be employed. This is an efficient primitive introduced in [Ate04] for realizing fair exchange of RSA signatures.

The basic idea of the improved scheme can be highlighted as follows. User $U_i$’s credential is $S_i = h(ID_i)^{2d} \mod N$, i.e., SCPC’s RSA signature on the square of the hashed user identity (in contrast to $S_i = h(ID_i)^d \mod N$ in [CL12]). For user authentication, $U_i$ will encrypt his/her credential $S_i$ using ElGamal encryption of SCPC’s other public key $y = g^a$ by computing $P_1 = S_i \cdot y^r \mod N$ and $P_2 = g^r \mod N$, where $g \in \mathbb{Z}_N^*$ of big order and $u$ is SCPC’s secret decryption key. In this improvement, SCPC also plays the role of the trust authority in VES. To convince a service provider that $(P_1, P_2)$ does encrypt his/her credential $S_i$ (i.e. SCPC’s RSA signature for $ID_i$), $U_i$ must also provide an NZK proof $x$ to show that he or she knows a secret $r$ such that $\frac{P_1}{h(ID_i)^r} = (g^r)^r \mod N$ and $P_2 = g^r \mod N$. Such a proof $x$, is called ‘proving the equality of two discrete logarithms in a group of unknown order’ [Ate04], will convince the service provider without leaking any useful information about $U_i$’s credential $S_i$. For server authentication, service providers can simply
issue signatures as did [CL12], though the proposed changes give service providers the freedom to employ any secure signature scheme. The other procedures are the same as in the Chang-Lee scheme.

3.5.1 Initialization Phase

SCPC selects two large safe primes \( p \) and \( q \) to set \( N = pq \). Namely, there are two primes \( p' \) and \( q' \) such that \( p = 2p' + 1 \) and \( q = 2q' + 1 \). SCPC now sets its RSA public/private key pair \((e, d)\) such that \( ed \equiv 1 \mod 2p'q' \), where \( e \) is a prime. Let \( Q_N \) be the subgroup of squares in \( \mathbb{Z}_N^* \) whose order \( #G = p'q' \) is unknown to the public but its bit-length \( l_G = |N| - 2 \) is publicly known. SCPC randomly picks generator \( g \) of \( Q_N \), selects an ElGamal decryption key \( u \), and computes the corresponding public key \( y = g^u \mod N \). In addition, for completing the Diffie-Hellman key exchange SCPC chooses generator \( g \in \mathbb{Z}_N^* \), where \( n \) is another large prime number. SCPC also chooses a cryptographic hash function \( h(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^k \), where security parameter \( k \) satisfies \( 160 \leq k \leq |N| - 1 \). Another security parameter \( \epsilon > 1 \) is chosen to control the tightness of the ZK proof [GAT00]. Finally, SCPC publishes \((e, N, h(\cdot), \epsilon, g, y, g, n)\), and keeps \((d, u)\) secret.

3.5.2 Registration Phase

In this phase, upon receiving a register request, \( SCPC \) gives \( U_i \) fixed-length unique identity \( ID_i \), and issues credential \( S_i = h(ID_i)^2d \mod n \). \( S_i \) calculated as SCPC’s RSA signature on \( h(ID_i)^2 \) is an element of \( Q_N \), which will be the main group we are calculating.

As in [CL12], each service provider \( P_j \) with identity \( ID_j \) should maintain a pair of signing/verifying keys for a secure signature scheme (not necessarily RSA). \( \sigma_j(Msg, SK_j) \) denotes the signature \( \sigma_j \) on message \( Msg \) signed by \( P_j \) using signing key \( SK_j \). \( SVer(Msg, \sigma_j, PK_j) \) denotes verifying of signature \( \sigma_j \) with public key \( PK_j \), which outputs ‘1’ or ‘0’ to indicating if the signature is valid or invalid, respectively.

3.5.3 Authentication Phase

In this phase, RSA-VES is employed to authenticate a user, while a normal signature is used for service provider authentication. The details are illustrated in
3.5. Proposed Improvement

Fig. 3.2 and further explained as follows:

1. $U_i$ sends a service request with nonce $n_1$ to service provider $P_j$.

2. Upon receiving $(\text{Req}, n_1)$, $P_j$ calculates its session key material $Z = g^k \mod n$ where $k \in \mathbb{Z}_N^*$ is a random number, sets $u = Z || ID_j || n_1$, issues a signature $v = \sigma_j(u, SK_j)$, and then sends $m_2 = (Z, v, n_2)$ to the user, where $n_2$ is a nonce selected by $P_j$.

3. Upon receiving $m_2$, $U_i$ sets $u = Z || ID_j || n_1$. $U_i$ terminates the conversation if $\text{SVer}(u, v, PK_j) = 0$. Otherwise, $U_i$ accepts service provider $P_j$ because the
3.5. Proposed Improvement

signature is valid. In this case, $U_i$ selects a random number $t \in \mathbb{Z}_n^*$ to compute $w = g^t \mod n$, $k_{ij} = Z^t \mod n$, and the session key $K_{ij} = h(ID_j||k_{ij})$. For user authentication, $U_i$ first encrypts his/her credential $S_i$ as $P_1 = S_i \cdot y^r \mod N$, $P_2 = g^r \mod N$, where $r$ is a random integer with binary length $l_G$. Next, $U_i$ computes two commitments $a = (y^e)^{r_1} \mod N$ and $b = g^{r_1} \mod N$, where $r_1 \in \{0, 1\}^{l_G+k}$ is also a random number. After that, $U_i$ computes the evidence showing that credential $S_i$ has been encrypted in $(P_1, P_2)$ under public key $y$. For this purpose, $U_i$ calculates $c = h(K_{ij}|w||n_2||y^e||P_2||y^e||g|[a]|b)$ and $s = r_1 - c \cdot r$ (in $\mathbb{Z}$). Then, $x = (P_1, P_2, a, b, c, s)$ is the NIZK proof for user authentication. In fact, it is precisely, the processes of generating $x$ which is the proof part of RSA-VES [Ate04]. Finally, $U_i$ encrypts his/her identity $ID_i$, new nonce $n_3$, and $P_j$’s nonce $n_2$ using session key $K_{ij}$ to get ciphertext $CT = E_{K_{ij}}(ID_i||n_3||n_2)$, and thereafter sends $m_3 = (w, x, CT)$ to service provider $P_j$.

4. To verify $U_i$, $P_j$ calculates $k_{ij} = w^k \mod n$, the session key $K_{ij} = h(ID_j||k_{ij})$, and then uses $K_{ij}$ to decrypt $CT$ and recover $(ID_i, n_3, n_2)$. Then, $P_j$ computes $y^e = P_1^e / h(ID_i)^2 \mod N$, $a = (y^e)^s \cdot (y^e)^c \mod N$, $b = g^s \cdot (P_2)^c \mod N$, and checks if $(c, s) \in \{0, 1\}^k \times \{0, 1\}^{l_G+k}$ and $c = h(K_{ij}|w||n_2||y^e||P_2||y^e||g|[a]|b)$. If the output is negative, $P_j$ aborts the conversation. Otherwise, $P_j$ accepts $U_i$ and believes that they have shared the same session key $K_{ij}$ by sending $U_i m_4 = (V)$ where $V = h(n_3)$.

5. After $U_i$ receives $V$, he checks if $V = h(n_3)$. If this is true, then $U_i$ believes that they share the same session key $K_{ij}$. Otherwise, $U_i$ terminates the conversation.

3.5.4 Security Discussion

We now analyse the security of the improved SSO scheme by focusing on the security of the user authentication part, especially soundness and credential privacy due to two reasons. On the one hand, the unforgeability of the credential is guaranteed by the unforgeability of RSA signatures, and the security of service provider authentication is ensured by the unforgeability of the secure signature scheme chosen by each service provider. On the other hand, other security properties (e.g., user anonymity and session key privacy) are preserved, since these properties have been
formally proved in [CL12] and the corresponding parts of the Chang-Lee scheme are kept unchanged.

Soundness requires that without holding valid credential $S^*$ corresponding to a target user $U^*$, an attacker, who could be a collusion of users and service providers, has at most a negligible probability of generating proof $x^*$ and going through user authentication by impersonating user $U^*$. The soundness of the above improved SSO scheme relies on the soundness of the NIZK proof, which also guarantees the soundness of RSA-VES, defined as the second property of Definition 1 in [Ate04]. Namely, if the user authentication part is not sound, i.e., an attacker can present valid proof $x^*$ without holding the corresponding credential $S^*$ in non-negligible probability, then this implies the NIZK proof of proving equality of two discrete logarithms in a group of unknown order is not sound, contradictory to the analysis given in Section 3.7 of [Ate04].

Credential privacy or credential irrecoverableness requires that there be a negligible probability of an attacker recovering a valid credential from the interactions with a user. Again this property can be deduced from the signature hiding property of RSA-VES, defined as the third property of Definition 1 in [Ate04]. Signature hiding means that an attacker cannot extract a signature from VES without help from the user who encrypted the signature or the trusted authority who can decrypt a VES. So, if this improved SSO scheme fails to meet credential privacy, it implies that Ateniese’s RSA-VES fails to satisfy signature hiding, which is contrary to the analysis given in Section 3.7 of [Ate04]. In fact, soundness and signature hiding are the two core security properties to guarantee the fairness of digital signature exchange using VES.

More rigorous security proofs require to first formally define these two properties, and these are interesting topics for further study.

### 3.6 Conclusion

In this chapter, we demonstrated two effective impersonation attacks on Chang and Lee’s single sign-on (SSO) scheme [CL12]. The first attack shows that their scheme cannot protect the privacy of a user’s credential, and thus, a malicious service provider can impersonate a legal user in order to enjoy the resources and services from other service providers. The second attack violates the soundness of authentication by giving an outside attacker without credential the chance to
impersonate even a non-existent user and then freely access resources and services provided by service providers. We also discussed why their well-organised security arguments are not strong enough to guarantee the security of their SSO scheme. In addition, we explained why Hsu and Chuang’s scheme [HC09] is also vulnerable to these attacks. In addition, by employing an efficient verifiable encryption of RSA signatures introduced by Ateniese [Ate04], we proposed an improved the Chang-Lee scheme to achieve soundness and credential privacy. The unresolved problems for future work are to formally define authentication soundness and construct efficient and provably secure single sign-on schemes.
Chapter 4

A Provably Secure Single Sign-On Scheme

4.1 Introduction

As suggested in the previous chapter, it is necessary to formally define the soundness of authentication and to construct efficient and provably secure single sign-on schemes for mobile device users in distributed systems and networks. To design a secure SSO scheme, intuitively, there are three basic security requirements which should be considered: completeness, soundness and credential privacy. Completeness of authentication \cite{BR93a} requires that: (a) both sides accept each other if they have matched the conversation; (b) the probability that one side accepts the other one who actually has not engaged in the matching conversation is negligible. Soundness means that an unregistered user without a credential should not be able to access the services offered by service providers. Credential privacy guarantees that colluding dishonest service providers should not be able to fully recover a user’s credential and then impersonate the user to log in to other service providers. Soundness of credential based authentication, however, has not been formally studied yet although it is important, and the preserving of both soundness and credential privacy is still a challenge in designing SSO \cite{WYX12}.

In 2010, Han et al. \cite{HMSY10} proposed a generic construction of SSO schemes. This construction relies on broadcast encryption plus zero knowledge (ZK) proof \cite{FFS88} showing that the prover knows the corresponding private key of a given public key. So, implicitly, each user is assumed to have been issued a public key in a public key infrastructure (PKI). In the setting of an RSA cryptosystem, however, such ZK proof is very inefficient due to the complexity of interactive communications between the prover (a user) and the verifier (a service provider). In addition,
the requirement of PKI services for each user imposes a heavy burden on the implementation of SSO scheme. The function of session key establishment is also desired in order to secure the further communication.

In order to solve these problems, this chapter first formally defines the single sign-on schemes with authenticated key exchange. Then, we propose an SSO scheme according to the formal model by exploiting the Schnorr signature due to its simplicity and unforgeability [GMR89, Mao04]. In particular, this scheme uses Schnorr signature to generate a user’s credential and then, to authenticate him/her the user uses his/her credential as a private key to issue a Schnorr signature on some information generated in each session. As did in Chang-Lee scheme, a variant of Diffie-Hellman key exchange mechanism is employed to establish the session key. Furthermore, the security of the proposed protocol is discussed.

The rest of this chapter is organized as follows. The next section specifies a formal model for SSO with a unified definition of soundness and credential privacy. The proposed SSO scheme is given in section 4.3. The security of the proposed protocol is discussed in section 4.4. Finally, section 4.5 concludes this chapter.

4.2 Formal Model

In this section we present a formal model to define single sign-on schemes which support session key establishment. This model called authenticated key exchange single sign-on (AKESSO). This section also provides the security requirements of AKESSO. In particular, we list the components (e.g. syntax) of AKESSO, define correctness, describe an adversary model, and formally specify three security properties, including secure credential-based user authentication, secure credential-based service provider authentication, and session key security.

**Definition 4.1.** An authenticated key exchange single sign-on (AKESSO) scheme is comprised of trusted credential provider (TCP), group of service providers \( P \) and group of users \( U \). It consists of eight algorithms and protocol: initialization algorithm \( \text{Init}() \), identity generation algorithm \( \text{IdGen}() \), credential generation algorithm \( \text{CGen}() \), credential verification algorithm \( \text{CVer}() \), user proof generation algorithm \( \text{UPGen}() \), user proof verification algorithm \( \text{UPVer}() \), service provider proof generation algorithm \( \text{SPPGen}() \), and service provider proof verification algorithm \( \text{SPPVer}() \), and key exchange protocol \( \Pi \).
4.2. Formal Model

1. \textit{Init}(\lambda): Taking security parameter \(\lambda_0\) (or \(\lambda_1\)) as input, outputs the public/private key pair \((PK, SK)\) for TCP (or \((PK_j, SK_j)\) for \(P_j \in P\)).

2. \textit{IdGen}(RI_i): Taking registration information \(RI_i\) as input, outputs unique identity \(ID_i\) for user \(U_i \in U\).

3. \textit{CGen}(ID_i, SK): Taking identity \(ID_i\) and TCP’s private key \(SK\) as input, outputs credential \(C_i\) for user \(U_i\).

4. \textit{CVer}(C_i, ID_i, PK): Taking credential \(C_i\), identity \(ID_i\), and TCP’s public key \(PK\) as input, outputs ‘1’ or ‘0’ for accepting or rejecting credential \(C_i\) respectively.

5. \textit{UPGen}(C_i, ID_i, PK, M): Taking credential \(C_i\), identity \(ID_i\), TCP’s public key \(PK\) and temporal message \(M\) generated in a session as input, outputs user proof \(up_i\) showing user \(U_i\)’s knowledge of credential \(C_i\).

6. \textit{UPVer}(up_i, ID_i, PK, M): Taking user proof \(up_i\), identity \(ID_i\), TCP’s public key \(PK\), and temporal message \(M\) generated in a session as input, outputs ‘1’ or ‘0’ for accepting or rejecting \(up_i\) as a valid credential proof w.r.t. identity \(ID_i\) respectively.

7. \textit{SPPGen}(SK_j, M'): Taking service provider \(P_j\)’s private key \(SK_j\) and temporal message \(M'\) generated in a session as input, outputs service provider proof \(spp_j\) showing \(P_j\)’s knowledge of \(SK_j\).

8. \textit{SPPVer}(spp_j, PK_j, M'): Taking service provider proof \(spp_j\), \(P_j\)’s public key \(PK_j\), and temporal message \(M'\) generated in a session as input, outputs ‘1’ or ‘0’ for accepting or rejecting \(spp_j\) as a valid service provider proof w.r.t. public key \(PK_j\) respectively.

9. \(\Pi\): This is a key exchange protocol run by user \(U_i\) with private input \(C_i\) and service provider \(P_j\) with private input \(SK_j\). After the completion of each protocol instance, \(U_i\) will output session key \(K_{ij}\) if he/she accepts \(P_j\). Similarly, after the completion of each protocol instance \(P_j\) will output session key \(K_{ji}\) if it accepts \(U_i\). (Ideally, \(K_{ij}\) and \(K_{ji}\) are expected to be the same value.)

\textbf{Remark 4.1.} The above definition focuses on public key based AKESSO with non-interactive proofs. It could be extended to support interactive proofs, where \(sp_i\) and
ssp\_j are generated by interactive protocols run by user \( U_i \) and service provider \( P_j \). However, defining symmetric key based AKESSO is an area which is beyond the scope of this paper.

**Remark 4.2.** Compared to Han et al.’s formal model given in \([HMSY10]\), we require key exchange in AKESSO, and each user does not need to hold a public/private key pair. However, in Han et al.’s definition TCP (called IdP in their paper) is less trusted as it will not be able to impersonate any user: Each user will run a zero knowledge protocol to show that he/she knows the private key corresponding to the public key embedded in his/her credential.

Before formally defining security properties, it is obvious that an AKESSO must be **correct**. Credential \( C_i \) generated by trusted credential provider TCP will be valid. User proof \( u_{ip_i} \) issued properly by user \( u_i \) who holds a valid credential \( C_i \), will be accepted by service provider \( P_j \) according to the \( UPVer \) algorithm, service provider proof \( spp_j \) issued properly by \( P_j \) will be accepted by user \( U_i \) according to the \( SPPVer \) algorithm, and \( U_i \) and \( P_j \) will accept each other and output the same session key if they honestly run the key exchange protocol \( \Pi \). Formally, we define correctness as the following:

**Definition 4.2. (Correctness)** An AKESSO scheme is called correct if it satisfies all the following conditions:

1. For any \( RI_i \) and any key pair \( (PK, SK) \), if \( ID_i \leftarrow IdGen(RI_i) \) and \( C_i \leftarrow CGen(ID_i, SK) \), then \( CVer(C_i, ID_i, PK) = 1 \).

2. For any \( ID_i \), any key pair \( (PK, SK) \) and any \( M \), if \( C_i \leftarrow CGen(ID_i, SK) \) and \( u_{ip_i} \leftarrow UPGen(C_i, ID_i, PK, M) \), then \( UPVer(u_{ip_i}, ID_i, PK, M) = 1 \).

3. For any key pair \( (PK_j, SK_j) \) and any \( M' \), if \( spp_j \leftarrow SPPGen(SK_j, M') \), then \( SPPVer(spp_j, PK_j, M') = 1 \).

4. For any user \( U_i \) with valid credential \( C_i \) and service provider \( P_j \) with private key \( SK_j \), if both of them run the key exchange protocol \( \Pi \) honestly, then they will accept each other and output the same session key, i.e., \( K_{ij} = K_{ji} \).

Informally, an AKESSO scheme is secure if all the desired functionalities given in the above definition can be carried out only by the proper entities, i.e., not by attackers who are allowed to access all possible resources in a rigorously specified
adversary model. In fact, we shall define the security of SSO authentication which corresponds to items 1) to 3), and session key privacy which corresponds to item 4).

To further define these security properties, we specify the adversary model as follows: Let $\Pi_{TCP}$ be the trusted authority oracle with its key pair $(SK, PK)$, $\Pi_{U,P}^i$ be the user oracle simulating a set of all registered users, interacting with the service provider oracle in session $i$, and $\Pi_{P,U}^j$ be the service provider oracle simulating a set of all registered service providers, interacting with the user oracle in session $j$. Probabilistic polynomial time (PPT) adversary $A$ can ask the following oracle queries.

1. $O_1$: $\text{Register}(\Pi, U)$ — Upon receiving this query, $\Pi_{TCP}$ runs the $\text{IdGen}(RI_{A_i})$ and $\text{CGen}(ID_{A_i}, SK)$ algorithms, and outputs new user identity $ID_{A_i}$ with corresponding credential $C_{A_i}$ to $A$ who can verify the credential by running $\text{CVer}(\cdot)$.

2. $O_2$: $\text{Register}(\Pi, P)$ — Upon receiving this query, the system will run $\text{Init}(\lambda_1)$ and output $P_{A_j}$’s private/public key pair $(SK_{A_j}, PK_{A_j})$ together with identity $SID_{A_j}$ to $A$.

3. $O_3$: $\text{Execute}(U_i, P_j)$ — Upon receiving this query, $\Pi_{U,P}^i$ and $\Pi_{P,U}^j$ will execute protocol as $U_i$ and $P_j$ in $\Pi$, respectively. The exchanged messages between them will be recorded and sent to $A$. Here, we require that both $U_i$’s credential and $P_j$’s private key are not been corrupted by $A$ via $O_1$ and $O_2$ oracles.

4. $O_4$: $\text{Send}(U_i, m, f)$—This query sends the message $m$ as message flow $f \in \{0, 1, \cdots, n\}$ to the user oracle $\Pi_{U,P}^i$ which simulates a user $U_i$, and then, the oracle computes message honestly in $\Pi$, and sends responses back to $A$, where $n$ is the total number of messages transmitted in protocol $\Pi$. If a user is the protocol initiator by default, $A$ can also start a new session by asking $\text{Send}(U_i, \emptyset, 0)$, where $\emptyset$ denotes an empty set.

5. $O_5$: $\text{Send}(P_j, m, f)$—This query sends the message $m$ as message flow $f \in \{0, 1, \cdots, n\}$ to the user oracle $\Pi_{P,U}^j$ which simulates a service provider $P_j$, and then, the oracle computes message honestly in $\Pi$, and sends responses back to $A$. If a service provider is the protocol initiator by default, $A$ can also start a new session by asking $\text{Send}(P_j, \emptyset, 0)$.
4.2. Formal Model

6. $O_6$: \textit{Reveal}($\Pi, i$)—This query models the leakage of session key in session $i$. This query only can be asked when a session key has been shared between a service provider and a user in session $i$.

**Remark 4.3.** $O_3$ simulates the real environment for passive attacker $A$ who can eavesdrop all messages exchanged between $U_i$ and $P_j$ when executing protocol $\Pi$. If $A$ knows $U_i$’s credential $C_i$ and $P_j$’s private key $SK_j$, oracle $O_3$ is not necessary as $A$ can run protocol $\Pi$ by itself on their behalf. If $A$ knows one of these two secrets but not both, $A$ can run protocol $\Pi$ with $U_i$ ($P_j$) whose secret is not released by executing oracle $O_4$ ($O_5$).

**Remark 4.4.** $O_4$ simulates the real environment for active attacker $A$ who may obtain service provider $P_j$’s private key $SK_j$, send message $m$ as message flow $f \in \{0, 1, \cdots, n\}$ to target user $U_i$ and then get the corresponding response. To answer this oracle, $U_i$ will generate his/her response according to the specification of protocol $\Pi$ and send it to $A$. Note that if $U_i$ has not received all necessary previous messages that match this message with message flow $f$, this oracle request will be rejected, since it is meaningless to $U_i$. In fact, $O_4$ also provides adversary $A$ oracle access on algorithm $UPGen(\cdot)$ since $\Pi_{U,P}$ will run $UPGen(\cdot)$ somehow in executing $\Pi$. In our construction, $UPGen(\cdot)$ is the Schnorr signature generation algorithm. In this case, on the one hand, oracle $O_4$ may be no stronger than the signing oracle in Game-UFCMA reviewed in section IV, since temporal message $M$, one input of algorithm $UPGen(\cdot)$, may be jointly decided by $U_i$ and $A$ (playing the role of $P_j$), rather than just by $A$. So, it may be hard for $A$ to get $U_i$’s user proof for any arbitrary message $M$. On the other hand, adversary $A$ may, in fact, not be weaker than the forger in Game-UFCMA since besides $O_4$ we also offer other oracle queries, which may increase $A$’s attack capability.

To formally define soundness and credential privacy, it is necessary to discuss the difference between them, since the majority of existing schemes only consider credential privacy. Credential privacy requires unforgeability and unrecoverability. The former guarantees that any PPT adversary $A$ has only a negligible probability for successfully forging valid credential $C_t$ of target user $U_t$ in the credential generation phase, while the latter requires that in the user authentication phase, any $A$ can only recover $C_t$ with a negligible probability. Soundness is also critical in the user authentication phase as it ensures that there is a negligible probability that any
A without a valid credential can generate user proof $up$ that passes through user authentication. The existing studies [HMSY10, CL12] only focus on whether a valid credential can be forged or recovered by attackers, but do not consider if a valid credential is definitely necessary for generating a valid user proof. We shall define these three properties as a single definition (but one for users and one for service providers).

Let $A^O$ denotes adversary $A$ who has access to all oracle queries in $O = \{O_i| i = 1, 2, \cdots, 6\}$ in the adversary model; let credential holder $U_i$ with identity $ID_i$ and credential $C_i$, and service provider $P_j$ with identity $SID_j$ and key pair $(SK_j, PK_j)$ be two polynomial-time Turing machines. Let $U_i$ and $P_j$ interact with each other, and place $A$ between $U_i$ and $P_j$. $\epsilon$ denotes a negligible function. We define secure credential-based user authentication as follows:

**Definition 4.3. (Secure credential-based user authentication (SCUA))** An AKESSO scheme achieves secure credential-based user authentication, if PPT adversary $A$ has negligible advantage $Adv^{SCUA}(A^O)$ for creating a valid user proof without holding the corresponding credential. Formally, for any PPT $A$, $Adv^{SCUA}(A^O) \triangleq Pr[(ID_t, up_t, M) \leftarrow A^O[UPVer(up_t, ID_t, PK, M) = 1] \leq \epsilon$ with the following restrictions:

- $A$ has not obtained credential $C_t$ corresponding to $ID_t$ via $O_1$ - Register($\Pi$, $U$) oracle; and
- $A$ has not obtained valid user proof $up'_t$ for message $M$ by asking any oracle in $O$, in particular $O_3$ and $O_4$.

Similarly, the definition of secure service provider authentication is given below:

**Definition 4.4. (Secure service provider authentication (SSPA))** An AKESSO scheme achieves secure service provider authentication if PPT adversary $A$ has negligible advantage $Adv^{SSPA}(A^O)$ for forging a valid service provider proof without holding the corresponding service provider’s private key. Formally, for any PPT $A$, $Adv^{SSPA}(A^O) \triangleq Pr[(PK_t, M', spp_t) \leftarrow A^O[SPPVer(PK_t, M', spp_t) = 1] \leq \epsilon$ with the following restrictions:

- $A$ has not obtained the private key $SK_t$ corresponding to $SID_t$ via $O_2$ - Register($\Pi$, $P$) oracle;
4.3 Proposed Single Sign-On Scheme

- A has not obtained valid service provider proof spp for message $M'$ by asking any oracle in $O$, in particular $O_3$ and $O_5$.

Here, we review the freshness and test query $Test(\prod, i)$ for defining session key security \cite{BR93a}. An adversary can get session keys by asking $O_6$. We say the session key is fresh if and only if the $O_6$ query has not been asked w.r.t. this session. In other words, the fresh session key must be unknown to the adversary. For simplicity, we call the test query $O_7$, which is a game defined as follows:

- $O_7 — Test(\prod, i)$: In protocol $\prod$, if $\prod^{P, U}_i$ and $\prod^{i, P}_i$ accept and share the same fresh session key in session $i$, upon receiving this query, by tossing coin $b$ the correct session key is returned if $b = 0$, otherwise, a random session key is returned. $A$ can only ask this query once and $A$ needs to output one bit $b'$ as the result of guessing $b$. $A$’s advantage in attacking the session key security (SKS) of protocol $\prod$ is defined as $Adv_{sk}(\prod) = |2 \Pr[b' = b] - 1|$, where $O' = O \cup \{O_7\}$.

Session key security \cite{BR93a} models adversary $A$’s inability to distinguish the real session key and a random string, as formally defined below.

**Definition 4.5. (Session Key Security)** We say an AKESSO satisfies session key security if for any PPT adversary $A$, $Adv_{sk}(\prod) \leq \epsilon$, where $O' = O \cup \{O_7\}$.

Finally, we can give the definition of a secure authenticated key exchange single sign-on scheme.

**Definition 4.6. (Secure Authenticated Key Exchange Single Sign-On Scheme)**
An AKESSO scheme is called secure if it is correct and satisfies SCUA, SSPA, and session key security.

### 4.3 Proposed Single Sign-On Scheme

This section presents a secure single sign-on scheme with user anonymity for remote user authentication in distributed systems and networks. We use a Schnorr signature \cite{Sch89, Sch91} to overcome the drawbacks in the Chang-Lee scheme as their user proof cannot provide soundness and credential privacy while the Schnorr signature can. As a provably unforgeable signature scheme \cite{PS00}, Schnorr signature allows a signer to authenticate him/herself by signing a message without releasing
4.3. Proposed Single Sign-On Scheme

<table>
<thead>
<tr>
<th>TCP</th>
<th>The trusted credential provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_j$</td>
<td>A service provider</td>
</tr>
<tr>
<td>$U_i$</td>
<td>A user</td>
</tr>
<tr>
<td>$SID_j$</td>
<td>The unique identity of $P_j$</td>
</tr>
<tr>
<td>$ID_i$</td>
<td>The unique identity of $U_i$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>The credential of $U_i$</td>
</tr>
<tr>
<td>$x$</td>
<td>The long term private key of TCP</td>
</tr>
<tr>
<td>$y$</td>
<td>The public key of TCP</td>
</tr>
<tr>
<td>$E_k(M)$</td>
<td>Symmetric encryption of message M using key $k$</td>
</tr>
<tr>
<td>$D_k(C)$</td>
<td>Symmetric decryption of ciphertext $C$ using key $k$</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>A secure hash function</td>
</tr>
</tbody>
</table>

Table 4.1: Notations in the Proposed SSO Scheme

any other useful information about his/her private signing key. In the proposed scheme, the TCP first issues the credential for each user by signing the user’s identity $ID_i$ according to the Schnorr signature. Then, by treating his/her credential as another public/private key pair the user can authenticate him/herself by signing a Schnorr signature on a temporary message generated in the protocol. By contrast, each service provider maintains its own public/private key pair in any secure signature scheme so that it can authenticate itself to users by simply issuing a normal signature. Finally, as happens in the Chang-Lee scheme [CL12], the session key is established by running a variant of the Diffie-Hellman key exchange protocol, and user anonymity is guaranteed by symmetric key encryption. The notations used in the scheme are summarized in Table 4.1.

**System Setup Phase:** In this phase, TCP initializes his/her public and private parameters as a Schnorr signature scheme. First, TCP picks large primes $p$ and $q$ such that $q|p - 1$, chooses generator $g$ of large safe prime order $q$ in cyclic group $G$. Then, TCP sets its private key $SK = x$, where $x \in \mathbb{Z}_q^*$ is a random number, and publishes its public key $PK = y$, where $y = g^x \mod p$.

**Registration Phase:** In this phase, the user asks TCP for registration, then TCP issues unique identity $ID_i$ via $IdGen(RI_i)$ and signs a Schnorr signature $(a, e, C)$ for user’s identity as credential generation algorithm $CGen(ID_i, SK)$. $C$ is kept secret by the user, while $(a, e)$ will be made public. The details are given below.

- **User Registration:** When user $U_i$ asks for registration, TCP selects unique identity $ID_i$ and generates credential $C_i = (a, e, C)$ for $U_i$ by selecting a randomness $r \in \mathbb{Z}_q^*$ and computing $a = g^r \mod p$, $e = h(a, ID_i)$, and $C = r + xe$
mod $q$. Then, TCP sends identity $ID_i$ and credential $C_i$ which is a Schnorr signature for $ID_i$ to user $U_i$, where $C$ should be kept secret.

- Service Provider Registration: Each $P_j$ maintains a public/private key pair $(PK_j, SK_j)$ of any secure signature scheme. Here, algorithms $SPPGen(\cdot)$ and $SPPVer(\cdot)$ are identical to the signature generation and verification algorithms respectively.

**Authentication Phase:** In this phase, to authenticate him/herself user $U_i$ signs a Schnorr signature on the newly established session key $K_{ij}$ using credential $C$ the signing key, while $U_i$’s session key material $k_2$ is used as the commitment. Note that the corresponding verification key of $C$ is $g^C$, which can be recovered by computing $g^C = a \cdot g^e \mod p$. For service provider authentication, any provably secure signature scheme can be used to authenticate a service provider in the proposed scheme. The session key is established by using the modified Diffie-Hellman key exchange scheme which has been formally proved in [CL12], and user anonymity and unlinkability are preserved by using symmetric key encryption to encrypt $a$, $e$, and user’s identity $ID_i$. The details of this phase are illustrated in Figure 4.1 and further explained below.

1. User $U_i$ chooses random nonce $n_1$ and sends $M_1 = (\text{Req}, n_1)$ to $P_j$, where $\text{Req}$ is a service request.

2. Upon receiving $(\text{Req}, n_1)$, $P_j$ picks random number $r_1 \in \mathbb{Z}_q^*$, computes its session key material $k_1 = g^{r_1} \mod p$, $u = h(k_1||SID_j||n_1)$ and signs $u$ to get signature $v = SPPGen(SK_j, u)$, and sends $M_2 = (k_1, v, n_2)$ to the user.

3. User $U_i$ first computes $u = h(k_1||SID_j||n_1)$ and verifies the signature $v$ by checking if $SPPVer(PK_j, u, v) = 1$. If the output is “0”, $U_i$ terminates the protocol. Otherwise, $U_i$ accepts service provider $P_j$’s authentication, and then selects random number $r_2 \in \mathbb{Z}_q^*$ to compute $k_2 = g^{r_2} \mod p$, $k_{ij} = k_1^2 \mod p$, and the session key $K_{ij} = h(SID_j||k_{ij})$. After that, $U_i$ signs $K_{ij}$ using his/her credential secret $C$ by calculating $e_i = h(k_2, K_{ij})$, $z = r_2 + Ce_i \mod q$ and $\omega = E_K(ID_i||n_3||n_2||e||a)$, where $n_3$ is a nonce chosen by $U_i$. Finally, $U_i$ sends $M_3 = (\omega, z, k_2)$ to service provider $P_j$.

4. To verify $z$, $P_j$ first calculates $k_{ij} = k_2^1 \mod p$, derives session key $K_{ij} = h(SID_j||k_{ij})$ and decrypt $\omega$ with $K_{ij}$ to recover $ID_i||n_3||n_2||e||a$. Then, $P_j$
4.3. Proposed Single Sign-On Scheme

\[ u = h(k_1 \| \text{SID}_j \| n_1) \]
\[ SPPVer(PK_j, u, v) \xrightarrow{?} 1 \]
\[ k_2 = g^{e_i} \mod p \]
\[ K_{ij} = h(SID_j \| k_2) \]
\[ e_i = h(k_2, K_{ij}) \]
\[ z = r_2 + C \cdot e_i \]
\[ \omega = E_{k_2}(ID_j \| n_1 \| n_2 \| e \| a) \]
\[ M_4 = (\omega, z, k_2) \]
\[ V' = h(n_3) \]
\[ V' \xrightarrow{?} V \]

Figure 4.1: Participant Identification Phase of the Proposed SSO Scheme

checks if \( e = h(a \| ID_1) \). If this does not hold, \( P_j \) aborts the protocol. Otherwise, the service provider computes \( e_i = h(k_2, K_{ij}) \) and verifies \( z \) by checking if \( g^z = k_2 \cdot a^{e_i} \cdot (y^a)^{e_i} \mod p \). If this holds, \( P_j \) accepts \( U_i \)'s authentication, believes that they have shared the same session key \( K_{ij} \), and sends \( V = h(n_3) \) as \( M_4 \) to \( U_i \).

5. User \( U_i \) computes \( V' = h(n_3) \) and checks if \( V' = V \). If this holds, \( U_i \) believes that he/she has shared the same session key \( K_{ij} \) with \( P_j \).
4.4 Security Analysis

The proposed scheme employs the Schnorr signature scheme \cite{Sch89, Sch91} to generate credentials for users, uses a modified Diffie-Hellman key exchange scheme to establish the session key, signs a Schnorr signature on the hashed session key for user authentication, uses any secure signature scheme for server authentication, and uses symmetric key encryption to ensure user anonymity. The secure authenticated key exchange single sign-on (AKESSO) scheme requires secure credential-based user authentication (SCUA), secure service provider authentication (SSPA), and a secure session key. To prove the security of the proposed AKESSO, it will only be necessary to prove SCUA and SSPA because (1) the proposed scheme only improves parts of key generation, user authentication and service provider authentication in the Chang-Lee scheme \cite{CL12}, while user anonymity and session key establishment have not been modified; and user anonymity and session key security have been proved in \cite{CL12} and discussed in \cite{WYX12} without revealing any problems. The following is a formal analysis of the security of the proposed AKESSO scheme.

**Theorem 4.1.** *(Correctness)* The proposed construction is a correct AKESSO scheme according to Definition 2.

**Proof.** This can be verified according to Definition 2 given in Section II.

Informally, the proposed AKESSO scheme guarantees SSPA as each service provider employs a secure signature scheme. To prove SCUA, we need to show that Definition 4.3 holds for the proposed AKESSO scheme by assuming the unforgeability of the Schnorr signature scheme.

**Theorem 4.2.** *(Secure Credential-based User Authentication)* In the proposed AKESSO scheme, if there is PPT adversary $A$ who has non-negligible advantage $\text{Adv}^{\text{SCUA}}(A^O)$ as specified in Definition 3, then the Schnorr signature scheme is existentially forgeable under UFCMA attacks as defined in Section IV.

**Proof.** As adversary $A$, with access to all oracles in $\mathcal{O} = \{O_1, \ldots, O_6\}$, has non-negligible advantage $\text{Adv}^{\text{SCUA}}(A^O)$, according to Definition 3 this implies that at least one of the following two cases is true:

- **Case (1):** With non-negligible probability $\epsilon_1$, $A^O$ is able to derive credential $C_t$ corresponding to *unregistered* target identity $ID_t$. 
• **Case (2):** With non-negligible probability $\epsilon_2$, $A^O$ is able to forge a valid user proof for new message $M$ w.r.t. a *registered* target identity $ID_t$.

If either Case (1) or Case (2) is true, we can construct an algorithm $B$ that is able to break the unforgeability of the Schnorr signature, where $B$ runs $A^O$ as a sub-program for fulfilling its purpose.

**Case (1).** Suppose that $B$ is given a target Schnorr signature scheme with parameter $(p, q, h(\cdot))$ and public key $y = g^x \mod p$, where the private key $x$ is not known to $B$. $B$’s strategy for winning Game-UFCMA with non-negligible probability is to set up an AKESSO scheme for $A$ and to simulate oracles in $O$ so that $A$ cannot distinguish the difference between this simulated environment and a real AKESSO scheme. Therefore, $A$ will be able to successfully derive credential $C_t$ for unregistered identity $ID_t$ with probability $\epsilon_1$. After that, $B$ can adapt this credential into a forged Schnorr signature for a new message and thus break the unforgeability of the Schnorr signature scheme.

How does $B$ sets up such a simulated AKESSO scheme for $A$. First, $B$ sets $y$ as the public key of TCP and gives $y$ to $B$. Then, each oracle in $O_i$ ($i = 1, \ldots, 6$) can be simulated as follows. To simulate an $O_1$ query, $B$ can ask its own signing oracle to get Schnorr signature $C_i$ for each identity $ID_i$ and then reply $(ID_i, C_i)$ to $A$. To simulate an $O_2$ query, $B$ can simply run $Init(\lambda_1)$ to get public/private key pair $(SK_j, PK_j)$ for an identity $SID_j$, and then forward $(SID_j, SK_j, PK_j)$ to $A$. As $B$ knows all users’ credentials and all service providers’ private keys, it can simulate oracles $O_3$, $O_4$, $O_5$ and $O_6$ by executing the whole protocol $\prod$, running one move on behalf of a user, running one move on behalf of a service provider, and revealing a session, respectively. Note that as $ID_t$ is an unregistered identity in this case, the corresponding user $U_t$ will not be involved in any oracle $O_i$ ($i = 1, \ldots, 6$).

It is not difficult to see that the above simulated system is indistinguishable from a real system from $A$’s point of view. Hence, $A$ will be able to output credential $C_t$ for target identity $ID_t$ with non-negligible probability $\epsilon_1$, where $ID_t$ is not asked in $O_1$ queries. Therefore, $B$ will simply forward $C_t$ as a forged Schnorr signature for message $ID_t$. Since $ID_t$ is not asked in $O_1$ queries, $A$ does not ask $ID_t$ in its signing oracle, i.e., $ID_t$ is a new message for $B$. So, $B$’s forged message-signature pair $(ID_t, C_t)$ is valid according to the definition of Game-UFCMA (refer to Section IV). Moreover, $B$’s success rate is exactly the same as $A$’s, i.e., $\epsilon_1$, which is non-negligible. Consequently, this means that $B$ successfully breaks the unforgeability
of the Schnorr signature scheme.

**Case (2).** This can be proved in a similar way to Case (1) but $B$ will embed its target Schnorr signature scheme in the user proof generation algorithm for registered target user $U_t$ with identity $ID_t$. Details are given as follows.

Suppose that $B$ is given a target Schnorr signature scheme with parameter $(p,q,h(\cdot))$ and public key $y' = g^{x'} \mod p$, where private key $x'$ is not known to $B$. First, $B$ sets $y = g^x \mod p$ as the public key of TCP by selecting random number $x$ as TCP’s private key. For any identity $ID_i$ except target identity $ID_t$, to answer an $O_1$ query $B$ can directly issue credential $C_i$ for $ID_i$ by generating a Schnorr signature for $ID_i$ as $B$ knows TCP’s private key. In contrast, $B$ will take $(a',e',x')$ as the credential $C_t$ for target identity $ID_t$, where $e' \in \{0,1,\cdots,q-1\}$ is a random number, $a' \in \mathbb{Z}_p^*$ is set as $a' = y'^e \cdot y^{e'} \mod p$, and $h(a',ID_t)$ is set as $e'$. So, we have $g^{x'} = a'y^{h(e',ID_t)} \mod p$. Note that $B$ does not know the value of $x'$ and it will be not required to reveal $C_t$ to $A$ because $ID_t$ is the target identity. In addition, here we can artificially fix the hash value for such a special input $(a',ID_t)$ because the Schnorr signature is secure in random oracle where hash function can be viewed as random function $[PS00]$. All other oracles in $O$ can be simulated as in Case (1), except $A$ asks $O_3$ and $O_4$ queries in which $U_t$ with identity $ID_t$ is involved. In such scenarios, $B$ can simulate $U_t$ to output valid user proof $up_t$ w.r.t. credential $C_t$ by executing the whole protocol $\prod$ or running one move with necessary help from its own signing oracle w.r.t. public key $y'$.

Again, it is not difficult to see that the above simulated system is indistinguishable from a real system from $A$’s point of view. Hence, with probability $\epsilon_2$ $A$ will be able to output valid user proof $up_t$ for message $M$ w.r.t. target identity $ID_t$, where $M$ is not asked in $O_3$ and $O_4$ queries. Therefore, $B$ can simply forward $up_t$ as a forged Schnorr signature for message $M$. Since $M$ is not asked in $O_3$ and $O_4$ queries, $A$ does not ask $M$ in its signing oracle, i.e., $M$ is a new message for $B$. So, $B$’s forged message-signature pair $(up_t,M)$ is valid according to the definition of Game-UFCMA (refer to Section IV). Moreover, $B$’s success rate is exactly the same as $A$’s, i.e., $\epsilon_2$, which is non-negligible. Consequently, this means that $B$ successfully breaks the unforgeability of the Schnorr signature scheme.

**Remark 4.5.** In Case (1), $A^O$ could directly forge $C_t$, recover $C_t$ after executing protocol $\prod$ with user $U_t$ or eavesdropping on the messages between $U_t$ and some service providers, or derive $C_t$ in any other possible way, though $A^O$ is not allowed
to obtain $C_t$ by simply asking $O_3$ oracle w.r.t. $ID_t$. Hence, if our AKESSO fails to satisfy the unforgeability or unrecoverability of the credential, then the Schnorr signature is forgeable. Similarly, in Case (2) $A^O$ could directly forge user proof $up_t$ without credential $C_t$, observe and adapt existing user proofs generated by $U_t$ into user proof $up_t$ for message $M$, or compute $up_t$ in any other way, though $A^O$ is not allowed to obtain any user proof for the same message $M$ by simply asking $O_3$ and $O_4$ oracles w.r.t. $ID_t$. Hence, if our AKESSO fails to satisfy soundness of credential-based authentication [WYX12], then the Schnorr signature is forgeable.

As the Schnorr signature scheme is proved to be secure under the discrete logarithm assumption [PS00], Theorem 4.2 assures that the proposed AKESSO scheme achieves secure credential-based user authentication under the discrete logarithm assumption.

**Theorem 4.3. (Secure Service Provider Authentication) In the proposed AKESSO, if there is PPT adversary $A$ who has non-negligible advantage $\text{Adv}_{\text{SSPA}}(A^O)$ as specified in Definition 4, then the signature scheme employed by service providers is existentially forgeable under UFCMA attacks as defined in Section IV.

Proof. Since a service provider proof is directly generated as a normal signature by the corresponding service provider, Theorem 4.3 can be formally proved as we did for Case (2) in Theorem 1. Note that here we do not need to discuss Case (1) as in Theorem 1, because each service provider is required to register its public/private key pair.

**Theorem 4.4. According to Definition 6, the proposed AKESSO scheme is secure under the assumption that all digital signatures employed in the scheme are existentially unforgeable against UFCMA attacks as specified in Section IV.

Proof. By Theorem 1, Theorem 2, Theorem 3 and session key security proved in [CL12], Theorem 4 holds according to Definition 6.

### 4.5 Conclusion

Most existing single sign-on schemes have a number of security problems and are vulnerable to various types of attacks. In this chapter, we first formalized an authenticated key exchange single sign-on scheme. In particular, we formally defined secure authentication for both users and service providers because this had
not been done before [Wyx12]. Then, a Schnorr mechanism based SSO scheme was proposed to overcome the drawbacks of the Chang-Lee scheme [CL12] while preserving its advantages. In this new scheme, to preserve credential generation privacy, the TCP signs a Schnorr signature [Sch89, Sch91] on user’s identity; and to protect credential privacy and soundness, the user exploits his/her credential as a signing key to sign a Schnorr signature on the hashed session key. In fact, the Schnorr signature mechanism [Sch89, Sch91] is more efficient than the RSA mechanism which was employed by the Chang-Lee scheme. Thus, the proposed scheme reduces the computation cost, enhances confidentiality, while preserving soundness and credential privacy.
Chapter 5

A Generic Framework of Three-Factor Authentication

5.1 Introduction

Two factor authentication schemes were introduced in the previous two chapters. For the user who has high security requirements, however, two factor authentication schemes may be not secure enough. To resolve this problem, three factor authentication schemes have been introduced. Many existing three factor schemes, however, have security problems and privacy issues. In order to address this problem, Huang et al. [HXC+11] proposed a generic framework for three factor authentication. This framework upgrades two factor authentication scheme to three factor authentication scheme without any additional requirement. It also preserves the privacy of user’s biometric characteristics, while without the requirement of trusted devices. Huang et al.’s framework employs the ‘fuzzy extractor’ [DRS04] to generate a biometric key. The ‘fuzzy extractor’ uses Hamming distance, set difference and edit distance to tolerate errors. These distance measurements, however, have not been widely accepted by the majority of biometric applications [WQ10]. In addition, the process of Huang et al.’s framework can be reduced from running underlying scheme twice to running it once. Huang et al. also have not deeply analysed the practicalness and they have not provided a proper concrete scheme since that they put the above work in the future.

This chapter proposes an improved generic framework of three factor authentication which based on [HXC+11]. This improved framework is more efficient and practical while remains all advantages of [HXC+11]. According to the improved framework, a provably secure concrete instantiation is provided along with its implementation analysis and privacy discussion. In particular, we propose a security
model for three-factor-based authentication schemes which support session key establishment. A formal proof of our concrete instantiation is also provided according to this security model.

The rest of this chapter is organised as follows. Section 5.2 reviews and discusses the two well known biometric identification mechanisms. After that, Section 5.3 reviews Huang et al.’s framework and then provides an improved generic framework for three-factor authentication. The concrete instantiation with analysis and comparison are given in Section 5.4, in which, formal security proof and privacy discussion of this instantiation are also provided. Finally, Section 5.5 concludes this chapter.

5.2 Biometric Identification Mechanisms

In 1999, Juels and Wattenberg [JW99] proposed the first biometric identification scheme, fuzzy commitment, using Hamming distance to tolerate errors. Later, in 2002, Juels and Sudan [JS02] introduced a provably secure ‘fuzzy vault’ scheme, in which, a user chooses a long-bit secret key (treated as a biometric key) in advance, and hides it using the user’s biometric template. In the ‘fuzzy vault’, however, the Euclidean distance measurement is used to tolerate errors. In 2004, Dodis et al. [DRS04] proposed a provably secure ‘fuzzy extractor’ which generates a random pair strings $R$ as a biometric key and a corresponding auxiliary string $P$ directly from the user’s biometric template. The ‘fuzzy extractor’ uses Hamming distance, set difference and edit distance to tolerate errors. The ‘fuzzy vault’ has been widely accepted since the Euclidean distance measurement is suitable for the majority of biometric applications [WQ10], while the distance measurements used in the ‘fuzzy extractor’ are not. This is also the reason why we choose the ‘fuzzy vault’ for biometric key generation. In 2008, Teoh and Ong [AT08] proposed a randomised dynamic quantisation transformation (RDQT), which is based on fuzzy commitment, to binarize biometric data, satisfying randomness and uniqueness. Meanwhile, Sheng et al. [SHFD08] presented a template-free biometric-key generation, which also can generate a key directly from a biometric template. This section reviews the ‘fuzzy extractor’ which has been employed by Huang et al.’s framework [HXC+11], and the ‘fuzzy vault’ scheme which is employed in our proposed framework.
5.2. Biometric Identification Mechanisms

5.2.1 Fuzzy Extractor

The ‘fuzzy extractor’ has two procedures, a generation procedure (Gen) and a reproduction procedure (Rep). After a user scans his biometric features, the Gen extracts uniquely random $R$ and corresponding auxiliary $P$ from user’s biometric template $w$. In the authentication phase, the inputs of Rep are $P$ and unidentified biometric template $w'$; the output of Rep is the corresponding $R$ iff the difference between $w$ and $w'$ is within an acceptable error tolerance. The error tolerance in the scheme depends on three error correcting techniques, namely Hamming distance, set difference and edit distance. The definition of the ‘fuzzy extractor’ was introduced by Dodis et al. [DRS04, DORS08]. To formally review this concept, we introduce the following notations.

- $t$: the fuzzyness of the ‘fuzzy extractor’;
- $A, B$: two probability distributions;
- $M$: a metric space on $N$ points with distance function $dis(\cdot)$;
- $m$: the min-entropy of $A$ given $B$, which can be calculated by computing the logarithm of average probability of value $A$ given $B$;
- $U_l$: the uniform distribution on $l$-bit binary strings;
- $SD(A,B)$: the statistical distance between $A$ and $B$ such that $SD(A,B) = \frac{1}{2} \sum_v |\Pr(A = v) - \Pr(B = v)|$;

Definition 5.1. An $(M, m, l, t, \epsilon)$–fuzzy extractor is a pair of randomised procedures Gen and Rep, respectively, with the following properties:

1. Gen is a probabilistic generation procedure with input $w \in M$, which outputs public helper $P \in \{0,1\}^*$ and an ‘extracted’ random string $R \in \{0,1\}^l$. For any distribution $W$ on $M$ of min-entropy $m$, if $<R,P> \leftarrow$ Gen($W$), then it requires that $SD(<R,P>, <U_l,P>) \leq \epsilon$.

2. Reproduction procedure Rep, can recover $R$, if and only if $P$ and $w'$ are provided as inputs, where $w' \in M$, satisfies $dis(w,w') \leq t$. Namely, if $<R,P> \leftarrow$ Gen($W$), then $Rep(w',P) = R$.
The ‘fuzzy extractor’ provides a good insight into biometric identification since it extracts a unique random ‘private’ key directly from the user’s biometric features. However, as a theoretical biometric key generation scheme for public key cryptosystem, the ‘fuzzy extractor’ has not been widely implemented since the distance measures in it are less accepted than the Euclidean distance measurement in biometric applications [WQ10].

5.2.2 Fuzzy Vault

In 2002, Juels and Sudan [JS02] proposed a cryptographic construction for data protection and user authentication by using fingerprints and smart cards, called the ‘fuzzy vault’. The errors in the ‘fuzzy vault’ have been tolerated by the Euclidean distance measurement which has been widely accepted by the majority of biometric applications. The operations of the ‘fuzzy vault’ are described as follows.

First, a user’s biometric features are scanned and his/her biometric template $X$ is extracted. Then, s/he selects and uses a polynomial $Pol$ to encrypt secret string $K$ (treated as the biometric key) which has been chosen by the user in advance. The user evaluates $Pol$ on all elements in $X$ and chooses a large number of random chaff points which do not lie on $Pol$ as the noise. The final vault $V$ is the collection of the genuine minutiae points which lie on $Pol$ and the chaff points which do not lie on $Pol$.

To recover secret string $K$ from vault $V$, the user needs to offer his/her biometric template $X'$, if the difference between $X$ and $X'$ is $|X - X'| < \epsilon$, where $X - X' = \{x | x \in X, x \notin X'\}$, then polynomial $Pol$ can be reconstructed because a sufficient number of points on $Pol$ can be identified and an error correcting scheme is used. Thus, $K$ can be successfully recovered once $Pol$ is available.

In 2003, Clancy et. al [Cla03] proposed a secure smart card-based fingerprint authentication scheme by using Juels and Sudan’s ‘fuzzy vault’. Later, in 2007, Nandakumar et. al [NJP07] proposed a fully automatic implementation by employing the ‘fuzzy vault’, and using helper data to align unidentified fingerprints accurately. The improved scheme used both location $(x, y)$ and orientation attribute $\theta$ of a minutiae point to record the biometric data, where $(x, y)$ is the row and column indicates in the image as the location, and $\theta$ is the orientation on the X-axis. The helper data is high curvature points extracted from the fingerprint orientation field, thus it neither affects the security nor leaks any information about the biometric template.
One year later, Nagar, Nandakumar and Jain [NNJ08] improved the security and matching accuracy of Nandakumar et. al’s fingerprint-based ‘fuzzy vault’ scheme by employing additional minutiae descriptors [Fen08], which capture local ridge orientation and ridge frequency information in the neighbourhood of a minutia. The results in [NNJ08] showed that the improved scheme reduces the false acceptance rate (FAR) and significantly increases the vault security. The operation of fingerprint based ‘fuzzy vault’ follows:

Let a locking/unlocking pair \( (\text{Lock}, \text{Unlock}) \) is complete \( \epsilon \)-fuzziness if the following holds. For every secret string \( k \) and every pair of biometric template sets \( (X, X') \), such that \( |X - X'| \leq \epsilon \) for integer \( \epsilon \), then \( \text{Unlock}(X', \text{Lock}(X, k)) = k \) with overwhelming probability.

**Vault Encoding (Lock):**

1. \( \frac{X}{K, \text{Pol}} \xrightarrow{\text{Pol}} \left[ P_X(K) \right] \rightarrow L \)
   
   Procedure \( P_X(K) \) denotes that the ‘fuzzy vault’ encrypts user’s secret \( K \) in polynomial \( \text{Pol} \), and evaluates \( \text{Pol} \) on all elements in the user’s biometric sample \( X \), which is represented as an unordered set. The output of \( P_X(K) \) is Locking set \( L \);

2. \( \frac{CP}{L} \xrightarrow{\text{Gen}} V \)
   
   The user selects chaff points \( CP \) which play the role of noise as the inputs of \( \text{Gen} \), where chaff points \( CP \) do not lie on \( \text{Pol} \), while \( L \) does. \( r \) denotes the number of points which lie on \( \text{Pol} \) in \( V \), and \( s \) denotes the number of points which do not lie on \( \text{Pol} \) in \( V \), where \( s >> r \). The output of \( \text{Gen} \) is \( V \) such that \( V = CP \cup L \).

**Vault Decoding (Unlock):**

1. \( \frac{X'}{V, H} \xrightarrow{\text{Rec}} \left[ \text{Pol} \right] \rightarrow \text{Pol} \)
   
   For the user who requests to recover \( \text{Pol} \), the ‘fuzzy vault’ first uses original helper data \( H \) and the requester’s help data \( H' \) to adjust the orientation of the fingerprint, and then runs procedure \( \text{Rec} \) to reconstruct \( \text{Pol} \) from input \( V \) if the difference between \( X \) and the requester’s extracted biometric template \( X' \) satisfies \( |X - X'| < \epsilon \), where \( X - X' = \{ x | x \in X, x \notin X' \} \);

2. \( \frac{\text{Pol}}{} \xrightarrow{\text{De(Pol)}} K \)
   
   The procedure \( \text{De(Pol)} \) denotes the recovering algorithm which outputs the secret key \( K \) by giving the input polynomial \( \text{Pol} \).
5.3. A Generic Three-factor Authentication Framework

Here, $r$ is the number of genuine points which lie on $Pol$ in $V$, and this depends on the number of features which have been extracted from $X$. The security of the ‘fuzzy vault’ is in proportion to the number of chaff points. The degree of polynomial is presented as $n$. Parameter $\epsilon$ denotes error tolerance. Helper data $H$ consists of the high curvature points and the ordinate value of vault $V$, and $H$ does not leak the information of the user’s biometric features [NJP07, NNJ08]. The security of the ‘fuzzy vault’ is based on (a) the difficulty in distinguishing the set of genuine minutiae points from a set of chaff points in vault $V$ and (b) the difficulty to reconstruct the polynomial $Pol$ in vault $V$.

5.3 A Generic Three-factor Authentication Framework

This section first reviews Huang et al.’s scheme, and then provides a more efficient and practical framework.

5.3.1 Review of Huang et al.’s Framework

Huang et al.’s framework employs the ‘fuzzy extractor’ to generate a uniquely long-bit random string as the biometric key of the user. By running the underlying two-factor scheme twice, a three-factor scheme is constructed. Specifically, the first running uses password and smart card as normal. Then, in the second time, the user replaces the password by a biometric key and runs the underlying protocol again, thus achieving a three-factor authentication. Huang et al.’s framework consists of three phases:

Registration:
The processes of registration includes the following steps:

1. User $U_i$ chooses initial password $PW_1$;

2. Upon $U_i$ scanning his/her biometric features, biometric template $X$ is extracted, and then a pair $(R, P)$ is outputted by running $Gen(X)$;

3. Let second password $PW_2 = h(R)$, where $h(\cdot)$ is a cryptographic hash function chosen by $U_i$. 
4. $U_i [PW_1] \xleftarrow{2\text{-Factor-Reg}} S [SK_1] \rightarrow Data_1$;
   By running the underlying two-factor registration protocol (2-Factor-Reg),
   user $U_i$ uses initial password $PW_1$ and server $S$ uses secret key $SK_1$ to generate
   $Data_1$;

5. $U_i [PW_2] \xleftarrow{2\text{-Factor-Reg}} S [SK_2] \rightarrow Data_2$.
   $Data_2$ is generated by running the 2-Factor-Reg again, in which $U_i$ uses $PW_2$
   and $S$ uses $SK_2$;

6. Server stores $Data_1$ and $Data_2$ in $SC$ and gives it to $U_i$;

7. $U_i$ updates $SC$ by adding $Data_3 = (P, h(\cdot), Rep(\cdot))$ in it, where $P$ is the
   auxiliary string for biometric key recovery, $h(\cdot)$ and $Rep(\cdot)$ are the descriptions
   of the corresponding hash function algorithm and the reproduction procedure, respectively.

The scheme supposes that $PW_1, PW_2$ will be deleted immediately from the server
side upon completion of the corresponding steps because of the assumption that in
this phase, the server is fully trusted.

**Authentication:**
User $U'_i$ first inserts $SC$ into the card reader and enters the password and scans
his/her biometric features. We use $X'$ to denote the extracted biometric template.
The authentication phase is as follows.

1. The smart card computes $R$ via $Rep(\cdot)$ and calculates $PW_2 = h(R)$. The
   identical $R$ can be reproduced if and only if the difference between $X$ and $X'$
   satisfying $dis(X, X') < t$;

2. $U'_i [PW_1, SC(Data_1)] \xleftarrow{2\text{-Factor-Auth}} S [SK_1]$;
   User $U'_i$ with $(PW_1, Data_1)$ and $S$ who with $SK_1$ execute the authentication
   phase (2-Factor-Auth) of the underlying two-factor authentication protocol;

3. $U'_i [PW_2, SC(Data_2)] \xleftarrow{2\text{-Factor-Auth}} S [SK_2]$;
   $U'_i$ and $S$ run the 2-Factor-Auth again with $PW_2, Data_2$ and $SK_2$, respectively.

The user successfully passes user authentication iff $S$ is accepted in both step 2 and
step 3.
Password Changing:
The password can be changed by running password changing protocol (2-Factor-Password-Changing) in the underlying two-factor scheme after successfully logging and updating the $SC$ accordingly. The biometrics can be changed by running step 2 and step 3 in the registration phase, then the user and server execute 2-Factor-Password-Changing and update the corresponding data in $SC$.

5.3.2 Improved Framework

Based on considerations of practicality, we use the ‘fuzzy vault’ to replace the ‘fuzzy extractor’ for biometric key generation, because the Euclidean distance measurement in the ‘fuzzy vault’ has been widely accepted by the majority of biometric applications, while the distance measures in the ‘fuzzy extractor’ have not. Moreover, to enhance the efficiency and reduce the computational cost, our improved framework reduces the process from running underlying two-factor authentication scheme twice to running it once by combining the password and biometric key together and hashing it as the password of the underlying scheme. We assume that the server in the registration phase is trusted. The details are specified as follows:

Three-Factor-Registration: The process of registration include the following steps:

1. User $U_i$ chooses initial password $PW_1$, long-bit secret key (treated as the biometric key) $PW_2$, and computes $PW = h(PW_1 || PW_2)$;

2. Upon $U_i$ scanning his biometric features, the ‘fuzzy vault’ device extracts biometric template $X$ with its helper data $H$ from $U_i$’s biometric features;

3. Taking $X$, $PW_2$, and polynomial $Pol$ as inputs, $P_X(K)$ outputs locking set $L$, and the device then runs $Gen(CP,L) \rightarrow V$;

4. $U_i[PW] \xrightarrow{2\text{-factor-Reg}} S[SK] \rightarrow Data_1$, where $PW = h(PW_1 || PW_2)$.

   The user with $PW$ and the server with $SK$ run the registration phase of the underlying protocol.

5. Server stores $Data_1$ in smart card $SC$, and gives it to $U_i$;

6. $U_i$ updates $SC$ by adding $Data_2$ to it, where $Data_2 = (V,H,Rec(\cdot), De(\cdot), h(\cdot))$. $Rec(\cdot)$ and $De(\cdot)$ are the descriptions of the corresponding procedure in the fuzzy vault, and $h(\cdot)$ is the description of a hash function.
Three-Factor-Authentication:
To access services, user $U'_i$ inserts $SC$ to a card reader, which can extracts the data from the $SC$. Then, $U'_i$ inputs $PW_1$ and scans his/her biometric features, the extracted biometric template is $X'$ and its helper data is $H'$. The details are as follows:

1. The card reader extracts $X'$, $H'$ from $U'_i$’s biometric features, and reproduces $PW_2$ by the following two steps:
   Firstly, the ‘fuzzy vault’ device reproduces $Pol$ via the $Rec(·)$ procedure, if and only if input $X'$ satisfies $|X - X'| < \epsilon$; 
   Then, to reconstruct $PW_2$, taking $Pol$ as the input of $De(·)$, which outputs $PW_2$.
2. The smart card calculates $PW = h(PW_1||PW_2)$;
3. $U'_i [PW, Data_1] \xrightarrow{2\text{-factor-Auth}} S [SK]$;
   The user can successfully pass authentication if this step is success.

Three-Factor-Password-Changing:
The $PW_1$ can be changed by following steps.

1. After passing authentication, $U'_i$ sends the password changing request, inputs new password $PW_1''$, and scans the biometric template.
2. The ‘fuzzy vault’ device will recover the $PW_2$ by using the ‘fuzzy vault’ decoding scheme.
3. The smart card calculates $PW'' = h(PW_1''||PW_2)$.
4. $PW''$ is taken as the password and runs the password changing phase of the underlying protocol.

Biometric key $PW_2$ can be changed in a similar way. For this purpose, $U'_i$ chooses a new biometric key as $PW_2''$, then encrypts it via the ‘fuzzy vault’ device, outputs $V''$ and $H''$ which replaces current $V$ and $H$ of $Data_2$ in $SC$. The $SC$ calculates $PW'' = h(PW_1||PW_2'')$, then takes $PW''$ as the password and runs the password changing phase of the underlying protocol. $U'_i$ can also decide to use another finger to authenticate him. The process of finger changing is in a similar way.
5.4 Concrete Instantiation

Concrete instantiation chooses Yang et al.’s provably secure two-factor authentication protocol [YWWD08] as the underlying scheme. Yang’s scheme employs the Diffie-Hellman key exchange protocol to establish the session key, and uses an asymmetric key encryption/decryption scheme to protect the transmitting messages. In the registration phase, the server creates a credential with a long-term secret key by using a pseudorandom function and sends it to the user who then does the exclusive-or operation (xor) on it along with his/her hashed password, and stores the outputs in a smart card. Thus, only the server which has the long term secret key can generate the credential and only the user who has the password and smart card can recover the credential. To pass user authentication, the user need to recover the credential and send it to the server after encrypting it by using a public key encryption scheme. For server authentication, a secure signature scheme has been employed. The notations used in the concrete instantiation are shown in Table 5.1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ID_i$</td>
<td>User’s unique identity</td>
</tr>
<tr>
<td>$SID$</td>
<td>Server’s unique identity</td>
</tr>
<tr>
<td>$x$</td>
<td>Server’s long term secret key</td>
</tr>
<tr>
<td>$SK, PK$</td>
<td>Server’s public key pair for encryption</td>
</tr>
<tr>
<td>$SK', PK'$</td>
<td>Server’s key pair for signature scheme</td>
</tr>
<tr>
<td>$PW_1$</td>
<td>User’s password.</td>
</tr>
<tr>
<td>$PW_2$</td>
<td>User’s long-bit secret key</td>
</tr>
<tr>
<td>$SC$</td>
<td>Smart Card</td>
</tr>
<tr>
<td>$H$</td>
<td>Helper data used in the ‘fuzzy vault’</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Credential for $U_i$ created by $S$</td>
</tr>
<tr>
<td>$h$</td>
<td>One way strong hash function. ${0,1}^* \rightarrow {0,1}^l$</td>
</tr>
<tr>
<td>$P_X(K), Gen(), Rec(), De()$</td>
<td>Defined as in the ‘fuzzy vault’</td>
</tr>
<tr>
<td>$PRF_x$</td>
<td>${0,1}^k \rightarrow {0,1}^k$ pseudorandom function keyed by server’s long term secret key $x$.</td>
</tr>
<tr>
<td>$sid$</td>
<td>Session identifier.</td>
</tr>
<tr>
<td>$CT$</td>
<td>Cipher text.</td>
</tr>
</tbody>
</table>

Table 5.1: Notations in the Concrete Three-Factor Authentication
5.4.1 Concrete Protocol

The basic idea of our concrete protocol is that using PW such that $PW = h(PW_1||PW_2)$ as the password in Yang’s scheme, where $PW_1$ is the password known by the user, $PW_2$ is the biometric key which can be recovered by providing the smart card and the corresponding biometric features. A user can pass authentication only if s/he provides the correct password, smart card, and the biometric features. Thus, a three-factor authentication scheme is achieved.

**Registration**

We assume the communication channel in this phase is secure.

1. User $U_i$ obtains unique identity $ID_i$ from server $S$, and then chooses password $PW_1$, polynomial $Pol$, and biometric key $PW_2$;

2. The ‘fuzzy vault’ device extracts the biometric template and helper data $(X,H)$ from $U_i$’s biometric features, and runs procedures $P_X(PW_2) \rightarrow L$ and $Gen(CP,L)$ to encrypt $PW_2$ in $V$. Then, it calculates $PW = h(PW_1||PW_2)$.

3. Server $S$ generates credential $C_i$ such that $C_i = PRF_x(h(ID_i))$, and hides it with initial password $PW_{init}$ such that $B = C_i \oplus PW_{init}$.

4. $S$ issues $SC = \{ID_i, SigData, AutData, EncData\}$ to $U_i$. Here, $SigData$ is the description of the signature algorithm together with related parameters; $AutData = \{B,V,H,h,Rec(),De()\}$; $EncData$ is the description of an encryption algorithm together with the parameters. $h$, $Rec()$, and $De()$ are the description of the hash function and the ‘fuzzy vault’ procedures, respectively.

5. Upon receiving $SC$, $U_i$ updates $B$ by computing $B = C_i \oplus PW_{init} \oplus PW$.

**Login-and-Authentication Phase**

User $U_i’$ inserts his/her smart-card in a card reader, inputs password $PW_1’$ and scans his biometric features. The ‘fuzzy vault’ device extracts the biometric template and helper data $X’, H’$, then the ‘fuzzy vault’ device calculates $Pol = Rec(X’, H’, V, H)$, and $PW_2’ = DeP(Pol)$. The smart card $SC$ calculates $C_i’=B \oplus PW’$, where $PW’ = h(PW_1'||PW_2)$. Then, the protocol runs as follows:

1. $U_i’ \rightarrow S$: $M_1 = (ID_i, sid, g^a)$
   
   User $U_i’$ sends identity $ID_i$, session ID $sid$, and user’s session key material $g^a$ to $S$, where $a$ is a random number chosen by $U_i$;
2. $S \rightarrow U'_i$: $M_2 = (SID, sid, g^b, Sig_{SK'}(SID, ID_i, sid, g^a, g^b))$

S sends his identity $SID$, session ID $sid$, and a signature with signing key $SK'$ to $U'_i$.

3. $U'_i \rightarrow S$: $M_3 = (ID_i, sid, CT)$

$U_i$ checks the signature first. If it is not valid, $U_i$ terminates the conversation. Otherwise, $U_i$ computes $M_3 = (ID_i, sid, CT)$ and sends it to $S$, where $CT = E_{PK}(C'_i, ID_i, SID, sid, g^a, g^b)$ and $E_{PK}(M)$ denotes the asymmetric key encryption on message $M$ under public key $PK$;

4. After decrypting $CT$ by using $SK$, $S$ checks $C'_i$, and rejects $U_i$ if $C'_i = PRF_x(h(ID_i))$ does not hold. Otherwise, $S$ accepts $U_i$, and believes that they share the same session key $g^{ab}$.

**Password-Changing**

The change of password $PW_1$ contains the following steps.

1. After successfully logging in, $U_i$ chooses new password $PW'_1$.

2. $U_i$ calculates $PW_{new} = h(PW'_1 || PW_2)$ and $B_{new} = B \oplus PW \oplus PW_{new}$.

3. Replace $B$ with $B_{new}$ in the smart card.

The biometric key $PW_2$ and the biometric features can be changed in a similar way, in which case, the vault $V$ and help data $H$ also need to be updated on smart card.

### 5.4.2 Analysis of Implementation

We first analyse the capacity of the smart card. During the registration phase, the point $v_i$ in set $V$ are presented as three-tuple $v_i = (x, y, \theta)$ ($i = \{1, 2, ..., r + s\}$), where $(x, y)$ is the row and column coordinates in the image as the location, and $\theta$ is the orientation which respect to the X-axis. The number of points in vault $V$ depends on $r$ and $s$, where $r$ is the number of points which lie on $P$ and $s$ denotes the number of points which do not lie on $P$ (treated as chaff points or noise). Here, $s \approx 10r$. Nandakumar, Jain, and Pankanti \[NJ07\] showed that a 128-bit secret key requires an 8-degree polynomial to encrypt the key, and the lengths of $x, y, \theta$ are $6, 5, 5$, respectively, in field $F = GF(2^{16})$. Table 5.2 taken from \[NJ07\], shows the parameters in different databases:
5.4. Concrete Instantiation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>FVC2002-DB2</th>
<th>MSU-DBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image size</td>
<td>560 × 296 at 569 dpi resolution</td>
<td>640 × 480 at 500 dpi resolution</td>
</tr>
<tr>
<td>r in V</td>
<td>18-24</td>
<td>24-36</td>
</tr>
<tr>
<td>s in V</td>
<td>200-206</td>
<td>300-312</td>
</tr>
<tr>
<td>Total points in V</td>
<td>224</td>
<td>336</td>
</tr>
<tr>
<td>n</td>
<td>7-10</td>
<td>10-12</td>
</tr>
<tr>
<td>Length of secret key k</td>
<td>128-bit</td>
<td>128-bit</td>
</tr>
<tr>
<td>Length of V</td>
<td>448 Bytes</td>
<td>672 Bytes</td>
</tr>
</tbody>
</table>

Table 5.2: Parameters in Different Databases [NJP07]

The length of help data which depends on the points of maximum curvature in the flow curves can be ignored. Thus, only a half KB of the ‘fuzzy vault’ parameters needs to be stored in the smart card, and this is acceptable.

Now, we discuss the recognition rate of the ‘fuzzy vault’. [NNJ08, NJP07] employ genuine acceptance rate (GAR) and false acceptance rate (FAR) to analyse the recognition accuracy. The results show that both GAR and FAR are influenced by \( n \) which is the degree of polynomial. The change of \( n \) affects both GAR and FAR; \( n \) is in an inverse proportion to GAR and FAR. Fig. 5.1 are the results of both GAR and FAR in the implementation provided in [NNJ08].

![Graphs showing GAR and FAR](image)

(a) Genuine Accept Rate (GAR)  
(b) False Accept Rate (FAR)

Figure 5.1: GAR and FAR of the ‘Fuzzy Vault’ [NNJ08]

Generally, the FAR is \( 10^{-4} \) when \( n = 6 \), and it tends to zero when \( n \) is increase to 8; all the results of GAR in the figure are practical even when \( n = 12 \), where GAR > 70%. In fact, GAR may be also acceptable even it reduces to 30%, as this means that a genuine user can pass authentication successfully by trying three or four fingerprint identifications on average.
### Table 5.3: Comparison of Schemes

<table>
<thead>
<tr>
<th>Name of scheme</th>
<th>Properties</th>
<th>Store Password or Biodata in DB</th>
<th>Registration phase</th>
<th>Login-and-Authentication phase</th>
<th>Change password freely</th>
<th>Biometrics privacy</th>
<th>Key Exchange</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li and Hwang’s scheme [LH10]</td>
<td>×</td>
<td>L1</td>
<td>L1</td>
<td>×</td>
<td>×</td>
<td></td>
<td>Vulnerable to man-in-the-middle attack</td>
<td></td>
</tr>
<tr>
<td>Li et al.’s scheme [LNM+11]</td>
<td>×</td>
<td>L1</td>
<td>L1</td>
<td>√</td>
<td>×</td>
<td></td>
<td>Fails to provide strong authentication</td>
<td></td>
</tr>
<tr>
<td>Das’s scheme [Das11]</td>
<td>×</td>
<td>L1</td>
<td>L1</td>
<td>√</td>
<td>×</td>
<td></td>
<td>Vulnerable to Off-line guessing password attack</td>
<td></td>
</tr>
<tr>
<td>Kim-Lee-Yoo scheme [KLY03]</td>
<td>×</td>
<td>2 Exp</td>
<td>4 Exp</td>
<td>√</td>
<td>√</td>
<td></td>
<td>Vulnerable to impersonation attack</td>
<td></td>
</tr>
<tr>
<td>Bhargav et al.’s scheme [BSSM+07, BSSB06]</td>
<td>√</td>
<td>3 Exp</td>
<td>5 Exp</td>
<td>×</td>
<td>√</td>
<td></td>
<td>Secure under three-factor requirements</td>
<td></td>
</tr>
<tr>
<td>Fan and Lin’s scheme [FL09]</td>
<td>√</td>
<td>L1&amp;L2</td>
<td>1 E/D</td>
<td>×</td>
<td>√</td>
<td></td>
<td>Secure under three-factor requirements</td>
<td></td>
</tr>
<tr>
<td>Proposed scheme</td>
<td>×</td>
<td>L1</td>
<td>1 DH; 1 Sig; 1 E/D</td>
<td>√</td>
<td>√</td>
<td></td>
<td>Secure under three-factor requirements</td>
<td></td>
</tr>
</tbody>
</table>

X: False  
√: True  
L1: The phase only contains the hash operation and exclusive operation  
L2: The phase employs symmetric key encryption/decryption  
E/D: The phase computes asymmetric key encryption and decryption  
Exp: The phase calculates large modular exponentiation  
Sig: The participant signs and verifies digital signature  
DH: The plain Diffie-Hellman key exchange operation

### Security

<table>
<thead>
<tr>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerable to man-in-the-middle attack</td>
</tr>
<tr>
<td>Fails to provide strong authentication</td>
</tr>
<tr>
<td>Vulnerable to Off-line guessing password attack</td>
</tr>
<tr>
<td>Vulnerable to impersonation attack</td>
</tr>
<tr>
<td>Secure under three-factor requirements</td>
</tr>
</tbody>
</table>

Table 5.3: Comparison of Schemes
5.4. Concrete Instantiation

The comparison between our concrete instantiation and other three factor authentication schemes is given in Table 5.3. It is obviously that LH10, LNM+11, Das11, KLY03 support free password changing, and LH10, LNM+11, Das11 achieve lower computational cost. However, all of them have security flaws. Both BSSM+07 and FL09 are secure under the three-factor adversary model, but they do not support freely password changing and BSSM+07 does not support session key exchange. Our derived protocol protects user privacy, supports easy password changing and session key establishment, although its computation cost is not low but still acceptable.

5.4.3 Formal Security Proof of Instantiation Protocol

The formal security proof of the factor-based authentication scheme has been introduced as an open problem and a challenging issue from the point of view of security analysis HMZ+11, although some formal proofs have been provided XZF09, XZJ11, FL09, CK01, BR93a, BPR00. In HXC+11 (even in its supplementary file) Huang et al. only provided informally security discussion. In BR93a, BPR00, CK01, they provided generic models for formally proving the security of authenticated key exchange schemes, not for three-factor authentication schemes. So, the three-factor-based mutual authentication scheme which supports session key establishment has not been studied by these well-known models. This section proposes a security model for three-factor-based authenticated key exchange schemes. A formal proof of our proposed concrete scheme is also provided in our security model.

The basic idea of our concrete protocol is that a server creates credential $C$ for a user via pseudorandom function $PRF(\cdot)$ with his/her long term secret key $x$, then the user encrypts it by doing the exclusive operation along with combined password and biometric key, which outputs encrypted credential $B$ stored in the smart card. The user recovers the credential iff s/he provides the correct password and biometric features. For user authentication, the user encrypts and sends his/her identity and credential $C'$ to the server. Upon receiving it, server calculates credential $C$ with $x$ according to user identity, and compares $C$ and $C'$. If they have matched, then the server accepts the user’s request. Otherwise, the server rejects it. Server authentication has been preserved by a secure signature scheme. Assumptions for security proofs are list below:
Assumptions:

1. The ‘fuzzy vault’ scheme in $\Pi$ is secure due to [NJP07, NNJ08].

2. Information stored in smart card $SC$ can be extracted by an attacker if he/she can obtain $SC$ [Cla03].

3. No one has exactly the same biometric feature as others.

4. The case of a person without specific biometric features (such as a person without fingerprints) is ignored here since it is such a rare circumstance.

We place probabilistic polynomial time ($PPT$) adversary $A$, who can make queries to any instance, between user $U_i$ in user set $U$ and sever $S_j$ in server set $S$. Let $\Pi_{U,S}^{sid}$ denotes the user oracle, interacting with the server in session $sid$ and $\Pi_{S,U}^{sid}$ denotes the server oracle, interacting with user in the session $sid$. It is obvious that if protocol $\Pi$ is secure when $A$ knows two out of three factors, then $\Pi$ is still secure when only one factor has been leaked to $A$. Therefore, we only consider the case of two corrupted factors. The oracle queries which can be made by $A$ are defined as follows.

**Adversary Model ($AM$):**

1. $Register(\Pi, S_j)$—Upon receiving this query from $A$, server oracle acts as $S_j$ to run the registration phase with $A$, and issues identity $ID_i$ and sends smart card $SC$ to $A$.

2. $Execute(U_i, S_j, sid)$—This oracle query models all passive attackers who can eavesdrop on all messages transmitted between $U$ and $S$ in session $sid$ in $\Pi$. Upon receiving this query, $\Pi_{U,S}^{sid}$ and $\Pi_{S,U}^{sid}$ will execute protocol as $U_i$ and $S_j$ in $\Pi$, respectively. The messages exchanged between them will be recorded and sent to $A$ as responses.

3. $Send(U_i, S_j, sid, M_m, m)$—This query sends message $M_m$ with sequence of message flow $m$ to server oracle $\Pi_{S,U}^{sid}$ which simulates $S_j$, and then, the oracle will compute a respond honestly in $\Pi$, and send the response to $A$.

4. $Send(S_j, U_i, sid, M_{m'}, m')$—This query sends message $M_{m'}$ with sequence of message flow $m'$ to user oracle $\Pi_{U,S}^{sid}$ which simulates $U_i$, and then, the user oracle will compute a respond honestly in $\Pi$, and send the response to $A$. 
Upon receiving the query with $m' = \lambda$, where $\lambda$ is an empty set, from $A$, the user oracle will start a new session and send a service request message to $A$.

5. **Reveal($\prod, U_i, S_j, sid$)**—This query models the leakage of a session key in session $sid$ between user $U_i$ and server $S_j$. This query only can be made when a session key has been shared between the server and the user in session $sid$. Upon receiving this query, the user oracle will send the shared session key to $A$.

6. There are three corrupt queries:

   (a) **Corrupt($U_i, pw, SC$)**. Upon receiving this query, user oracle will send back the user $U_i$’s password and the data stored in the smart card to the adversary;

   (b) **Corrupt($U_i, pw, Bio$)**. Upon receiving this query, user oracle will send back the user $U_i$’s password and the biometric template to the adversary;

   (c) **Corrupt($U_i, SC, Bio$)**. Upon receiving this query, user oracle will send back the user $U_i$’s biometric template and the data stored in the smart card to the adversary;

   In a concrete attack, $A$ can only make one corrupt query in the target session.

7. **Test($U_i, S_j, sid$)**—This query can be made by $A$ only after a session key has been shared between $U_i$ and $S_j$ in a fresh session $sid$. If so, then a coin $b$ is tossed, if it lands $b = 0$, then this test query oracle outputs the session key. Otherwise, a fixed-length random string is returned. $A$ needs to outputs $b' = 0$ (or $b' = 1$) as the result of distinguishing the session key and a random string. $A$ can only ask this query once.

The definitions of matching conversations, secure mutual authentication and secure key exchange [BR93a] are reviewed as follows.

**Definition 5.2. (Matching Conversations):** Fix number of moves $R = 2\rho - 1$ and $R$-move protocol $\prod$. Run $\prod$ in the presence of adversary $A$ in the AM and consider two oracles $\prod^{sid}_{U,S}$ and $\prod^{sid}_{S,U}$ that engage in conversations $K$ and $K'$, respectively. $(\tau, \alpha, \beta)$ denotes that $A$ is given response $\beta$ back after asking $\alpha$ to an oracle at time $\tau$. If $\alpha = \tau$, then it means that protocol $\prod$ starts a new session. Let $*$ denotes the final decision of $R$-move protocol $\prod$. 
1. We say that $K'$ is a matching conversation to $K$ if there exist $\tau_0 \prec \tau_1 \prec \ldots \prec \tau_R$ and $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ such that $K$ is prefixed by $(\tau_0, \lambda, \alpha_1)$, $(\tau_2, \beta_1, \alpha_2)$, $(\tau_2, \beta_1, \alpha_\rho)$, $(\tau_2, \alpha_\rho)$, $(\tau_3, \alpha_2, \beta_2)$, $(\tau_0, \alpha_\rho)$, $(\tau_2, \alpha_\rho, \alpha_\rho)$, and $K'$ is prefixed by $(\tau_1, \alpha_1, \beta_1)$, $(\tau_3, \alpha_2, \beta_2)$, $(\tau_2, \alpha_\rho, \alpha_\rho)$.

2. We say that $K$ is a matching conversation to $K'$ if there exist $\tau_0 \prec \tau_1 \prec \ldots \prec \tau_R$ and $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ such that $K'$ is prefixed by $(\tau_1, \alpha_1, \beta_1)$, $(\tau_3, \alpha_2, \beta_2)$, $(\tau_2, \alpha_\rho, \alpha_\rho)$, and $K$ is prefixed by $(\tau_0, \alpha_\rho)$.

Let $\prod_{U_i, S_j}$ (or $\prod_{S_j, U_i}$) denotes that the oracle who acts as user $U_i$ (or server $S_j$) communicates with server $S_j$ (or user $U_i$). Let $\text{No} - \text{Matching}^{A, U_i}(k)$ (or $\text{No} - \text{Matching}^{A, S_j}(k)$) be the event that there exist $U_i, S_j$ and $\text{sid}$ such that $\prod_{U_i, S_j}^{\text{sid}}$ (or $\prod_{S_j, U_i}^{\text{sid}}$) has accepted $A$ as $\prod_{S_j, U_i}^{\text{sid}}$ (or $\prod_{U_i, S_j}^{\text{sid}}$), while $\prod_{S_j, U_i}^{\text{sid}}$ (or $\prod_{U_i, S_j}^{\text{sid}}$) has not engaged in a matching conversation. In other words, it is the event that user $U_i$ (or server $S_j$) believes that server $S_j$ (or user $U_i$) is communicating with him, but in fact, it is adversary $A$ who impersonates server $S_j$ (or user $U_i$).

**Remark 5.1.** The above definition is defined for the case of $R = 2\rho - 1$ moves protocol. For the case of $R = 2\rho$ moves protocol, the definition can be changed trivially. So, we are not going to discuss it here.

**Definition 5.3. (Secure Three-Factor Mutual Authentication (STMA))** We say that $\prod$ is a secure mutual authentication protocol if for any PPT adversary $A$ in the AM, the following properties are satisfied.

1. If oracles $\prod_{U_i, S_j}^{\text{sid}}$ and $\prod_{S_j, U_i}^{\text{sid}}$ have matched conversations, then they accept each other.

2. $\prod_{U_i, S_j}^{\text{sid}}$ accepted implies a matching conversation: the probability of $\text{No} - \text{Matching}^{A, U_i}(k)$ is negligible, where $S_j$ should not be registered by $A$. (Secure server authentication)

3. $\prod_{S_j, U_i}^{\text{sid}}$ accepted implies a matching conversation: the probability of $\text{No} - \text{Matching}^{A, S_j}(k)$ is negligible, where $U_i$ should not be registered by $A$. (Secure user authentication)

**Definition 5.4. (Secure Three-Factor Authenticated Key Exchange (STAKE))** A Protocol $\prod$ is called STAKE if the following properties hold for any adversary $A$ in the AM:
• $\prod$ is a STMA protocol;

• if the session is fresh in protocol $\prod$, and both $\prod_{U_i,S_j}^{sid}$ and $\prod_{S_j,U_i}^{i}$ complete matching conversations, then they have shared the same session key;

• the advantage $\text{Adv}^A(k)$ is negligible.

Note that:

1. Session freshness requires satisfying follow probabilities:
   • $\prod_{U_i,S_j}^{sid}$ and/or $\prod_{S_j,U_i}^{sid}$ accepted;
   • no queries to reveal the session key have been made to $\prod_{U_i,S_j}^{sid}$ or $\prod_{S_j,U_i}^{sid}$;

2. $\text{Adv}^A(k) = |\text{Good} - \text{guess}^A(k)| - \frac{1}{2}$, where the Good-guess is the event such that $A$ wins the game of $\text{AKE}$ [BR93a];

To prove the security of our concrete scheme, we show that if $A$ can successfully pass user or server authentication with a non-negligible probability, then we can construct a PPT Turing machine $T$ to solve the hard problems by employing $A$ with a non-negligible probability. The concrete protocol is reviewed as follows:

1. $U_i \to S$: $M_1= (ID_i, sid, g^a)$

2. $S \to U_i$: $M_2= (SID, sid, g^b, \text{Sig}_{SK'}(SID, ID_i, sid, g^a, g^b))$

3. $U_i \to S$: $M_3= (ID_i, sid, CT)$, where $CT = E_{PK} \left( C'_i, ID_i, SID, sid, g^a, g^b \right)$

4. $S$ checks credential $C'_i$. $U_i$ will pass user authentication if and only if $C'_i = \text{PRF}_x (h(ID_i))$.

Now, the shared session key is $g^{ab}$.

**Lemma 5.1. (Secure User Authentication)** In the proposed protocol $\prod$, if the pseudorandom function (PRF) is replaced by an ideal random function, the public key encryption (PKE) scheme is secure against CCA2 attack, and $\prod_{S_j,U_i}^{sid}$ has accepted, then for any PPT adversary $A$ in the AM, the probability of $\text{No-Matching}^{A,S_j}(k)$ is negligible.

**Proof.** This can be proved by contradiction. If there exists an adversary $A$ who can pass user authentication with non-negligible probability $\epsilon$, then we can construct a PPT Turing machine $T$ without known secret key $x$ to solve a hard problem, i.e.
winning the game of $PRF$ (Game-PRF), with a non-negligible probability by using $A$.

In the Game-PRF, a challenger sends two different plaintexts $P_0$ and $P_1$ to the $PRF$ test query, then the $PRF$ test query will answer with result $PRF_x(P_b)$ to the challenger, where $b$ is the result of coin tossing. After that, the challenger needs to output $b' = 0$ or $b' = 1$ as its guess to value $b$. Let $Pr_{adv}[PRF]$ be the advantage of guessing, which is defined as $Pr_{adv}[PRF] = Pr_{win} - \frac{1}{2}$, where $Pr_{win}$ denotes the correct guessing rate. In this game, we give the challenger a power to ask the output of $PRF$ by providing a message $M_{pt}$. Upon receiving this request, the $PRF$ oracle $\prod_{PRF}$ will output a response $PRF_x(M_{pt})$ by using server’s secret key $x$. Here we require that the asked message $M_{pt}$ can not be sent as one of input to the $PRF$ test query.

The basic idea is that to win Game-PRF, $T$ simulates an environment of our concrete protocol to convince adversary $A$ that this simulation is the real environment of concrete protocol execution. On the other side, $A$ should only has a negligible probability to know the truth, i.e. this is not a real protocol environment but a simulation. In such a simulation, $T$ communicates with $A$ who has the ability to break our concrete protocol in some way in a session with session ID $sid$ with a non-negligible probability. Then, in order to win Game-PRF, $T$ will make use of $A$’s ability to make the decision of which input message has been used to generate the output $PRF_x(P_b)$ with a non-negligible probability.

The simulation is constructed as follows. In the simulation, $T$ answers all oracle queries made by $A$. To achieve this goal, $T$ needs to setup $(SK, PK)$ for the public key scheme and $(SK', PK')$ for the signature scheme, while $T$ does not know the value of long term secret key $x$ which is for $\prod_{PRF}$. $\prod_{U_i,S_j}^{sid}$ denotes the user oracle who has password $PW_1$, smart-card $SC$, and corresponding biometric template $X$ which can recover biometric key $PW_2$ with the $SC$. $\prod_{S_j,U_i}^{sid}$ denotes the server oracle who has $PRF$ oracle $\prod_{PRF}$. In our concrete protocol, $A$ can make the following queries:

- **Register($\prod, S_j$)**—Upon receiving this query from $A$, $T$ runs the registration phase with $A$ with the help of $\prod_{PRF}$. In particular, $T$ needs to record all identities which have been registered into what we called compromised table.

- **Execute($U_i, S_j, sid$)**—In $\prod_{i,1}$, $\prod_{U_i,S_j}^{sid}$ and $\prod_{S_j,U_i}^{sid}$ generate and record all messages transmitted between $U_i$ and $S_j$ in session $sid$, then send these messages
to $A$.

- **Send($U_i, S_j, sid, M_m, m$)—** $A$ can send $M_1$ to $T$, then $T$ responds to $M_2$ by using $SK'$ to sign a signature as the protocol specified. Upon receiving $M_3$ from $A$, $T$ sends the result of user authentication according to $M_1$ and $M_3$ by using $SK$ to decrypt the ciphertext and asking $\prod_{PRF}$ in order to verify the credential.

- **Send($S_j, U_i, sid, M_{m'}, m'$)—** Upon receiving a new session query $Send(S_j, U_i, sid, M_{m'}, m')$, $T$ asks $\prod_{sid}^{U_i, S_j}$ to send the first message $M_1$ to $A$. After receiving the corresponding message $M_2$, $T$ checks the signature by using $PK'$. If the signature is valid, $T$ asks $\prod_{PRF}$ and encrypts its output to form message $M_3$.

- **Corrupt($U_i, \text{factor}_a, \text{factor}_b$)—** Upon receiving this query, $\prod_{sid}^{U_i, S_j}$ will send the corresponding two factors according to $a$ and $b$, where $a, b \in \{pw, SC, Bio\}$ and $a \neq b$.

If $A$ can pass user authentication successfully with a non-negligible probability without asking $\prod_{sid}^{U_i, S_j}$, there must exist a matching conversation between $A$ and $T$ who simulates server $S_j$ if the following happens. First, $A$ asks $\text{Corrupt}(U_i, \text{factor}_a, \text{factor}_b)$ to obtain two factors, then sends the first message to $T$ who then responds with the second message. Finally, $A$ forms the third message to $T$.

Now, we show how $T$ makes use of $A$ to win Game-PRF with non-negligible advantage as follows. We assume that $A$ attacks at least once among $q_s$ sessions, while $T$ does not know which session $A$ is going to attack. Now, $T$ chooses a session out of $q_s$ sessions randomly. Then, the probability of $A$ passing user authentication in this session is $\frac{1}{q_s} \cdot \epsilon$.

To avoid the case that $A$ found that this environment is only a simulation, in the rest $q_s - 1$ sessions, $T$ redirects the identity $ID_r$, which is included in the first message, to oracle $\prod_{PRF}$ which will respond $PRF_x(ID_r)$ back to $T$. Then, $T$ records this identity into the compromised table and checks whether $A$ has passed the user authentication by matching $PRF_x(ID_r)$ with the credential which is encrypted in the third message. If they are matched, then $T$ responds to $A$ that $T$ accepts $A$’s login request. Otherwise, $T$ rejects $A$’s request. For these sessions, $T$ just randomly guesses the value of $b$, so the probability that $T$ wins the game is $\frac{1}{2}$.

To use $A$, after receiving first message $M_1 = (ID_{new}, sid, g^a)$, $T$ forms $M_2 = (\text{SID}, sid, g^b, \text{Sig}_{SK'}(\text{SID}, ID_{new}, sid, g^a, g^b))$ by using $SK'$ and sends it to $A$. If $A$
can successfully pass user authentication, s/he must be able to forge third message \( M_3 = (ID_{\text{new}}, sid, CT) \), where \( CT = E_{PK}(C'_{\text{new}}, ID_{\text{new}}, SID, sid, g^a, g^b) \). Now, \( T \) requires to start the Game-PRF by choosing two distinct messages \( y_0 = h(ID_{\text{new}}) \) and \( y_1 = R_1 \), and sends \((y_0, y_1)\) to the PRF test query. The query responds \( PRF_x(y_b) \) to \( T \), then \( T \) decrypts \( CT \) to recover \( C'_{\text{new}} \) and checks whether the response is the same as \( C'_{\text{new}} \). If it is, then it outputs \( b' = 0 \) as the guessed result of \( b \). Otherwise, it outputs \( b' = 1 \).

We now analyze the probability of game winning. We assume that \( A \) forges user \( U_{\text{new}} \), and passes user authentication successfully in polynomial time \( \tau \), with non-negligible probability \( \epsilon \), asking \( q_R \) times \( \text{Register}(\prod, S_j) \), \( q_E \) times \( \text{Execute}(U_i, S_j, sid) \), \( q_S \) times \( \text{send} \) query in \( q_s \) sessions. The formula of calculating probability \( Pr_{\text{adv}}[PRF] \) of three different corrupting cases should be the same but with different \( \epsilon \) because we do not care how \( A \) can pass the user authentication. If \( A \) does not select this special session, the probability of game wining without the help of \( A \) is \( \frac{1}{2} \). Otherwise, if \( A \) indeed attacks this special session chose by \( T \), then the probability is concerned as follows. The probability of \( A \) pass authentication is \( \epsilon \), so the probability that we win the Game-PRF is \( (\epsilon \cdot 1 + (1 - \epsilon) \cdot \frac{1}{2}) \). Because if \( A \) has passed authentication, then we have 100% probability to win the game. On the other side, \( A \) may also failed with the probability of \( (1 - \epsilon) \), in this case, we have \( \frac{1}{2} \) probability to win the game. Thus,

\[
Pr_{\text{adv}}[PRF] = \frac{1}{q_s} \cdot \left( \epsilon \cdot 1 + (1 - \epsilon) \cdot \frac{1}{2} \right) + \frac{q_{sq_s} - 1}{q_s} \cdot \frac{1}{2} - \frac{1}{2}
\]

It is clear that \( Pr_{\text{adv}}[PRF] \) is non-negligible since \( \epsilon \) is non-negligible, and \( T \) spends \( \tau' = \tau + \tau_2 \) time to win games, where \( \tau_2 \) is the executing time of \( T \) interaction with the test query. It is obvious that both \( \tau \) and \( \tau' \) are polynomial times, thus, \( \tau' \) is also a polynomial time. Therefore, \( T \) can win Game-PRF with non-negligible advantage \( Pr_{\text{adv}}[PRF] \), and this contradicts assumption.

**Lemma 5.2.** (Secure Server Authentication) In proposed protocol \( \Pi \), if the signature scheme is unforgeable against adaptive chosen message attacks, and \( \Pi^{\text{sid}}_{U_i, S_j} \) has accepted, then for any PPT adversary \( A \) in the AM, the probability of \( \text{No-Matching}^{A,U_i}(k) \) is negligible.
Proof. This can be proved by contradiction. If $A$ has been accepted by $\prod_{U_i,S_j}^{sid}$ with non-negligible probability of $\text{No-Matching}^{A,U_i}(k)$, then we can construct a PPT machine $T$ which can win the Game-UFCMA \cite{GMR88} by employing $A$.

In Game-UFCMA, there is a signature signing oracle $\prod_{\text{Sign}}$. A challenger who has got the $PK'$ can make a signing query to a signature on any message $M_i$, and can also verify the signature by using $PK'$. Finally, the challenger outputs new message $M_{new}$ which the signing oracle has not been asked to sign together with a forged signature. The challenger wins if the signature is valid for $M_{new}$ under $PK'$.

Let $\Pr_{\text{win}}[\text{SIG}]$ denotes the probability advantage of game winning.

The basic idea is that to win Game-UFCMA, $T$ simulates an environment of our concrete protocol to convince adversary $A$ that this simulation is the real concrete protocol. On the other side, $A$ should only has a negligible probability to know the truce, i.e. this is not a real protocol environment but a simulation. In such simulation, $T$ communicates with $A$ who has the ability to successfully forge server’s signature in a session with session ID $sid$ with a non-negligible probability. Then, $T$ will make use of $A$’s ability to win Game-UFCMA with a non-negligible probability.

To use $A$, $T$ need to simulate $A$’s view as follows. In the simulation, $T$ answers all oracle queries made by $A$. To achieve this goal, $T$ needs to setup all parameters except signing key $SK'$. In our concrete scheme, $A$ can ask following quires:

- $\text{Execute}(U_i,S_j,sid)$— In $\prod_{U_i,S_j}^{sid}$ and $\prod_{S_j,U_i}^{sid}$ generate and record all messages transmitted between $U_i$ and $S_j$, then send them to $A$.

- $\text{Send}(U_i,S_j,sid,M,m)$— $A$ can send $M_1$ to $T$, then $T$ responds $M_2$ by asking the $\prod_{\text{Sign}}$ of $\prod_{S_j,U_i}^{sid}$. Upon receiving $M_3$ from $A$, $T$ sends the result of user authentication according to $M_1$ and $M_3$.

- $\text{Send}(S_j,U_i,sid,M_{m'},m')$— Upon receiving new session query $\text{Send}(S_j,M,\lambda)$, $T$ asks $\prod_{U_i,S_j}^{sid}$ to send first message $M_1$ to $A$. After receiving corresponding $M_2$, $T$ checks the signature, and forms $M_3$ if the signature is valid.

If $A$ can successfully pass server authentication with a non-negligible probability, there must exist a matching conversation between $A$ and $T$ who simulates user $U_i$ if the following happens. In the simulation, first, $T$ chooses message $M_1 = (T,sid,g^a)$, and sends it to $A$. If $A$ can successfully pass server authentication, then $A$ will form message $M_2 = (SID,sid,g^b,\text{Sig}_{SK'}(SID,T,sid,g^a,g^b))$ and send it to $T$. 

To win the Game-UFCMA with $A$’s help, $T$ sends $M = (SID, T, sid, g^a, g^b)$ together with the signature in $M_2$ to the test query. We assume that $A$ forges server $S$ and passes server authentication successfully in polynomial time $\tau$, with non-negligible probability $\epsilon$, asking $q_E$ times to $\text{Execute}(U_i, S_j, sid)$ and $q_S$ times to send a query, which contains $q_r$ times $\text{Send}(S_j, U_i, sid, M_m, m')$. Let $\eta$ be the probability of $T$ winning Game-UFCMA when $A$ has failed to pass server authentication. The probability is concerned as follows. In $q_s$ times send query made by $A$, we choose one query to help us to answer the Game-UFCMA. The probability of $A$ pass sever authentication is $\epsilon$, so the probability of we win the Game-UFCMA is $(\epsilon \cdot 1 + (1 - \epsilon) \cdot \eta)$. Because that if $A$ has passed authentication, then we have 100% probability to win the game. On the other side, $A$ may also failed with the probability of $(1 - \epsilon)$, in this case, we have the probability of $\eta$ to win the game. For the rest queries, the probability of game wining without the help of $A$ is $\eta$. Thus,

$$
\Pr_{\text{win}}[SIG] = \frac{1}{q_s} \cdot (\epsilon \cdot 1 + (1 - \epsilon) \cdot \eta) + \frac{q_s-1}{q_s} \cdot \eta
$$

It is clear that $\Pr_{\text{win}}[SIG]$ is non-negligible since $\epsilon$ is non-negligible. The time $T$ spent to win the games is $\tau' = \tau + \tau_3$, where $t_3$ is the executing time of $T$ spends in GAME-UFCMA. $\tau'$ is a polynomial time because both $\tau$ and $\tau_3$ are polynomial times. Therefore, we can construct PPT machine $T$ to win Game-UFCMA of the signature scheme, with non-negligible probability, and this is a contradiction.

**Theorem 5.3.** (Secure Three-Factor Mutual Authentication (STMA)) In proposed protocol $\Pi$, if: (A) the PRF is replaced by an ideal random function and PKE scheme is secure against CCA2 attack; (B) the signature scheme is unforgeable against chosen message attack; (C) at least one of $\text{Send}^{sid}_{U_i, S_j}$ and $\text{Send}^{sid}_{S_j, U_i}$ has accepted; then for any PPT adversary $A$ in the AM, the probabilities of both No-Matching$^{A_{U_i}}(k)$ and No-Matching$^{A_{S_j}}(k)$ are negligible.

**Proof.** Obviously, the first condition of Definition 5.3 holds because it is easy to verify that our concrete protocol is correct. In addition, by Lemma 5.1 and Lemma 5.2 the second and third conditions of Definition 5.3 also hold. Therefore, Theorem 5.3 holds.

**Theorem 5.4.** (Secure Three-Factor Authenticated Key Exchange (STAKE)) In proposed protocol $\Pi$, if (A) the PRF is replaced by an ideal random function and the
5.4. Concrete Instantiation

A PKE scheme is secure against CCA2 attack; (B) the signature scheme is unforgeable against chosen message attack; then for any PPT adversary $A$ in the AM, the advantage $Adv^A(k)$ of $A$ winning the game of AKEP in a fresh session is negligible.

Proof. According to the Definition 5.4, STAKE need to meet three conditions. The first condition is that protocol $\Pi$ is required to satisfies STMA. This condition is achieved because Theorem 5.3. The second condition is that for a fresh session in protocol $\Pi$, if complete conversations are matched, then the same session key must be shared between these two communicating parties. This condition is achieved because that in our concrete scheme, the key exchange is the plain two-move Diffie-Hellman protocol [CK01], and this condition is a well-known property and it was proved. For the third condition, the advantage $Adv^A(k) = |\Pr[Good - guess^A(k)] - \frac{1}{2}|$ is non-negligible due to [CK01]. Thus, $\Pi$ is a secure three-factor authenticated key exchange protocol.

5.4.4 Privacy Discussion

The proposed framework provides strong protection of user privacy. First, the server does not know any information about the user’s biometric template since the user need not provide biometric features to the server. Second, the information in SC is also unable to leak biometric information to others. In SC, only vault $V$ and helper data $H$ are related to the biometric features, however, $V$ has been added along with a large number of chaff points as noise. Thus, the probability of successfully recovering the biometric template is negligible due to [NJP07]. Moreover, helper data $H$ in the ‘fuzzy vault’ does not reveal the user’s biometric templates since they are global features which do not leak any information of local characteristics [NJP07] and two different finger templates can extract very similar helper data [NJP07].

A different $PW_2$ has been chosen, then a new $V$ has been generated. In another words, the same biometric feature can encrypt different keys, and output different vault $V$. Thus, a user can use the same biometric feature in different servers with different biometric keys, and output different vault $V$. 
5.5 Conclusion

The proposed improved framework for three-factor authentication is efficient and practical in distributed systems and networks. The framework upgrades two-factor authentication schemes to three-factor authentication schemes; the derived scheme protects user’s privacy, and enhances security. In addition, a provably secure concrete authentication scheme has been provided with formal security proof and an analysis which shows the concrete scheme is more secure and practical.
Chapter 6

Conclusion

This chapter concludes the thesis in two parts: the summary of contributions and the promotion of open problems.

6.1 Contributions

This thesis focuses primarily on two techniques for remote user authentication: single sign-on and three-factor authentication. It aims: (a) to prevent attacks on SSO mechanisms by analysing and formally defining SSO mechanisms; (b) to provide a provably secure SSO scheme based on the proposed formal model and (c) to offer a generic framework of three-factor authentication with provably secure concrete instantiation for the user who has higher security requirements. The main contributions of this thesis are as follows:

- Chapter 3 first provides some new insights into a recent single sign-on scheme proposed by Chang and Lee [CL12]. Next, based on this analysis, Chapter 3 points out the shortcomings of the Chang-Lee scheme and identifies two potential attacks with an analysis of their success probability. In particular, these two impersonation attacks show that their scheme is actually insecure as it fails to meet credential privacy and soundness of authentication. This chapter also makes a carefully analysis on the issues of how to design single sign-on scheme. Finally, the drawbacks of the Chang-Lee scheme are overcome by employing the efficient verifiable encryption of RSA signatures (RSA-VES) which was proposed for fair exchange by Ateniese [Ate99].

- In Chapter 4 we have formalised the security model of single sign-on with authenticated key exchange. In particular, we have pointed out the difference between soundness and credential privacy. The proposed model presents
a unified definition of formally specifying soundness and credential privacy for authenticated key exchange single sign-on (AKESSO). According to the formal model, this chapter proposes a provably secure single sign-on authentication scheme which satisfies soundness, preserves credential privacy, meets user anonymity, supports session key exchange. Due to its high efficiency, the scheme is suitable for mobile device users in distributed environments.

- Chapter 5 proposes an improved generic framework for three-factor authentication. This framework can upgrade a two factor authentication scheme to a three factor authentication scheme. The derived three-factor scheme is suitable for environments where the underlying two-factor scheme is specified. Compare with Huang et al.’s scheme [HXC+11], the proposed generic framework enhances efficiency and it is more practical. A provably secure concrete instantiation of the generic framework is also provided. In particular, we have provided an performance analysis, a formal security proof and a privacy discussion of the concrete instantiation.

6.2 Open Problems

As mentioned in Chapter 3, the open problems are to formally define authentication soundness and construct efficient and provably secure single sign-on schemes. Han et al.’s model [HMSY10] requires additional PKI for users but it does not require the third party to be fully trusted. Our formal model of SSO does not require users holding public key certificate, however, it may be not mature because it requires a fully trusted third party. So, another challenge is how to provide the same security level while reducing the trust level of the third party, and without requiring PKI for users. These challenges may be considered as a future work.
Bibliography


