

國立臺灣大學電機資訊學院電機工程學研究所

博士論文

Graduate Institute of Electrical Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Doctoral Dissertation



兼容的身分認證與金鑰交換協定

CAKE: Compatible Authentication and
Key Exchange Protocol

莊允心

Yun-Hsin Chuang

指導教授：雷欽隆 教授

Advisor: Professor Chin-Laung Lei

中華民國 109 年 10 月

October 2020




摘要



隨著網際網路和無線網路的普及以及物聯網 (IoT) 的快速發展，遠端應用程式逐漸融入我們的日常生活。遠端身分認證與金鑰交換或金鑰協議(AKE/AKA)是一種能讓使用者和伺服器相互認證並建立共同會議密鑰的機制，以達到能在開放式網路中安全地進行通訊。認證機制從簡單的單因子密碼認證發展到較複雜的雙因子和三因子(多因子)認證，來保護資訊不被未經授權者存取。與雙因子身分認證機制相比，三因子(多因子)認證機制能抵抗重送攻擊及智慧卡遺失攻擊。近年來，由於隱私意識的抬頭，為了保護個人隱私，使用者會有以匿名方式進行身分認證的需求，許多滿足使用者匿名性的身份認證與金鑰交換或金鑰協議(AAKE/AKA)已陸續被提出，以防止使用者的真實身份被洩露。在傳統的匿名身份認證機制中，即使使用者使用匿名身份登入，由於使用者每次登入皆使用相同的匿名身份，故可藉由登入與相互認證時所傳輸的訊息來追蹤使用者，使用者不可追蹤性的概念因此被提出來討論，以避免使用者因洩漏傳輸資訊而被追蹤。

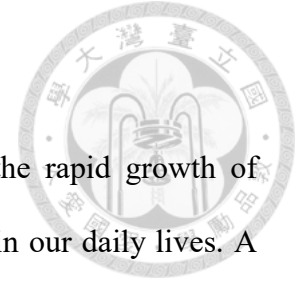
本論文研究了現今具使用者隱私保護的三因子認證機制，根據對這些機制的觀察與討論，我們提出了「具隱私保護且適用於多重伺服器環境的身分認證及金鑰協定」的設計方針。依據設計方針，我們設計了三個遠端身分認證及金鑰協定，分別適用於一般多重伺服器環境、遠距醫療系統 (TMIS)、及物聯網(IoT)，它們因應不同情境的需求而具有不同的特性。傳統的遠端身分認證及金鑰交換協定僅提供使用者與伺服器之間的認證或使用者和使用者之間的認證。我們創先提出「兼容的身分認證與金鑰交換(CAKE)」的概念以提供跨類別的身分認證以及金鑰交換，並且就具有第五代行動通訊技術的智慧城市為例提出一個具體實現，並且將其擴



展為具有使用者隱私保護的兼容身分認證與金鑰交換(ACAKE)協定。此協定是史上第一個兼具兼容身分認證、多因子認證、適用於多重伺服器環境、使用者匿名性、使用者不可追蹤性、完全向前保密性、會員可撤銷、獨立認證、無須儲存表單、無須分配公開金鑰、及正規證明… 等特性的遠端身分認證及金鑰交換協定，它不僅適用於智慧城市，也適用於其他單一類別、雙重類別、及多角色類別的應用情境。對於所提出的四個協定，我們皆提供了完整的正規安全性證明，並將其與現今相關協定進行比較，以顯示我們的協定的優點與貢獻。

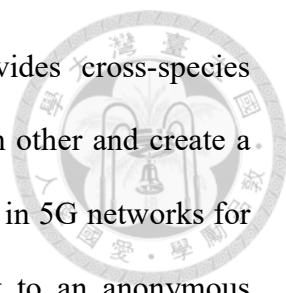
關鍵字：生物特徵、物聯網、金鑰協議、隱私、物理不可複製函數、智慧城市、遠距醫療系統、第五代行動通訊技術。

Abstract



With the popularity of Internet and wireless networks and the rapid growth of Internet of Things (IoT), remote applications gradually participate in our daily lives. A remote authentication and key exchange (AKE) or key agreement (AKA) protocol is a mechanism for letting authorized users and servers authenticate mutually and establish a session key to communicate securely through open networks. Numerous AKE/AKA protocols have been developed from a simple password based authentication to two-factor and three-factor (multi-factor) authentication for protecting information or resources from the unauthorized users. In comparison with the password based and the two-factor AKE/AKA protocols, a three-factor (multi-factor) AKE protocol can withstand replay attacks and prevent stolen smart card attacks. With the rapidly development of the sense of privacy, users want to access remote servers anonymously; hence, many anonymous AKE/AKA (AAKE/AAKA) protocols are proposed to prevent the leakage of user's identity. In an ordinary AAKE/AAKA protocol, a user logs in to the servers by a duplicate anonymous identity in each session which causes the relationship between each login is exposed; hence, the concept of user untraceability has been proposed recently to prevent users being tracked by the transmitted messages.

In this dissertation, we survey relevant three-factor AAKE/AAKA protocols. According to the observation and discussion of these relevant protocols, we propose the guidelines for designing a secure AAKE/AAKE protocol. We then obey the guidelines to propose three AKE protocols, which are designed for general multi-server environments, Telecare Medical Information Systems (TMIS), and Internet of Things (IoT), respectively. All of the existing AKE/AKA protocols are designed for either client-server or client-client authentication. We bring up the concept of a compatible



authentication and key exchange (CAKE) protocol, which provides cross-species authentication that any two valid entities can authenticate with each other and create a secure session key for secure communications. We take a smart city in 5G networks for example to propose a three-factor CAKE protocol, and extend it to an anonymous CAKE (ACAKE). This protocol is the first AAKE protocol that simultaneously achieves compatible authentication, three-factor authentication, applicability of multi-server environments, user anonymity, user untraceability/unlinkability, perfect forward secrecy, member revocation, independent authentication, table free, public key announcement free, and formal security proof. The proposed CAKE/ACAKE protocol is not only applicable to smart cities but also applicable to other present systems. We give formal security proofs of the four proposed protocols, analyze their performances, and compare them to the relevant protocols to show the advantages and contributions.

Keywords: biometric, IoT, key agreement, privacy, PUF, smart city, TMIS, 5G.

誌謝



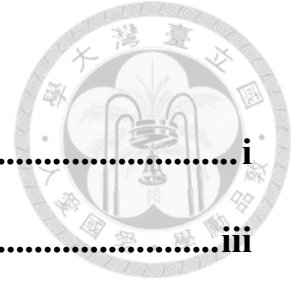
感謝求學路上眾多恩師們的教導與栽培，尤其是雷欽隆教授、曾育民教授、劉康滿教授、和黃森山教授。大學時期的教授們以及導師黃森山教授為我的數學知識打下深厚的基礎，實習階段的指導教授劉康滿教授增進我的教學技巧。碩士班指導教授曾育民教授帶領我進入研究的殿堂，讓我體驗到研究的樂趣，點燃我對學術研究的熱愛，並奠基我的研究之路。博士班指導教授雷欽隆教授很溫暖與寬厚，讓我自由選擇研究議題，並適時給予指導與協助，指導我發表了幾篇刊登在高水準國際期刊和國際會議的論文。非常感謝雷教授和曾教授的指導，讓我能順利完成此篇博士論文。

感謝丈夫、家人、同事、朋友們的鼓勵與支持，讓我能家庭、事業、學業兼顧，尤其是貼心的丈夫，時常照顧兩位年幼的孩子，讓我能有時間完成論文。感謝研究路上許多一起集思廣益、共同討論的夥伴們：蔡東佐學長、紀博文學長、許宏誌同學、王銘宏同學、郭建廷學長、吳睿迪同學、阮鶴鳴學長、蔡佩真學妹、陳亭霓學妹、范宜安學弟、曾理學弟、以及所有 DCNSLab 的夥伴們。感謝系辦陳美月小姐及行政人員承辦業務及林家慶學弟協助試場事務及代辦許多相關手續。

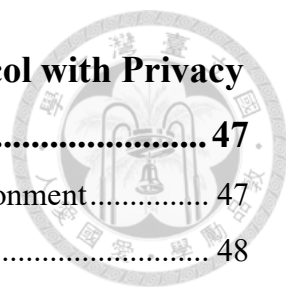
感謝雷欽隆教授、郭斯彥教授、和顏嗣鈞教授協助審查研究計畫，感謝雷欽隆教授、曾育民教授、謝續平教授、李育杰教授、廖宜恩教授、郭斯彥教授、顏嗣鈞教授、王勝德教授、和紀博文助理教授協助審查論文初稿，以及許多匿名的審稿者，他們寶貴的意見讓此論文更臻完備。

莊允心 2020 年 10 月

Contents



摘要	i
Abstract	iii
誌 謝	v
List of Figures	ix
Chapter 1 Introduction	1
1.1 Research Motivation	1
1.2 Objectives and Contribution	8
1.3 Dissertation Organization	9
Chapter 2 Preliminaries	11
2.1 Biometric	11
2.2 Physical Unclonable Function	11
2.3 Fuzzy Extractor	13
2.4 Elliptic Curve Cryptography	15
2.5 Bilinear Pairing	15
2.6 Mathematical Problems and Assumptions	16
2.7 Notations	17
Chapter 3 Related Work	19
3.1 Three-Factor AAKA	19
3.1.1 Single Server Environments	20
3.1.2 Pseudo Multi-Server Environments	21
3.1.3 Multi-Server Environments	23
3.2 Three-Factor AAKA for TMIS	25
3.3 AKE for IoT	29
3.4 AKE for a Smart City	30
3.5 Design Guideline	32
Chapter 4 Security Model	40
4.1 Threat Assumptions	40
4.2 Adversarial Model	41



Chapter 5 An Independent Three-Factor AKA Protocol with Privacy Preserving for Multi-Server Environments..... 47

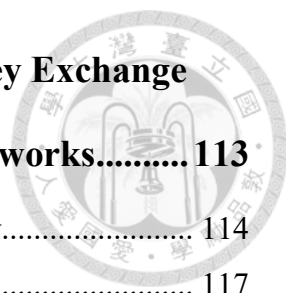
5.1 The Framework of an AAKA Protocol in a Multi-server Environment..... 47
5.2 The Proposed Protocol (Protocol 1: General AAKA) 48
5.2.1 Initialization Phase..... 49
5.2.2 Registration Phase..... 49
5.2.3 Login and AKA Phase 51
5.2.4 Password and Biometric Change Phase..... 55
5.3 Characteristic Analysis 56
5.4 Security Analysis 58

Chapter 6 Privacy Protection for TMIS with Multiple Servers Using a Biometric-based AKA Protocol 73

6.1 The Framework of an AAKA Protocol in TMIS 74
6.2 The Proposed Protocol (Protocol 2: AAKA for TMIS)..... 75
6.2.1 Initialization Phase..... 75
6.2.2 Registration Phase..... 76
6.2.3 Online Update Phase..... 78
6.2.4 Login and AAKA Phase 81
6.2.5 Password and Biometric Change Phase..... 83
6.3 Characteristic Analysis 83
6.4 Security Analysis 85

Chapter 7 PUF Based AKE Protocol for IoT without Verifiers and Explicit CRPs..... 94

7.1 The Framework of a PUF based AKE Protocol for IoT 94
7.2 The Proposed Protocol (Protocol 3: AKE for IoT)..... 95
7.2.1 System Assumptions..... 95
7.2.2 Initialization Phase..... 96
7.2.3 Registration Phase..... 97
7.2.4 IoT Node AKE Phase 101
7.3 Characteristic Analysis 103
7.4 Security Analysis 104



Chapter 8 CAKE: Compatible Authentication and Key Exchange

Protocol — Taking a Smart City for Example in 5G Networks..... 113

8.1 The Framework of a CAKE/ACAKE Protocol for a Smart City.....	114
8.2 The Proposed Protocol (Protocol 4: CAKE).....	117
8.2.1 Initialization Phase.....	117
8.2.2 Registration Phase.....	118
8.2.3 Online Update Phase.....	122
8.2.4 CAKE Phase	124
8.2.5 Password and Biometric Change Phase.....	127
8.3 The Proposed Protocol (Protocol 4: ACAKE).....	128
8.3.1 ACAKE Phase	129
8.4 Characteristic Analysis	130
8.5 Security Analysis	132

Chapter 9 Performance Analysis and Comparisons..... 150

9.1 Performance Analysis of the Proposed Protocols.....	150
9.1.1 Our Protocol 1 (General AAKA).....	153
9.1.2 Our Protocol 2 (AAKA for TMIS)	153
9.1.3 Our Protocol 3 (AKE for IoT)	154
9.1.4 Our Protocol 4 (CAKE/ACAKE)	156
9.2 Comparisons	157
9.2.1 Four Proposed Protocols.....	157
9.2.2 Our Protocol 1 and Relevant Secure AAKA Protocols	159
9.2.3 Our Protocol 2 and Relevant Secure AAKA Protocols for TMIS	161
9.2.4 Our Protocol 3 and Relevant PUF Based AKE Protocols for IoT	162
9.2.5 Our Protocol 4 and Relevant AKE Protocols for a Smart City	162

Chapter 10 Conclusions and Future Work..... 166

10.1 Conclusions.....	166
10.2 Future Work.....	167

Bibliography 170

List of Figures



Figure 1. Implementation diagram for an efficient fuzzy extractor	14
Figure 2. The framework of an AAKA protocol in a multi-server environment	48
Figure 3. Server registration phase of our Protocol 1 (General AAKA).....	53
Figure 4. User registration phase of our Protocol 1 (General AAKA).....	53
Figure 5. Login and AKA phase of our Protocol 1 (General AAKA).....	53
Figure 6. The framework of an AAKA protocol in TMIS	75
Figure 7. Server registration phase of our Protocol 2 (AAKA for TMIS)	77
Figure 8. User registration phase of our Protocol 2 (AAKA for TMIS)	77
Figure 9. Online update phase of our Protocol 2 (AAKA for TMIS)	79
Figure 10. Login and AAKA phase of our Protocol 2 (AAKA for TMIS).....	80
Figure 11. The framework of our Protocol 3 (AKE for IoT)	95
Figure 12. Registration phase of our Protocol 3 (AKE for IoT)	98
Figure 13. AKE phase of our Protocol 3 (AKE for IoT).....	98
Figure 14. Components of a smart city	114
Figure 15. The framework of the proposed CAKE/ACAKE protocol	116
Figure 16. Registration phase for a natural person in our Protocol 4.....	120
Figure 17. Registration phase for a server in our Protocol 4.....	120
Figure 18. Registration phase for an IoT device in our Protocol 4	120
Figure 19. Online update phase of our Protocol 4.....	125
Figure 20. CAKE phase of our Protocol 4	125
Figure 21. ACAKE phase of our Protocol 4.....	126

List of Tables



Table 1. Notations	18
Table 2. A survey of three-factor AAKA protocols	28
Table 3. Types of the proofs of relevant unsecure AAKA protocols	35
Table 4. Properties of relevant secure AAKA protocols	35
Table 5. Execution times of operations (in milliseconds)	152
Table 6. Estimated execution times on the user side in our Protocol 1	152
Table 7. Estimated execution times on the user side in our Protocol 2	152
Table 8. Computational costs of our Protocol 3 and relevant protocols	155
Table 9. Estimated execution times of our Protocol 4 (in milliseconds)	155
Table 10. Computational costs of the proposed protocols	155
Table 11. Properties of the proposed protocols	160
Table 12. Properties of our Protocol 1 and relevant secure protocols	160
Table 13. Computational costs of members executing the login and AKA phase	161
Table 14. Comparisons of our Protocol 2 and relevant AAKA protocols for TMIS ...	163
Table 15. Comparisons of our Protocol 3 and the relevant AKE protocols for IoT	164
Table 16. Comparisons of our Protocol 4 and the relevant protocols	165
Table 17. PQC algorithms selected by the NIST	168



Chapter 1

Introduction

1.1 Research Motivation

A remote mutual authentication and key exchange (AKE) scheme enables a user to remotely log in to a server through an unreliable channel to mutually authenticate with each other and establish a secure session key, which can be used for secret communication for exchanging data over a public network. An AKE protocol is usually called authentication and key agreement (AKA) protocol when two parties both contribute to the establishment of the session key during authentication.

With the rapid development of the wireless networks, a lot of AKE/AKA protocols have been proposed. All of the existing AKE/AKA protocols are designed for either client-server or client-client authentication. We brought up the concept of compatible authentication and key exchange (CAKE) protocol, which provides cross-species authentication that any two valid entities in the system can authenticate with each other and create a secure session key through open networks. A CAKE protocol not only provides client-server authentication, but also client-client and server-server



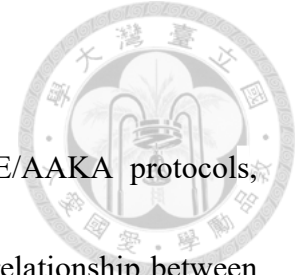
authentication. No CAKE protocol is proposed up to date.

AKE/AKA protocols designed for different purposes have different properties. In the following, we introduce the general properties of an AKE/AKA protocol: three-factor authentication, applicability for multi-server environments, user anonymity, user untraceability, independent authentication, perfect forward secrecy, member revocation, table free, public key management free, and formal security proof.

- (P1) **Three-factor authentication:** the first remote user authentication protocol was proposed by Lamport [1] in 1981 that users are verified by their username and the password. Since password authentication protocols need password tables and cannot withstand the replay attack that an intruder may replay a previously intercepted password, the first two-factor remote user authentication protocol was proposed by Hwang [2] in 1990 to overcome these problems. To enhance the security, many three-factor authentication protocols have been recently proposed. In general, two-factor authentication [3][4][5] adopts smartcard as the additional second factor to avoid replay attacks, and three-factor authentication usually employs biometric or physical unclonable function (PUF) as the additional third factor to avoid the smart-card-loss-attack.



- (P2) *Applicability of multi-server environments*: more and more commercial services nowadays are based on the multi-server architectures, in which mobile users remotely access resources distributed in several servers through open networks. Many AKE/AKA protocols have been developed for multi-server environments and low-power mobile devices, which the servers are regarded as independent entities that have distinct secret keys.
- (P3) *User anonymity*: traditional remote user authentication protocols only protect the user password and do not provide user privacy. For personal privacy, users want to access remote servers anonymously; hence, many anonymous AKE/AKA (AAKE/AAKA) protocols have been proposed to prevent the leakage of user's identity. In an AAKE/AAKA protocol, the real identity of a user would not be revealed to others when he/she logs into a server through a public channel. In some AAKE/AAKA protocol, which is designed for special purpose, even the server cannot get the user's real identity when the user logs into the server; this kind of property is called *strong user anonymity*. We do not focus on strong user anonymity in this dissertation, since it is only applicable to limited application circumstances.



- (P4) **User untraceability (unlinkability)**: in ordinary AAKE/AAKA protocols, even though a user uses an anonymous identity to login, the relationship between each login is exposed to third party since the user uses identical anonymous identity in each login. Some AAKE/AAKA protocols not only achieve user anonymity, but also achieve user untraceability (unlinkability), where a user cannot be traced by the login transmissions, i.e. no third party can derive the relation between any two login transmissions. Without loss of generality, to avoid the physical location tracking in an AKE/AKA protocol, users either can simply forge their location or use physical layer signatures [6] to simultaneously achieve privacy-preserving location authentication and user untraceability.
- (P5) **Perfect forward secrecy**: the session keys will not be compromised even if long-term secrets (private keys) used in the session key establishment are compromised. It is also called full forward secrecy.
- (P6) **Member revocation**: most AKE/AKA protocols did not deal with member revocation problem when a member is disabled or leaving. In comparison with traditional revocable AKE/AKA protocols using Certificate Revocation List (CRL), the advanced ones adopt time period to deal with revocation problem,



namely that a revoked or disabled member would not be able to get their newest secret key in the next time period.

- (P7) **Independent authentication**: a user and a server can independently authenticate with each other without the help of any third party.
- (P8) **Table free**: no table or list needs to be stored or maintained.
- (P9) **Public key announcement free**: no public key needs to be announced; either no public key or letting the identity be the public key.
- (P10) **Formal security proof**: it is formally proved that the security of the protocol is based on a well-known hard problem.

Many present authentication protocols are not secure on account of the lack of the formal proof and the poor design of the shared secrets. Once the sensitive secret has been leaked to the one who shouldn't get the secret, then some attacks may occur. We introduce some general security defects for a three-factor based AAKE/AAKA protocol in the following.

- (A1) **Replay attack**: This is a kind of man-in-the-middle attacks. The attacker maliciously repeats or delays a valid data transmission.
- (A2) **Privileged-insider attack – user/server impersonation attack**: A legal user or



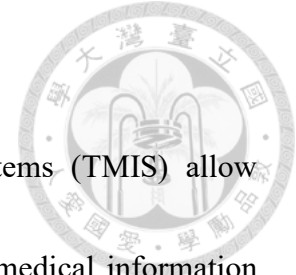
a legal server has ability to impersonate another user/server.

- (A3) *Smartcard-loss-attacks– offline password/identity guessing attacks*: When an attacker steals the smart card of a user, the attacker can offline guess the password/identity of the user.
- (A4) *Failure to forward secrecy*: The session keys will be compromised if long-term secrets used in the session key exchange are compromised.
- (A5) *Failure to user anonymity*: The real identity of an anonymous user would be disclosed to the third party.

With the growing need of user privacy preserving, a provably secure and efficient three-factor AAKE/AAKA protocol for multi-server environments is urgently required.

We will survey relevant protocols in Chapter 2 to show that no present AKE/AKA protocol has all the properties mentioned above (P1~P10). It inspires us to design a novel AKE/AKA protocol, which have most of the properties. We will propose an independent three-factor AAKA protocol for multi-server environments and a PUF based AKA protocol for IoT without verifiers and explicit CRPs in this dissertation.

The demand for telemedicine services grows rapidly with the rise of health consciousness, the development of Internet of Things (IoT), and the dramatic growth of



the world's older population. Telecare medical information systems (TMIS) allow patients remotely login medical service providers to acquire their medical information and track their health status through unsecured public networks. Hence, the privacy of patients is vulnerable to various types of security threats and attacks, such as the leakage of medical records or login footprints and the forgery attacks. In TMIS with single server, a patient usually communicates to the same medical service provider (server) through unreliable channels. In TMIS with multiple servers, a patient communicates to various servers through unreliable channels, and the various servers can be doctors, case managers, health centers, clinics, hospitals, etc. These servers should be regarded as independent entities with distinct private keys. Otherwise, the malicious server would masquerade as a patient or another medical server. Many three-factor AAKA protocols have been proposed for TMIS with single server, but none of them is applicable to TMIS with multiple servers. It inspires us to design a biometric-based three-factor AAKA protocol to protect user anonymity and untraceability in TMIS with multiple servers.

We bring up the concept of a compatible AKE (CAKE) protocol, which allows any two entities to mutually authenticate with each other through public channels, for



example, not only for client-server but also for client-client and server-server authentication. However, up to date, no CAKE protocol for a smart city is proposed; hence, we design a CAKE protocol for a smart city.

1.2 Objectives and Contribution

We review forty-nine three-factor based AAKE/AAKA protocols from 2013 to 2019 to show that none of them simultaneously achieves all the properties (P1~P10) introduced. According to the observation and discussion of the relevant protocols, we propose guidelines for designing a secure AKE/AKA protocol.

We obey the guidelines to design three novel three-factor AKA/AKE protocols for multi-server environments. One is for general situation, another is for TMIS, and the other is for IoT.

The benefits of the fifth generation (5G) of mobile technologies make the ideal of smart cities come true. We propose the first CAKE protocol for a smart city in 5G networks, which provides cross-species authentication and has all the properties (P1~P10). Any two valid entities can authenticate with each other and create a secure session key without the help of any third party while no password table and no public key management. The entity can be a natural person having biometric, an IoT device



embedded physical unclonable function (PUF), or a service provider. Moreover, we extend the CAKE protocol to propose an ACAKE protocol, which provides natural persons an anonymous option to protect their privacy. In addition, both the CAKE and ACAKE protocols also deal with entity revocation problem.

We construct an adversary model of three-factor AKE protocol with user anonymity in multi-server environments, and we give the formal security proofs of each proposed protocol in the random oracle model [7]. Our protocols are secure on mathematical assumptions: the elliptic curve decision Diffie-Hellman (ECDDH) problem, the elliptic curve computational Diffie-Hellman (ECCDH) problem, decisional bilinear Diffie-Hellman (DBDH) problem, and hash function assumptions.

We compare our protocols to relevant protocols to show the advantages and contributions and also show that our protocols are efficient enough for low-power mobile devices.

1.3 Dissertation Organization

We introduce the preliminaries in Chapter 2, survey the related work in Chapter 3, and construct an adversarial model for a three-factor (anonymous) AKE/AKA/CAKE protocol in multi-server environments in Chapter 4. The proposed independent



three-factor AKA protocol for multi-server environments (Protocol 1), privacy protection for TMIS with multiple servers using a biometric-based AKA Protocol (Protocol 2), PUF based AKA protocol for IoT without verifiers and explicit CRPs (Protocol 3), and CAKE/ACAKE protocol for a smart city in 5G networks (Protocol 4) are presented in Chapter 5, 6, 7, and 8, respectively. We analyze the performance of the proposed protocols and compare our protocols to relevant protocols in Chapter 9, and draw the conclusion and future work in Chapter 10.



Chapter 2

Preliminaries

The third factor in a three-factor authentication is usually an inherence factor, such as the biometric data of a user and the physical unclonable function (PUF) embedded in a physical device. However, the biometric data and the response of the PUF are noisy and non-uniformly distributed, which are not stable. A fuzzy extractor [8] is usually adopted to generate stable cryptographic keys with appropriate entropy from noisy and non-uniformly distributed data. In this chapter, we will introduce the PUF, the biometric, the fuzzy extractor, the elliptic curve cryptography, the bilinear pairing, the mathematical problems and assumptions, and the notations used in this dissertation.

2.1 Biometric

Biometric of a user can be the fingerprint, the iris scan, the face recognition, the voice, etc. We assume that there are trusted scan devices to capture and generate the biometrics of users.

2.2 Physical Unclonable Function

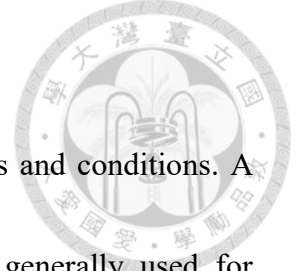
Physical unclonable function (PUF) is a physical entity embedded in a physical



device that provides a physically-defined output (response) for a given input (challenge), and the challenge and the corresponding response is called the Challenge Response Pair (CRP). Since the outputs of different physical entities for a same given input is nearly all distinct, the physically-defined output can be regards as a digital fingerprint that serves as a unique identifier for a physical device such as a microprocessor. PUFs provide alternate hardware fingerprints which can be used for lightweight authentication [9]. During semiconductor manufacturing, some unique physical variations would unintended occur to form a PUF. There are many implements of PUFs [10], and the most well-known ones are Coating PUFs, Silicon PUFs, Optical PUFs, and Acoustic PUFs. Silicon PUF has a sufficient amount of inter-chip variation to enable each IC to have unique output, such that the IC can be identified reliably and securely over a practical range of environmental variations [11]. Therefore, it can be adopted to uniquely identify each device in the IoT framework.

PUFs can be classified into strong PUFs and weak PUFs according to the number of possible CRPs [12]. A weak PUF can only support a small number of CRPs; hence, an attacker may get all the responses when he/she gets temporary accesses to the device.

Weak PUFs are typically used for key storage, since the responses are stable that a



challenge always yields the same response in various environments and conditions. A strong PUF can support a large number of CRPs; hence, it is generally used for authentication.

2.3 Fuzzy Extractor

A fuzzy extractor [8] or helper data algorithm can generate stable cryptographic keys with appropriate entropy from noisy and non-uniformly distributed biometric data and random PUF responses [13][14][15].

Definition 1 Let \mathcal{M} be a metric space on N points with distance function dis , and U_l denote the uniform distribution on l -bit binary strings. The statistical distance between two probability distributions A and B is $\text{SD}(A, B) = \frac{1}{2} \sum_v |\Pr(A = v) - \Pr(B = v)|$. A $(\mathcal{M}, \delta, l, d, \varepsilon)$ **fuzzy extractor** is a pair of procedures (Gen, Rep) [8][14] with the following properties:

- (1) Gen is a probabilistic generation procedure, which takes an input $w \in \mathcal{M}$, and outputs an extracted string $SS \in \{0, 1\}^l$ and a helper string $HLP \in \{0, 1\}^*$. For any distribution W on \mathcal{M} of min-entropy δ , the extracted string SS is nearly uniform even for those who observe the helper string HLP : if $(SS, HLP) \leftarrow \text{Gen}(W)$, then $\text{SD}((SS, HLP), (U_l, HLP)) \leq \varepsilon$.

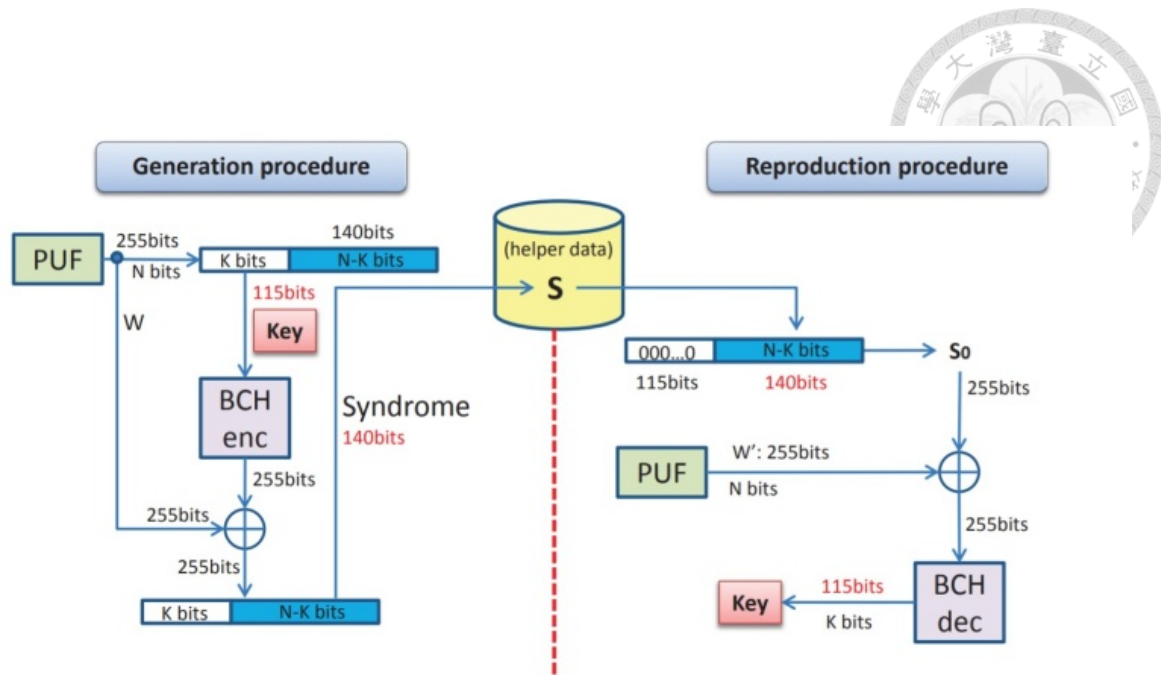


Figure 1. Implementation diagram for an efficient fuzzy extractor

(2) Rep is a deterministic reproduction procedure that can recover SS from the corresponding helper string HLP and any vector w' close to w . Namely, for all $w', w \in \mathcal{M}$ satisfying $dis(w', w) \leq d$, if $(SS, HLP) \leftarrow Gen(w)$, then $Rep(w', HLP) = SS$.

A fuzzy extractor is **efficient** if Gen and Rep run in expected polynomial time.

Using the Bose–Chaudhuri–Hocquenghem (BCH) codes and hash function can guarantee the elimination of noise from the collected noisy data and the uniform distribution of the derived key bits [8][13][14][15]. Figure 1 illustrates the implementation diagram for the efficient fuzzy extractor using the BCH code and syndrome concept ($N=255$), which is proposed by Kang et al. [14]. The helper data



along with a BCH decoder can be used to re-generate the correct response from the actual response of a PUF for a specific challenge.

2.4 Elliptic Curve Cryptography

Let p be a prime number, and let F_p denotes the field of integers modulo p . An elliptic curve E over F_p is defined by an equation of the form $y^2 = x^3 + ax + b$, where $a, b \in F_p$ satisfy $4a^3 + 27b^2 \neq 0 \pmod{p}$. A pair (x, y) , where $x, y \in F_p$, is a point on the curve if (x, y) satisfies $y^2 = x^3 + ax + b$. The set of all the points on E is denoted by $E(F_p)$. Let P be a point in $E(F_p)$, and suppose that P has prime order n . Then the cyclic subgroup of $E(F_p)$ generated by P is $G = \{\infty, P, 2P, 3P, \dots, (n-1)P\}$. [16]

2.5 Bilinear Pairing

Definition 2 Let E be an elliptic curve over a finite field F_q , which is a field of integers modulo a large prime number q , and $E(F_q)$ denotes the set of all the points on E . Let G_1 be an additive cyclic subgroup of points on $E(F_q)$, a point P be a generator of G_1 , and G_2 be a multiplicative group with the same order q . A **bilinear pairing** [17][18][19] is a map $\hat{e}: G_1 \times G_1 \rightarrow G_2$ which satisfies the following properties:

(1) *Bilinear*: $\hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}$ for all $a, b \in \mathbb{Z}_q^*$ and all $Q, R \in G_1$.

(2) *Non-degenerate*: there are two points $Q, R \in G_1$ such that $\hat{e}(Q, R) \neq 1$.



(3) *Computability*: there exists an efficient algorithm to compute $\hat{e}(Q, R)$ for all $Q, R \in G_1$.

2.6 Mathematical Problems and Assumptions

Given an elliptic curve E defined over a finite field F_q , four hard mathematical problems [16][17][18][19] that related to the dissertation are described as follows:

(1) ***Elliptic Curve Discrete logarithm (ECDL) Problem***: Given a point $Q = dP \in G_1$, to determine the integer d .

(2) ***Elliptic Curve Decision Diffie-Hellman (ECDDH) Problem***: Given $P, aP, bP, cP \in G_1$ for random $a, b, c \in Z_q^*$, to determine whether $cP = abP$. Note that DDH problem in bilinear pairing is easy: it is easy to verify if $\hat{e}(aP, bP) = \hat{e}(P, cP)$.

(3) ***Elliptic Curve Computational Diffie-Hellman (ECCDH) Problem***: Given $P, aP, bP \in G_1$ for random $a, b \in Z_q^*$, to find $abP \in G_1$.

(4) ***Decisional Bilinear Diffie-Hellman (DBDH) problem*** in $[G_1, G_2, \hat{e}]$: Given $P, aP, bP, cP \in G_1$ and $\hat{e}(P, P)^d \in G_2$ for random $a, b, c, d \in Z_q^*$, to determine whether $\hat{e}(P, P)^d = \hat{e}(P, P)^{abc}$.

The security of our protocols is based on the following assumptions:

Assumption 1 ECDL assumption: No probabilistic polynomial time algorithm can



solve ECDL problem with non-negligible advantage.

Assumption 2 ECDDH assumption: No probabilistic polynomial time algorithm can solve ECDDH problem with non-negligible advantage.

Assumption 3 ECCDH assumption: No probabilistic polynomial time algorithm can solve ECCDH problem with non-negligible advantage.

Assumption 4 DBDH assumption: No probabilistic polynomial time algorithm can solve DBDH problem with non-negligible advantage.

Assumption 5 Hash function assumption: There exists a secure one-way hash function $H: X \rightarrow Y$, which satisfies the following requirements:

- (1) *Preimage Resistance:* Given any $y \in Y$, it is hard to find $x \in X$ such that $H(x) = y$.
- (2) *Second Preimage Resistance:* Given any $x \in X$, it is hard to find $x' \in X$ such that $x' \neq x$ and $H(x') = H(x)$.
- (3) *Collision Resistance:* It is hard to find $x, x' \in X$ such that $x' \neq x$ and $H(x') = H(x)$.

2.7 Notations

We summarize the notations used in this dissertation in Table 1.



Table 1. Notations

Notation	Meaning
RC	trusted registration center
SCG	trusted security credential generator
DP	data provider
U_i / NP_i	i -th user/ i -th natural person
S_j	j -th server
D_l	l -th IoT device
ID_α	identity of the participant α
PW_i	password of U_i
B_i	biometric of U_i
PUF	physical unclonable function
SK	session key
K_α	private key of participant α
$K_{\alpha,t}$	secret key of member α in t -time-period
x / MSK	master secret key of RC
MRK	master private key
X / MPK	master public key of RC
t	index of time period
ΔT	maximum transmission delay
Gen/Rep	generation/ reproduction procedure of fuzzy extractor
SS_i / HLP_i	extracted/helper string of fuzzy extractor
C_A / R_A	challenge/ response string of PUF function
\hat{e}	a bilinear map on ECC
\parallel	string concatenation operation
\oplus	exclusive-or operation
$h()$	one way hash function mapping string to string
$H()$	one way hash function mapping string to point on ECC
$TG_{\hat{e}}$	time of executing a bilinear map on ECC
TG_{mul}	time of executing a multiplication of points on ECC
TG_{add}	time of executing a addition of points on ECC
T_{exp}	time of executing a exponential operation
T_{inv}	time of executing an inversion of scalars
T_{mul}	time of executing an multiplication of scalars
T_h	time of executing a one-way hash function
T_{sym}	time of executing a symmetric encryption/decryption
T_C	time of executing a Chebyshev chaotic map operation
T_{kdf}	time of executing a one-way key derivation function



Chapter 3

Related Work

3.1 Three-Factor AAKA

We review relevant three-factor AAKA protocols in this subsection. Traditional authentication protocols are designed for single server environments; however, many commercial services nowadays are based on multi-server architectures. There are two kinds of authentication protocols designed for multi-server architecture.

- ***Multi-server environment***: there is more than one server in the system, and the servers are regarded as distinct entities that have distinct secret keys. An authentication protocol for multi-server environment may suffer a malicious server attack, i.e. any server can impersonate another server or a legal user, if it is not well-designed.
- ***Pseudo multi-server environment***: it either a user have to register for each server or servers share the identical secret keys in some authentication protocols for multi-server environment; hence, all of these protocols are vulnerable to a malicious server attack, i.e. any server can impersonate another server. We classify this kind of authentication protocols pseudo multi-server environment in this

paper.



3.1.1 Single Server Environments

In 2013, Khan and Kumari [20] proposed a biometrics-based remote user authentication protocol with user anonymity, and Chaturvedi et al. [21] modified two biometrics-based remote user authentication protocols [22][23] to achieve user anonymity. In 2014, Islam [24] proposed a dynamic identity-based three-factor password authentication protocol using extended chaotic maps. In 2015, Cao and Ge [25] modified An's [26] three-factor authentication protocol to enhance the security and achieve user anonymity. In 2017, Choi et al. [27] showed that Cao-Ge protocol [25] may suffer various attacks, such as DoS attacks, off-line password attacks, user impersonation attacks, ID guessing attacks, etc; they further proposed an improvement. In the same year, Park et al. [28] also pointed out that Cao-Ge protocol [25] is vulnerable to server impersonation and offline identity guessing attacks; they also proposed an improvement. In 2019, Zhao et al. [29] pointed out that Park et al.'s protocol [28] may suffer offline password guessing attacks, and does not achieve user untraceability and perfect forward secrecy; they proposed a security-enhanced protocol using the extended Chebyshev chaotic maps. In 2019, Reddy et al. [30] modified two



two-factor anonymous AKA protocols [31][32] to three-factor anonymous AKA protocols.

3.1.2 Pseudo Multi-Server Environments

Chuang and Chen [33] proposed a multi-server AAKA protocol based on trust computing using smart cards and biometrics in 2014. However, the secret key of each server is the same in Chuang-Chen protocol. Later on, Maitra and Giri [34] and Choi et al. [35] both demonstrated that Chuang-Chen protocol [33] cannot withstand user impersonation attacks, masquerade attacks, smart card attacks, and DoS attacks, and fails in perfect forward secrecy. They both proposed improvements, but Maitra-Giri protocol [34] does not provide user anonymity.

Mishra et al. [36] also showed that Chuang-Chen protocol [33] is vulnerable to impersonation attacks, server spoofing attacks, and DoS attacks; they further proposed an improvement. However, Wang et al. [37] found that Mishra et al.'s protocol [36] may suffer masquerade attacks, replay attacks, and denial-of-service attacks, and proposed an improvement in 2016. In 2018, Yang and Zheng [38] claimed that Wang et al.'s protocol [37] cannot withstand privileged insider attacks, user impersonation attacks, and server spoofing attacks; they also proposed an improvement.



In 2015, Lu et al. [39] showed that Mishra et al.'s protocol [36] may suffer replay attacks and the password change phase is incorrect, and Lu et al. [40] showed that Mishra et al.'s protocol [36] is vulnerable to user impersonation attacks and server spoofing attacks, and does not achieve perfect forward secrecy, and they both proposed improvements. Later on, Moon et al. [41] demonstrated that Lu et al.'s protocol [39] cannot withstand user impersonation attacks and outsider attacks, and proposed an improvement. However, Guo et al. [42] found that Moon et al.'s protocol [41] does not resist insider, server spoofing, user impersonation and guessing attacks, and proposed an improvement. In 2018, Chaudhry et al.'s [43] demonstrated that Lu et al. protocol [40] is susceptible to user impersonation attacks, and proposed an improvement. However, the users have to register for each distinct server in Lu et al.'s [40] and Chaudhry et al.'s protocols [43]; otherwise, any server can impersonate a user to login another server since the registration center shares the identical secret keys with distinct servers. Therefore, their protocols are not applicable to a real multi-server environment.

Also in 2015, Wazid et al. [44] proposed a biometric-based AKA protocol in cloud computing, in which the user has to register at the service provider for each distinct cloud server.



3.1.3 Multi-Server Environments

In 2015, Lin et al. [45] also demonstrated that Chuang-Chen protocol [33] is defenseless against servers spoofing attacks and fails to protect the session key and the user's anonymity, and proposed an improvement. Note that the servers in Lin et al.'s [45] protocols have distinct secret keys; the servers in Chuang-Chen protocol [33] share the same secret key.

In the same year, Amin and Biswas [46] proposed a bilinear pairing based AKA protocol for multi-server environments; they analyzed Hsieh and Leu's two-factor authentication protocol [47] and modified it to a three-factor authentication protocol. However, Chandrakar and Om [48] found that Amin-Biswas protocol [46] lacks user untraceability, and cannot withstand identity and password guessing attacks, user-server impersonation attacks, and privileged insider attacks; they further proposed an improvement in 2017. We [49] found that Chandrakar-Om protocol [48] may suffer from malignant server attacks; when a user logs into a server, the server may obtain the user's secrets to impersonate the user. Also in 2017, Chandrakar and Om [50] proposed another anonymous three-factor remote authentication protocol for multi-server environments using an elliptic curve cryptosystem (ECC). However, we [49] still found



that Chandrakar-Om protocol [50] may suffer insider attacks; any user can impersonate another user.

Also in 2015, Jiang et al. [51] and He and Wang [52] both proposed anonymous remote biometrics AKA protocols in a multi-server environment. Odelu et al. [53] showed that He-Wang protocol [52] is vulnerable to a known session specific temporary information attacks and impersonation attacks. Odelu et al. [53] proposed an enhanced protocol with strong anonymity, i.e. user untraceability. In Odelu et al.'s protocol [53], the registration center (Gateway node) needs to store and manage users' temporal identity table.

In 2016, Park and Park [54] indicated that Chang et al.'s [55] two-factor authentication protocol may suffer off-line password guessing attacks, and further proposed a three-factor authentication using ECC. However, the registration center (or Gateway node) has to manage and store user's temporal identity table in Park-Park protocol [54]. In the same year, Choi et al.'s [56] modified Yoon and Kim's biometric-based user authentication protocol for wireless sensor networks [57] to provide user anonymity. However, we [49] showed that Choi et al.'s protocol [56] does not achieve user anonymity, because every legal user can compromise another user's



identity. Also in 2016, Irshad et al. [58] proposed a multi-server AAKA based on chaotic map without engaging registration center; however, the servers have to store the public keys of all the users.

In 2017, Reddy et al. [59] proposed a mutually authenticated key agreement protocol for multi-server environments. Xu et al. [60] demonstrated that Reddy et al.'s protocol [59] does not achieve user untraceability and cannot withstand privileged insider attacks, and proposed an improvement in 2019.

In 2018, Qi et al. [61] proposed a biometrics-based authentication key exchange protocol for multi-server TMIS using ECC. However, how to public the public keys of servers is an issue in Qi et al.'s protocol. Later on, Ali and Pal [62] proposed a three-factor based authentication protocol in multi-server environments using ECC. However, we [49] indicated that Ali-Pal protocol [62] is susceptible to a malignant server attack that the server would get the user's secrets to impersonate the user, who has ever login a server.

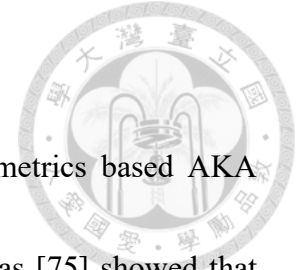
3.2 Three-Factor AAKA for TMIS

Many three-factor AAKA protocols have been proposed for TMIS with *single server*. In 2013, Das and Goswami [63] proposed a remote user authentication protocol



for connected health care with anonymity preserving. Later on, Wen [64] demonstrated that Das-Goswami protocol [63] has many security defects, such as user impersonation attacks, failure in forward security, and off-line password guessing attacks; they further proposed an improvement. In the next year, Xie et al. [65] pointed out that Wen's protocol [64] is insecure against the offline password guessing attacks and does not achieve user anonymity and forward security, and proposed an improvement. In 2015, Xu and Wu [66] found that Xie et al.'s protocol [65] does not resist de-synchronization attacks, and proposed an improvement.

In 2014, Tan [67] proposed a three-factor authentication protocol which provides user anonymity for TMIS with single server. Arshad and Nikooghadam [68] demonstrated that Tan's protocol [67] cannot withstand replay attacks, and proposed an improvement. In the next year, Das [69] and Lu et al. [70] simultaneously demonstrated that Arshad-Nikooghadam protocol [68] may suffer user impersonation attacks and off-line password guessing, and both proposed improvements. After then, Amin et al. [71], and Jiang et al. [72] showed that Lu et al.'s protocol [70] is insecure against user anonymity, new smart card issue, patient impersonation, and medical server impersonation attacks; they all proposed improvements.



In 2014, Mishra et al. [73] improved an un-anonymous biometrics based AKA protocol [74] to achieve user anonymity. In 2015, Amin and Biswas [75] showed that the Mishra et al.'s protocol [73] cannot withstand server impersonation, session key computation, and smart card stolen attacks, and proposed an improvement.

In 2016, Wazid et al. [76] showed that Amin et al.'s protocol [71] is vulnerable to privileged insider attacks through both smart card stolen and offline password guessing attacks, and also showed that Amin-Biswas protocol [75] is vulnerable to privileged-insider, stolen smart card, and offline password guessing, user impersonation as well as strong replay attacks; they further proposed an improvement. In the same year, Jiang et al. [77] proposed a three-factor authentication protocol with privacy preserving for e-Health clouds. However, Irshad and Chaudhry [78] identified a flaw in the mutual authentication phase of Jiang et al.'s protocol [77] that an adversary may launch a denial-of-service attack (DoS) against the server. In 2017, Zhang et al. [79] proposed a privacy protection for TMIS using a chaotic map-based three-factor authenticated key agreement protocol. In 2018, Wei et al. [80] demonstrated that Zhang et al.'s protocol [79] cannot withstand offline password/identity guessing attacks and user/server impersonation attacks, and proposed an improvement.



Table 2. A survey of three-factor AKA protocols

Single-server environment		Pseudo multi-server environment	
Protocol	Security defect	Protocol	Security defect
Generalized		Chuang-Chen [33]	(A2, A3, A4) [34][35][36][45]
Khan-Kumari [20]	-	Choi et al. [35]	-
Chaturvedi et al. [21]	-	Mishra et al. [36]	(A1, A2) [37][39][40]
Islam [24]	-	Wang et al. [37]	(A2, A3) [38]
Cao [25]	(A2, A3) [27][28]	Yang-Zheng [38]	-
Choi et al. [27]	-	Lu et al. [39]	(A2) [41]
Park et al. [28]	(A3, A4) [29]	Moon et al. [41]	(A2, A3) [42]
Zhao et al. [29]	-	Guo et al. [42]	-
Reddy et al. [30]	-	Lu et al. [40]	(A2) [43]
----- For TMIS		Chaudhry et al.[43]	-
Das-Goswami [63]	(A2, A3, A4) [64]	Wazid et al. [44]	-
Wen [64]	(A3, A4, A5) [65]	Multi-server environment	
Xie et al. [65]	(A1) [66]	Protocol	Security defect
Xu and Wu [66]	-	Lin et al. [45]	-
Tan [67]	(A1) [68]	Amin-Biswas [46]	(A2, A3) [48]
Arshad et al. [68]	(A2, A3) [69][70]	Chandrakar-Om[48]	(A2) [49]
Das [69]	-	Chandrakar-Om [50]	(A2) [49]
Lu et al.[70]	(A2, A5) [71] [72]	Jiang et al. [51]	-
Amin et al. [71]	(A2) [76]	He-Wang [52]	(A2) [53]
Jiang et al. [72]	-	Odelu et al. [53]	-
Mishra et al. [73]	(A2) [75]	Park-Park [54]	-
Amin-Biswas [75]	(A2) [76]	Choi et al. [56]	(A5) [49]
Wazid et al. [76]	-	Irshad et al. [58]	-
Jiang et al. [77]	-	Reddy et al. [59]	(A2) [60]
Zhang et al. [79]	(A2, A3) [80]	Xu et al. [60]	-
Wei et al. [80]	-	Qi et al. [61]	-
		Ali-Pal [62]	(A2) [49]

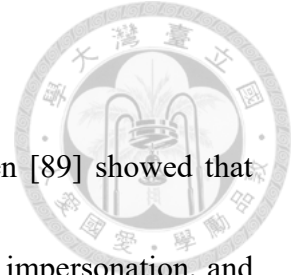


To the best of our knowledge, no three-factor AAKA protocol proposed for TMIS with *multiple servers*. We summarize the three-factor AAKE protocols, which are discussed in Section 2.1 and 2.2, in Table 2.

3.3 AKE for IoT

Internet of Things (IoT) is believed to play an important role of our daily life and is becoming a major part of smart home and smart city infrastructures in the near future. There are currently around 30 billion IoT connected devices, and will increase to around 75 billion connected devices [81] in 2025. They can communicate and interact with others over networks or worldwide internet and they can be remotely monitored and controlled. An authentication and key exchange (AKE) protocol for IoT framework provides IoT devices to authenticate mutually and establish a secure session key to communicate with each other through an open network.

Many PUF based AKE protocols have been proposed for the authentication between IoT device and server [82][83][84][85][86][87]. Two IoT devices need to authenticate mutually in some situations, such as wireless ad hoc network (WANET), mobile ad hoc network (MANET), vehicular ad hoc network (VANET). For the AKE between two IoT devices, Chatterjee et al. [88] proposed a PUF-based secure



communication protocol for IoT in 2017. In the next year, Braeken [89] showed that Chatterjee et al.'s protocol [88] is vulnerable to man-in-the-middle, impersonation, and replay attacks. Braeken proposed an alternative protocol [89] to solve these problems. However, in Braeken's protocol [89], servers need to be involved when two IoT devices want to authenticate each other. In 2019, Chatterjee et al. [13] proposed a PUF based AKE protocol for IoT without explicit CRPs in verifier database. Their protocol employs a hierarchical architecture that the verifier helps the subordinate IoT nodes to authenticate mutually and exchange their public keys; hence, the IoT nodes cannot establish a new session if they belong to different verifiers. Therefore, we propose a PUF based AKE protocol for IoT without verifier in this paper to resolve this issue; moreover, no explicit CRPs need to be maintained in our protocol. IoT nodes in the proposed protocol can freely authenticate mutually and establish a secure session key on their own without the help of any verifier.

3.4 AKE for a Smart City

Smart technologies play an important role in our daily life, such as smart cities, smart grids, smart homes, autonomous vehicles, drone, telemedicine, teleportation, remote educations, etc. With the development of smart technologies, the number of



Internet of Things (IoT) connected devices is currently around 30 billion, and will increase to around 75 billion in 2025 [81].

The fifth generation (5G) technology standard for cellular networks [90] provides high speed, high capacity, extremely low latency, and significant improvement on users' perceived quality of service (QoS), as compared to current 4G LTE networks. 5G networks are expected to be the panacea of the network issues of IoT [91][92][93]. 5G-Based IoT middleware systems have several security challenges [94], in which authentication and key exchange (AKE) protocol is the most significant security issue, since the entities communicate to each other in untrustworthy networks (i.e. the Internet). An AKE protocol allows two parties to authenticate mutually and create a secure common session key to communicate with each other.

To the best of our knowledge, no AKE protocol simultaneously achieves all the properties (P1~P10) mentioned in Section 1.1. Indeed, several AKE protocols for a smart city are proposed [95][96][97][98][99], and some existing anonymous AKE protocols [3][4][5][53][54][61][100][101] are applicable to IoT devices in a smart city. Most AKE protocols provide specific entities to remotely authenticate with each other, such as only for client-server or only for client-client authentication.

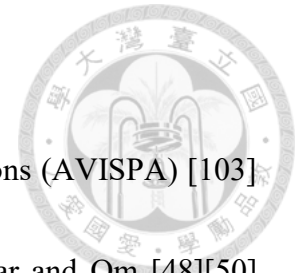


We propose the concept of a compatible AKE (CAKE) protocol that allows any two entities to mutually authenticate with each other through public channel, for example, not only for client-server but also for client-client and server-server authentication. However, up to date, no CAKE protocol for a smart city is proposed.

3.5 Design Guideline

We are going to discuss and analysis the relevant three-factor AAKA protocols [46][48][50][52][56][59][62][45][51][53][54][58][60][61], which are designed for multi-server environments and surveyed in Section 2.1. The comparisons of the unsecure ones [46][48][50][52][56][59][62] and secure ones [45][51][53][54][58][60][61] are listed in Table 3 and Table 4, respectively.

All the relevant unsecure protocols [46][48][50][52][56][59][62] do not have the proper formal proofs. Many present AKE protocols are not secure on account of the lack of formal proofs and the poor design of the shared secrets (G_1). Once the sensitive secret has been leaked to the one who shouldn't get the secret, then some attacks may occur. Choi et al. [56] did not provide the formal proof of their protocol. Amin and Biswas [46] and He and Wang [52] used Burrows–Abadi–Needham logic (BAN logic) [102] to prove the security of their protocol. Reddy et al.'s [59] used BAN logic and



Automated Validation of Internet Security Protocols and Applications (AVISPA) [103] simulation tool to prove the security of their protocols. Chandrakar and Om [48][50] and Ali-Pal [62] claimed that they gave formal security analysis using random oracle and the security of their protocols is based on BAN logic. Indeed, a random oracle is not a security model, and the security of their protocol is not based on existing difficult problems. A random oracle is just an oracle that responds to every query with a random response, which is uniformly chosen from its output domain, and it responds the same response for the same query (G2).

We summarized the types of the proofs of relevant unsecure protocols in a multi-server environment in Table 3. The results shown in Table 3 indicate that even though BAN logic [102] ensures the correctness of a protocol, it may fail to ensure the security of a protocol when the applied circumstance is out of the security assumptions (G3); even though AVISPA simulation tool [103] ensures that a protocol prevents outsider attacks, it cannot ensure that a protocol prevents insider attacks (G4).

The results shown in Table 3 also indicate that even though BAN logic [102] ensures the correctness of a protocol, it may fail to ensure the security of a protocol when the applied circumstance is out of the security assumptions (G3); even though



AVISPA simulation tool [103] ensures that a protocol prevents outsider attacks, it cannot ensure that a protocol prevents insider attacks. (G4).

The server can get the user's secret keys in Ali-Pal protocol [62]; hence, it is vulnerable to the insider (malignant server) attack in a multi-server environment. In Chandrakar-Om protocol 1 [48], RC verifies the validity of users by their A_i and these A_i are all identical; hence, their protocol is vulnerable to insider (malignant user) attacks (G5). In Chandrakar-Om protocol 2 [50], the server can compute the user's secret keys after the user login; hence, it may suffer insider (malignant server) attacks (G6). Hence, in order to avoid insider attacks, including malignant users and malignant servers, the validity of the participant cannot be verified by the identical parameter, and each participant cannot obtain any secret key of another participant.

In Choi et al.'s protocol [56], the secret $h(x||y)$ of each user are identical, and it is used to mask the real identity of the user; hence, their protocol does not achieve user anonymity. Thus, in order to achieve user anonymity, the user's real identity cannot be masked by a fixed value, which is identical to other user's value (G7).



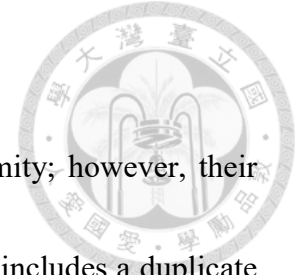
Table 3. Types of the proofs of relevant unsecure AAKA protocols

Protocol	Type of proof	Security defect
Amin-Biswas [46]	BAN	(A2, A3) [48]
Chandrakar-Om[48]	BAN+random oracle	(A2) [49]
Chandrakar-Om [50]	BAN+random oracle	(A2) [49]
He-Wang [52]	BAN	(A2) [53]
Choi et al. [56]	N/A	(A5) [49]
Reddy et al. [59]	BAN+AVISPA	(A2) [60]
Ali-Pal [62]	BAN+random oracle	(A2) [49]

Table 4. Properties of relevant secure AAKA protocols

	Lin [45]	Jiang et al. [51]	Odelu et al. [53]	Park- Park [54]	Irshad et al. [58]	Xu et al. [60]	Qi et al. [61]
(P3) User Anonymity	Y	Y	Y	Y	Y	Y	Y
(P4) User Untraceability	N	N	Y	Y	Y	Y	Y
(P7) Independent AKA	Y	N	N	N	Y	Y	N
(P8) Table free	N^{ab}	Y	N^a	N^a	Y	N^{ab}	Y
(P9) Public-key announcement free	Y	Y	Y	Y	N^{bc}	N^c	N^c

^aRC, ^busers, ^cservers.



Lin [45] and Jiang et al.'s [51] schemes achieve user anonymity; however, their schemes do not achieve user untraceability since the login message includes a duplicate value that the user may be traced by this duplicate value. Because the transmitted messages include duplicate parameter DID^{new} in adjacent sessions, Ali-Pal scheme [62] does not achieve user untraceability. Therefore, in order to achieve user untraceability, the transmitted messages can not include any duplicate parameter in different session; otherwise the user's login may be traced by the duplicate parameter (G8).

In Jiang et al. [51], Odelu et al. [53], Park-Park [54], and Qi et al.'s [61] protocol, RC has to be involved in each user login and authentication phase, i.e. dependent AKA (G9), it will add extra burden on RC and may cause the traffic bottleneck. In Lin [45] and Xu et al.'s [60] protocols, RC and users need to store key tables. In Odelu et al. [53] and Park-Park [54] protocols, RC needs to store the verification table about the users (G9). These protocols [45][53][54][60] are not table free. In Irshad et al. [58], Xu et al. [60], and Qi et al.'s [61] protocols, the user encrypts his/her identity by using the server's public key. How to ensure that the users would get the correct public keys of the servers becomes a problem. Either the users pre-store the public keys of each server (it is not table free) or the users make inquiries online (G9). Verifying



the authenticity of public keys is an issue. As shown in Table 4, none of the secure AKA schemes simultaneously achieve independent AKA, table free, and public key announcement free. (G9)

According to above observation and discussion, we propose the guidelines for designing a secure AKA protocol in the following.

- (G1) The security of the AKA protocol must be formally proved in a security model, which is also called formal model. The security of an AKA protocol must base on a well-known hard problem and mathematical assumptions that if there is an attacker, who can successfully attack the protocol, then the administrator can use the ability of the attacker breaking the security of the protocol to solve the well-known hard problem.
- (G2) A single random oracle is not a security model. A random oracle model is a security model, which uses a random oracle.
- (G3) BAN logic [102] can ensure the correctness of a protocol, it may fail to ensure the security of a protocol when the applied circumstance is out of the security assumptions
- (G4) AVISPA simulation tool [103] cannot ensure the security for insider attacks.



- (G5) To avoid insider (malignant server) attacks in multi-server environments, each server needs to have at least one distinct private key and cannot get any user's secret key.
- (G6) To avoid insider attacks, including malignant users and malignant servers, the validity of the participant cannot be verified by the duplicate parameter, and each participant cannot obtain any secret key of another participant.
- (G7) To achieve user anonymity, the user's real identity cannot be masked by the duplicate parameter.
- (G8) To achieve user untraceability, the transmitted messages can not include any duplicate parameter in different session; otherwise the user's login may be traced by the duplicate parameter.
- (G9) The following are three approaches that a user takes to let the server secretly get the user's identity to achieve user anonymity in an AAKE protocol.

Approach 1: The user encrypts his/her identity by using the server's public key

[58][60][61].

Approach 2: The user encrypts his/her identity by the pre-shared key between the

user and the server.



Approach 3: The third party, it usually be RC or helper, has to be involved in each

AKA phase [51][53][54].

A preferable design of an AAKE protocol in multi-server environments is adopting Approach 1 and letting the identity be the public key. Then the AAKE protocol would simultaneously achieve (P7) independent AKA, (P8) table free, and (P9) public-key announcement free.



Chapter 4

Security Model

In this chapter, we introduce threat assumptions, and construct an adversarial model for a three-factor AKE protocol with user anonymity in multi-server environments.

4.1 Threat Assumptions

Assume that an adversary might have following attack capabilities in an AKE protocol.

(AC1) Be an outsider or any one of the legitimate members [94].

(AC2) Eavesdrop, delete, replay, or modify any message transmitted over an unreliable channel [94].

(AC3) Offline try all the (identity, password) pairs within probabilistic polynomial time [104].

(AC4) Steal the smart card and analyze the instruction power consumption to extract the secret data from the smart card [94][105][106].

(AC5) Fake the biometric [107].



(AC6) Correctly predict the PUF's responses to arbitrary challenges with high probability [108].

(AC7) Individually get the (identity, password) pair, get the secret data in the smart card, and either fake the biometric or correctly predict the PUF's responses.

But breaking them all in polynomial time is not feasible.

4.2 Adversarial Model

We construct an adversarial model of a three-factor (anonymous) AKE/AKA/CAKE protocol with user anonymity in multi-server environments in this section. The adversarial model is defined in a random oracle model [7]. There are a trusted Registration Center (RC) and members in the adversarial model, and each member has a unique identity. Let Π_α^s denotes the s -th session of participant α .

Definition 3 Π_α^s and Π_β^t are said *partners* if α and β authenticate with each other and accept a common session key in α 's s -session and β 's t -session.

Let \mathcal{A} be a probabilistic polynomial time algorithm that simulates the behavior of an adversary, who can potentially control all the communications by asking queries described below, and the queries are responded by a Challenger \mathcal{B} (oracle).

- **Hash** (D): \mathcal{B} maintains a hash table, which is initially empty, to ensure the identical



responses and avoid the collision. If $Hash(D)$ has been asked before, then \mathcal{B} finds D in the hash table and returns the same response. Otherwise, \mathcal{B} generates a random value w , appends (D, w) to the hash table, and returns w .

- **Extract** (α): This query can be asked for entity α , who is not one of the existing member. \mathcal{B} executes the registration phase of the protocol and responds the corresponding results. It models the insider attack. (AC1)
- **Send** (Π_α^s, M): \mathcal{B} simulates the protocol and responds the corresponding results, which should be responded by α after α receiving the message M . It models the active attack. (AC1) (AC2)
- **Execute** (Π_α^s): \mathcal{B} returns the complete transcripts of an honest execution between entity α and its partner in its s -th session. It models the passive attack. (AC2)
- **Reveal** (Π_α^s): There are two kinds of reveal query.
 - **Reveal_{SK}** (Π_α^s): If entity α has accepted a session key, say SK , in its s -th session, then \mathcal{B} returns SK . Otherwise, \mathcal{B} returns “NULL”. This query models the known-session-key attack that an adversary cannot reveal other session keys when it compromises a session key. (AC1)
 - **Reveal_{ID}** (Π_α^s): \mathcal{B} returns the real identity of α . This query models the anonymity



attack that an adversary cannot obtain the identity of the target user when the identities of other users are revealed. (AC1)

- **Rot** (U_i , TYPE): \mathcal{B} returns the secret of U_i according to TYPE, where TYPE withstands the type of the factor that is used for authentication. This query can be asked for at most two factors, it models three-factor authentication. The goal is that \mathcal{A} cannot impersonate the user U_α even if \mathcal{A} gets any two of the three factors.

(AC1) (AC3) (AC4) (AC5) (AC6)

- **Corrupt** (Π_α^s): \mathcal{B} gives \mathcal{A} the complete long-term private keys of α , who is one of the existing member. This query models perfect forward secrecy that even if an adversary knows the private keys of the participant α , it cannot compute any previous session key of α . (AC1)

- **Test** (Π_α^s): \mathcal{A} can make a Test query only once at any time during the game. The following are two kinds of test query:

- **Test_{SK}** (Π_α^s): When this query is asked for the s -th session of member α , \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns the session key of α in the s -th session if $b=1$, and returns a random value if $b=0$.

- **Test_{ID}** (Π_α^s): When this query is asked for the s -th session of member α , \mathcal{B} flips



an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns the real identity of α if $b=1$, and returns a random value if $b=0$.

Definition 4 An oracle Π_α^s with its partner Π_β^t are said *fresh* if the common session key $SK \neq \text{NULL}$ between them is confirmed, both *Reveal* (Π_α^s) and *Reveal* (Π_β^t) queries have not been asked, and no *Corrupt* (α) or *Corrupt* (β) query is asked before *Send* (Π_α^s, M) or *Send* (Π_β^t, M) query.

\mathcal{A} can ask *Test* (Π_α^s) query only once at any time during the game, and may ask other queries during asking the *Test* (Π_α^s) query whenever Π_α^s is fresh. \mathcal{A} outputs its guess b' for the bit b in *Test* (Π_α^s) query eventually. The adversarial model is a random oracle model that oracles behave hash function as a true random function, which produces a random value for each new query.

Definition 5 The ability of \mathcal{A} distinguishing the session key or identity from a random value in *Test* query measures the advantage of \mathcal{A} attacking the protocol. Let *Succ* denotes the event that \mathcal{A} correctly guesses the unbiased bit b in the *Test* query, and let $\text{Adv}_\mathcal{P}(\mathcal{A}) = |2 \cdot \Pr(\text{Succ}) - 1|$ defines the *advantage* of \mathcal{A} attacking the protocol \mathcal{P} . Note that if \mathcal{A} guesses b totally in random, then $\Pr(\text{Succ})$ would approaches $1/2$ and $\text{Adv}_\mathcal{P}(\mathcal{A})$ is negligible.



Definition 6 An AKE/AKA/CAKE protocol achieves existential **perfect forward secrecy, secrecy of session key, unforgeability, and three-factor authentication** if no probabilistic polynomial time adversary \mathcal{A} has a non-negligible advantage in the following game played between \mathcal{A} and infinite set of oracles.

- 1) System is set up according to the initialization and registration phases of the protocol.
- 2) \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{SK}*.
- 3) No *Corrupt* (α) or *Reveal* (Π_α^S) query is asked before *Test_{SK}* (Π_α^S) query.
- 4) \mathcal{A} may ask at most two types of *Rot* query for the same entity α .
- 5) \mathcal{A} may ask other queries during asking the *Test_{SK}* (Π_α^S) query whenever Π_α^S is fresh, and eventually outputs its guess b' for the unbiased bit b in *Test_{SK}* (Π_α^S) query. The game is terminated.

Definition 7 An AAKE/AAKA/ACAKE protocol achieves existential **user anonymity** if no probabilistic polynomial time adversary \mathcal{A} has a non-negligible advantage in the following game played between \mathcal{A} and infinite set of oracles.

- 1) System is set up according to the initialization and registration phases of the



protocol.

2) \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash*,

Extract, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{ID}*.

3) No *Corrupt* (α) or *Reveal* (Π_α^S) query is asked before *Test_{ID}* (Π_α^S) query.

4) \mathcal{A} may ask at most two types of *Rot* query for the same entity α .

5) \mathcal{A} may ask other queries during asking the *Test_{ID}* (Π_α^S) query whenever Π_α^S is fresh, and eventually outputs its guess b' for the unbiased bit b in *Test_{ID}* (Π_α^S)

query. The game is terminated.



Chapter 5

An Independent Three-Factor AKA Protocol

with Privacy Preserving for Multi-Server

Environments

We will propose a biometric-based three-factor AAKA protocol for general multi-server environments (our Protocol 1) [109] in this chapter. This protocol is light weight, provably secure, and applicable to both single server and multi-server environments; meanwhile, it achieves user anonymity and user untraceability. We will describe the framework of an AAKA protocol in a multi-server environment first, and then expatiate on Protocol 1 and the characteristic analysis and the security analysis of Protocol 1. The security of Protocol 1 is based on the Elliptic Curve Computational Diffie-Hellman (ECCDH), the Decisional Bilinear Diffie-Hellman (DBDH), and hash function assumptions. In Section 9.1.1, we will show that the proposed Protocol 1 is efficient enough for low-power mobile devices.

5.1 The Framework of an AAKA Protocol in a Multi-server

Environment

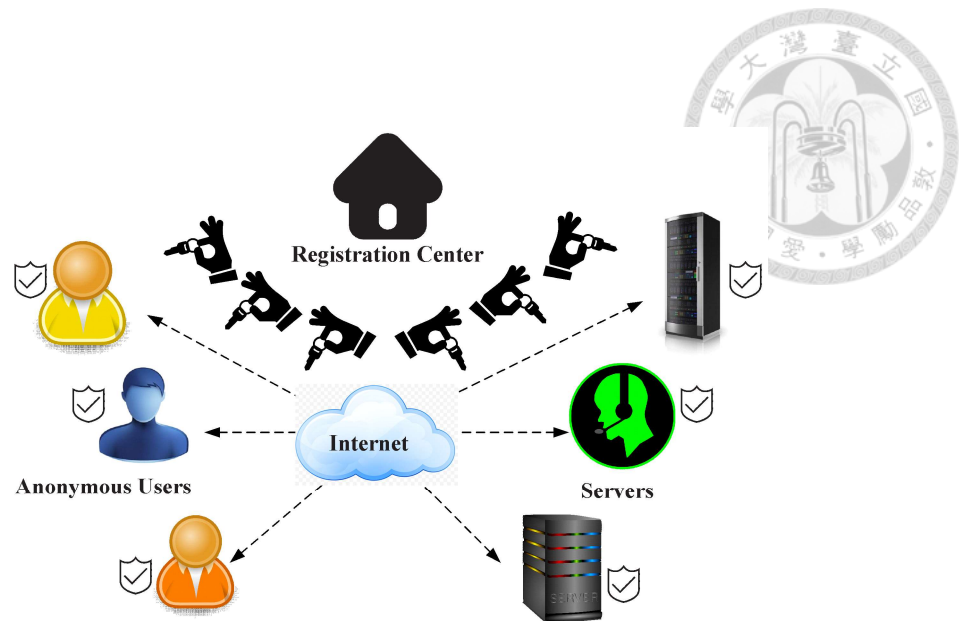


Figure 2. The framework of an AKA protocol in a multi-server environment

The system is established by RC in the initial. When a new user or a new server joins the system, he/she has to register on RC through a trusted channel, and RC will generate his/her private key and send it back to him/her through a trusted channel.

After the registration, a user can remotely log in to a server to authenticate mutually and establish a session key for the upcoming secure communications in public channels. The framework of an AKA protocol in a multi-server environment is illustrated in Figure 2.

5.2 The Proposed Protocol (Protocol 1: General AKA)

Our protocol consists of four phases: initialization, registration, login and AKA, and the password and biometric change. The details of these four phases are described in the following.



5.2.1 Initialization Phase

RC produces a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$, where G_1 is a subgroup of the additive group of points on an elliptic curve over a finite field $E(F_p)$ with order q , and G_2 is a multiplicative cyclic subgroup with the same order q over a finite field F_p . The general bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$ is operated under the Gap Diffie-Hellman group. The gap Diffie-Hellman (GDH) parameter generators, which satisfy the GDH assumptions, are believed to be constructed from the Weil and Tate pairings associated with super-singular elliptic curves or abelian varieties [19].

RC selects three secure one way collision-resistance hash functions $H: \{0, 1\}^* \rightarrow G_1$, $h_1: \{0, 1\}^* \rightarrow \{0, 1\}^l$ and $h_2: \{0, 1\}^* \rightarrow \{0, 1\}^l$, where l is an integer that stands for the length of the bit-string. RC selects a secret key x in Z_q^* , and keeps x as the master key. RC then decides the maximum transmission delay ΔT , and the generation function Gen and the reproduction function Rep of fuzzy extractor. RC chooses a generator point P in G_1 , and computes the public key $P_{pub} = x \cdot P$. Finally, RC publishes the system parameters $\mathcal{Pub} = \{G_1, G_2, P, \hat{e}, H, h_1, h_2, Gen, Rep, q, P_{pub}, \Delta T\}$.

5.2.2 Registration Phase

We are going to describe the server registration and user registration phases, and



they are illustrated in Figure 3 and Figure 4, respectively. All the messages are transmitted through trusted channels. Then, we describe the member revocation phase.

[Server registration phase]

When a new server S_j joins the system, the following steps are performed.

Step 1. S_j freely generates its identity ID_{S_j} , and then sends ID_{S_j} to RC.

Step 2. Upon receiving the identity ID_{S_j} from S_j , RC uses its master key x to generate

S_j 's secret key K_{S_j} , i.e. RC computes $K_{S_j} = x \cdot H(ID_{S_j})$, and then RC sends the secret key K_{S_j} and the system parameters Pub to S_j .

[User registration phase]

When a new user U_i with a portable device joins the system, he/she has to register with RC by executing the following steps.

Step 1. U_i freely decides his/her identity ID_{U_i} and his/her password PW_i . Note that ID_{U_i}

can be either the real identity of U_i or just a pseudonym, which enables U_i to

achieve strong anonymity. U_i imprints his/her biometric impression at the sensor

to get B_i and the portable device executes the generation function Gen on B_i to

generate the secret string SS_i and the helper string HLP_i , i.e. $(SS_i, HLP_i) = Gen(B_i)$.

U_i then computes the regenerated password $RPW_i = h(PW_i || SS_i)$ and sends $\langle ID_{U_i},$



RPW_i to RC.

Step 2. Upon receiving $\langle ID_{U_i}, RPW_i \rangle$ from U_i , RC uses its master key x to generate

U_i 's secret key K_{U_i} , i.e. $K_{U_i} = x \cdot H(ID_{U_i})$. RC then computes $C_i = H(ID_{U_i} || RPW_i)$

and the validation value $d_i = h_1(C_i)$, and encrypts K_{U_i} by C_i , i.e. $A_i = K_{U_i} + C_i$. RC

then stores $\{A_i, d_i, Pub\}$ into a new smart card, and sent the smart card to U_i in a

trusted environment.

Step 3. Upon receiving the smart card from RC, U_i inserts the smart card to the portable

device, and the device records the helper string HLP_i to the smart card.

Eventually, the smart card stores $\{HLP_i, A_i, d_i, Pub\}$.

[Member revocation phase]

When a legal user or a legal server wants to revoke the previous authorization or RC wants withdraw the authorization of a legal user or a legal server, RC adds the identity of the revoked member to the certificate revocation list (CRL) and issues the updated CRL to each member.

5.2.3 Login and AKA Phase

When a user U_i wants to log in to a server S_j , the following steps are performed.

All the messages are transmitted through untrusted channels. The Login and AKA phase



is illustrated in Figure 5.

Step 1. U_i inputs his/her identity ID_{U_i} and his/her password PW_i to the portable device, and imprints the biometric impression at the sensor to get B_i . The device produces the secret parameter SS_i^* by executing the reproduction function Rep on B_i and the helper string HLP_i , which is stored in the smart card, i.e. $SS_i^* = Rep(B_i, HLP_i)$. The device computes U_i 's regenerated password $RPW_i^* = h_1(PW_i || SS_i^*)$, $C_i^* = H(ID_{U_i} || RPW_i^*)$, and the validation value $d_i^* = h_1(C_i^*)$, and then checks whether d_i^* equals d_i or not. If $d_i^* = d_i$, the validity of U_i is confirmed and the device continues the procedure. Otherwise, it terminates the process.

Step 2. U_i 's device generates a random nonce n_U in Z_q^* and a timestamp T_U , and then computes $N_U = n_U \cdot P$, and derives the secret key $K_{U_i}^*$ by using A_i , which is stored in the smart card, and the computed C_i^* , i.e. $K_{U_i}^* = A_i - C_i^*$. The device computes $y_1 = \hat{e}(n_U \cdot P_{pub}, H(ID_{S_j}))$ and $y_2 = \hat{e}(K_{U_i}^*, H(ID_{S_j}))$, and mask U_i 's real identity ID_{U_i} by y_1 and the timestamp T_U , i.e. $AID = ID_{U_i} \oplus h_1(y_1 || T_U)$, and the validation value $v_U = h_1(ID_{U_i} || N_U || y_1 || y_2 || T_U)$. U_i 's device then transmits the login request $\langle ID_{S_j}, AID, N_U, T_U, v_U \rangle$ to S_j .

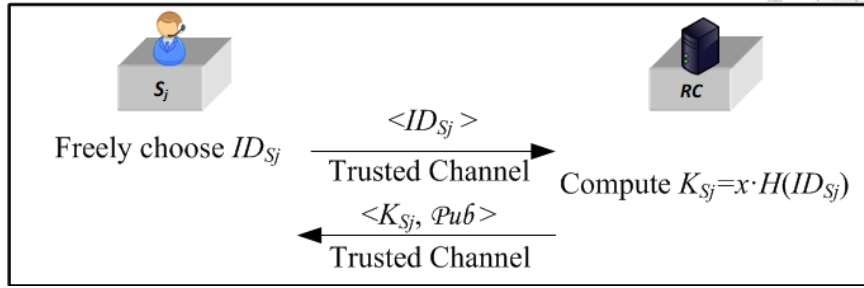


Figure 3. Server registration phase of our Protocol 1 (General AKA)

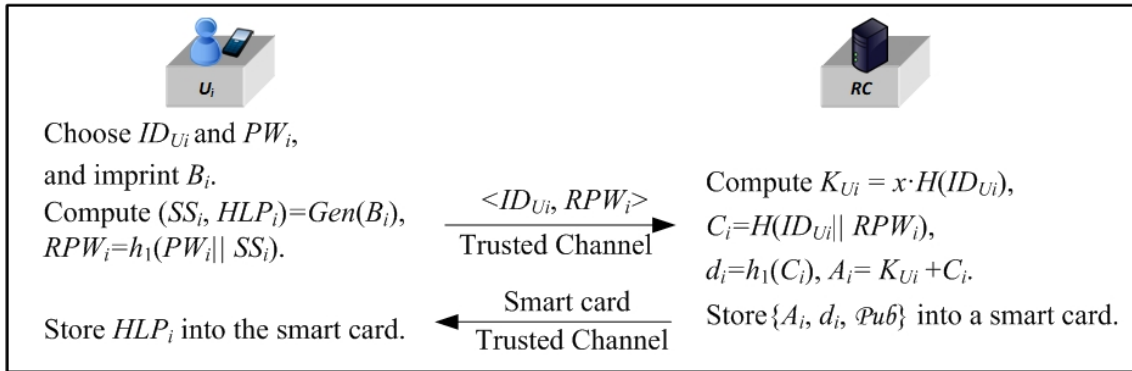


Figure 4. User registration phase of our Protocol 1 (General AKA)

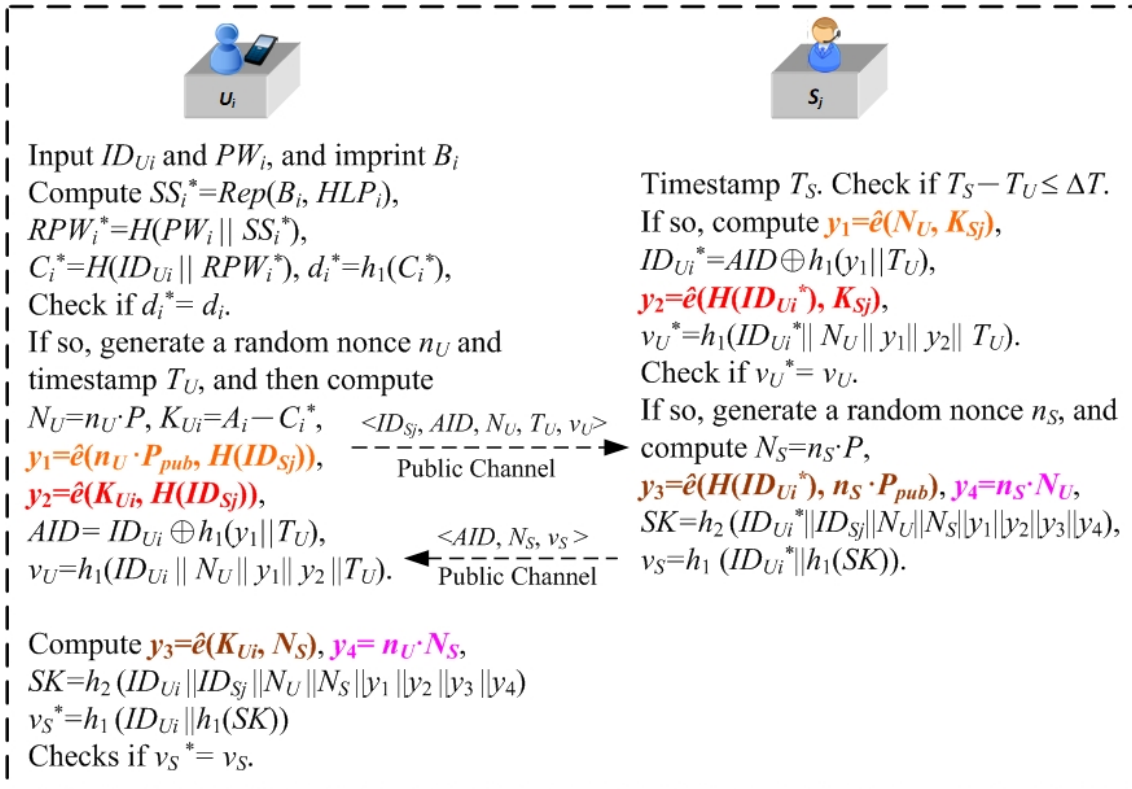


Figure 5. Login and AKA phase of our Protocol 1 (General AKA)



Step 3. When S_j receives the login request message from U_i , S_j generates a timestamp T_S , and verifies $(T_S - T_U)$. If $(T_S - T_U) > \Delta T$, S_j rejects the login request. Otherwise, S_j continues the process. S_j computes $y_1 = \hat{e}(N_U, K_{S_j})$ to decrypt U_i 's real identity $ID_{U_i}^*$ from AID , i.e. $ID_{U_i}^* = AID \oplus h_1(y_1 || T_U)$. S_j computes $y_2 = \hat{e}(H(ID_{U_i}^*), K_{S_j})$, which is the shared secret between U_i and S_j , and computes the validation value $v_U^* = h_1(ID_{U_i}^* || N_U || y_1 || y_2 || T_U)$. S_j then checks whether v_U^* equals v_U or not. If $v_U^* \neq v_U$, S_j rejects the login request; otherwise, S_j continues the process. S_j generates a random nonce $n_S \in Z_p^*$ and computes $N_S = n_S \cdot P$, $y_3 = \hat{e}(H(ID_{U_i}^*), n_S \cdot P_{pub})$, the co-contributed secret $y_4 = n_S \cdot N_U$, the session key $SK = h_2(ID_{U_i}^* || ID_{S_j} || N_U || N_S || y_1 || y_2 || y_3 || y_4)$, and the validation value $v_S = h_1(ID_{U_i}^* || h_1(SK))$. S_j then sends $\langle AID, N_S, v_S \rangle$ to U_i .

Step 4. After receiving $\langle AID, N_S, v_S \rangle$, U_i computes $y_3 = \hat{e}(K_{U_i}, N_S)$, the co-contributed secret $y_4 = n_U \cdot N_S$, the session key $SK = h_2(ID_{U_i} || ID_{S_j} || N_U || N_S || y_1 || y_2 || y_3 || y_4)$, and the validation value $v_S^* = h_1(ID_{U_i} || h_1(SK))$. U_i checks if $v_S^* = v_S$. If so, U_i adopts SK as the session key. Otherwise, U_i aborts it.

Correctness: The session keys computed by U_i and S_j are identical, since $y_1 =$

$$\hat{e}(n_U \cdot P_{pub}, H(ID_{S_j})) = \hat{e}(n_U \cdot xP, H(ID_{S_j})) = \hat{e}(n_U \cdot P, x \cdot H(ID_{S_j})) = \hat{e}(N_U, K_{S_j}), y_2 = \hat{e}(K_{U_i},$$



$$H(ID_{Sj}) = \hat{e}(x \cdot H(ID_{Ui}^*), H(ID_{Sj})) = \hat{e}(H(ID_{Ui}^*), x \cdot H(ID_{Sj})) = \hat{e}(H(ID_{Ui}^*), K_{Sj}), y_3 = \hat{e}(H(ID_{Ui}^*), n_S \cdot P_{pub}) = \hat{e}(H(ID_{Ui}^*), n_S \cdot xP) = \hat{e}(x \cdot H(ID_{Ui}^*), n_S \cdot P) = \hat{e}(K_{Ui}, N_S), \text{ and } y_4 = n_S \cdot N_U = n_S \cdot n_U \cdot P = n_U \cdot n_S \cdot P = n_U \cdot N_S.$$

5.2.4 Password and Biometric Change Phase

Users can change their password and biometric on their own by performing the following steps.

Step 1. It is the same as Step 1 in the login and AKA phase.

Step 2. U_i can change either one, or both of the password and the biometric.

Case 1. Password change only: U_i inputs the new password PW_i^{new} , and U_i 's device

computes $RPW_i^{new} = h(PW_i^{new} || SS_i^*)$, $C_i^{new} = H(ID_i || RPW_i^{new})$, $A_i^{new} = A_i - C_i^* + C_i^{new}$, and $d_i^{new} = h_1(C_i^{new})$. U_i 's device then replaces A_i and d_i with A_i^{new} and d_i^{new} , respectively.

Case 2. Biometric change only: U_i imprints new biometric impression B_i^{new} , and U_i 's

device executes the generation function Gen on B_i to generate the secret string SS_i and the helper string HLP_i^{new} , i.e. $(SS_i^{new}, HLP_i^{new}) = Gen(B_i^{new})$.

The device computes $RPW_i^{new} = h(PW_i || SS_i^{new})$, $C_i^{new} = H(ID_i || RPW_i^{new})$,

$A_i^{new} = A_i - C_i^* + C_i^{new}$, and $d_i^{new} = h_1(C_i^{new})$, and then replaces A_i , d_i , and HLP_i



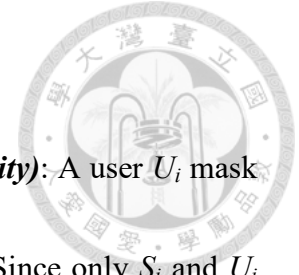
with A_i^{new} , d_i^{new} , and HLP_i^{new} , respectively.

Case 3. Change both password and biometric: U_i inputs new password PW_i^{new} , and imprints new biometric impression B_i^{new} . U_i 's device executes the generation function Gen on B_i to generate the secret string SS_i and the helper string HLP_i , i.e. $(SS_i^{new}, HLP_i^{new}) = Gen(B_i^{new})$. The device then computes $RPW_i^{new} = h(PW_i^{new} || SS_i^{new})$, $C_i^{new} = H(ID_i || RPW_i^{new})$, $A_i^{new} = A_i - C_i^* + C_i^{new}$, and $d_i^{new} = h_1(C_i^{new})$, and then replaces A_i , d_i , and HLP_i with A_i^{new} , d_i^{new} , and HLP_i^{new} , respectively.

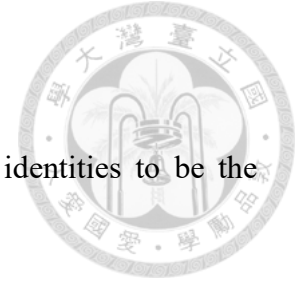
5.3 Characteristic Analysis

We analyze the properties of the proposed Protocol 1 point by point in the following.

- (P1) **Three-factor authentication**: We use biometric as the third factor in the authentication. That is, a server authenticates a user by three user's factors: user's password, the secrets stored in the smart card, and user's biometric. Thus the proposed Protocol 1 achieves three-factor authentication.
- (P2) **Applicability of multi-server environments**: Servers are regarded as independent entities and have distinct secret keys $K_{S_j} = x \cdot H(ID_{S_j})$ in Protocol 1, hence Protocol 1 is applicable to multi-server environments.



- (P3) **User anonymity** and (P4) **User untraceability (unlinkability)**: A user U_i masks its identity ID_i by $h_1(y_1 || T_U)$, where $y_1 = \hat{e}(n_U \cdot P_{pub}, H(ID_{S_j}))$. Since only S_j and U_i can compute y_1 , no third party can obtain y_1 to get U_i 's real identity ID_i , thus Protocol 1 achieves user anonymity. Since the n_U are different in each session, no third party can derive the relation between any two login transmissions. Moreover, we will formally prove that Protocol 1 achieves user anonymity on DBDH assumption in Theorem 3. Moreover, a user can log in to a server under a pseudonym to achieve strong anonymity if the user chooses the pseudonym to be his/her identity in the user registration phase.
- (P5) **Perfect forward secrecy**: We will prove that Protocol 1 achieves perfect forward secrecy on ECCDH assumption in Theorem 1.
- (P6) **Member revocation**: Protocol 1 does not deliberate on the member revocation problem, but it can use the certificate revocation list (CRL) to deal with the member revocation problem.
- (P7) **Independent authentication**: a user and a server in Protocol 1 can independently authenticate with each other without the help of any third party.
- (P8) **Table free**: no table needs to be stored or maintained in Protocol 1.



- (P9) **Public key announcement free:** Protocol 1 adopts the identities to be the public keys, hence no public key needs to be announced.
- (P10) **Formal security proof:** We will give the formal proofs of Protocol 1 in Section 5.4, and the security of Protocol 1 is based on the Elliptic Curve Computational Diffie-Hellman (ECCDH) and the Decisional Bilinear Diffie-Hellman (DBDH) problems.

5.4 Security Analysis

We analyze the security of the proposed protocol in the random oracle model [7] in this section. In the random oracle model, the hash function is assumed to be a true random function that produces a random value for each new query. The security of the proposed protocol is based on the Elliptic Curve Computational Diffie-Hellman (ECCDH), the Decisional Bilinear Diffie-Hellman (DBDH), and hash function assumptions. We give formal proofs of the proposed protocol in Theorem 1, Theorem 2, and Theorem 3 to demonstrate that the proposed protocol maintains session key secrecy and perfect forward secrecy, withstands user and server fogery attacks, and achieves user anonymity, respectively.



Theorem 1 *The proposed Protocol 1 maintains session key secrecy and perfect forward secrecy on the Elliptic Curve Computational Diffie-Hellman (ECCDH) assumption and hash function assumption.*

Proof: Suppose that \mathcal{A} is a probabilistic polynomial time adversary, who can break the session key secrecy or the perfect forward secrecy of Protocol 1 (general AAKA) with a non-negligible advantage ε , and given target user's identity ID_U and target server's identity ID_S . By using \mathcal{A} 's ability of breaking Protocol 1, we can construct an algorithm \mathcal{B} to solve the Elliptic Curve Computational Diffie-Hellman (ECCDH) problem with a non-negligible advantage. Let q_n denotes the number of sessions, and \mathcal{B} is given an instance $(G_1, G_2, P, \hat{e}, q, A = aP, B = bP)$ of the ECCDH problem. \mathcal{B} 's goal is to output abP . \mathcal{B} runs \mathcal{A} as a subroutine and simulates its attack circumstances. \mathcal{B} sets up the system by generating the private key x in Z_q^* and setting the public parameters $\mathcal{Pub} = \{G_1, G_2, P, \hat{e}, H, h_1, h_2, Gen, Rep, q, P_{pub}, \Delta T\}$, where $P_{pub} = x \cdot P$. \mathcal{B} permeates the ECCDH problem into the Send queries, which are asked by \mathcal{A} , in the l -th session. The probability of \mathcal{A} asking $Test_{SK}$ query in the l -th session is $1/q_n$.

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, \mathcal{A} may ask the following queries and obtain the corresponding



results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{SK}*, and no *Corrupt*

(α) or *Reveal* (Π_α^S) query is asked before *Test_{SK}* (Π_α^S) query.

In order to avoid the collision and ensure the identical responses of the hash queries, \mathcal{B} maintains three Hash lists L_H , L_{h_1} , and L_{h_2} , which are initially empty. \mathcal{B} simulates the oracle queries of \mathcal{A} as follows:

• **Hash:** The following are the three kinds of hash query in Protocol 1.

- **Hash** (H, D): When someone makes an H -query for D , \mathcal{B} returns W if (D, W, w)

$\in L_H$. Otherwise, \mathcal{B} randomly chooses $w \in Z_q^*$, computes $W = w \cdot P$, adds (D, W, w) to L_H , and returns W .

- **Hash** (h_1, D): When someone makes an h_1 -query for D , \mathcal{B} returns w if (D, w)

$\in L_{h_1}$; otherwise, \mathcal{B} randomly selects $w \in Z_q^*$, adds (D, w) to L_{h_1} , and returns w .

- **Hash** (h_2, D): When someone makes an h_2 -query for D , \mathcal{B} returns w if (D, w)

$\in L_{h_2}$. Otherwise, randomly chooses $w \in Z_q^*$, adds (D, w) to L_{h_2} , and returns w .

• **Extract:** The following are the two kinds of extract query in Protocol 1.

- **Extract_{user}** (ID_α, RPW_α): When someone asks a user Extract-query for ID_α , \mathcal{B}

simulates *Hash*(H, ID_α) query to get $H(ID_\alpha)$, simulates *Hash*($H, (ID_\alpha || RPW_\alpha)$)

query to get C_α , simulates *Hash*(h_1, C_α) query to get d_α , computes $K_\alpha = x \cdot H(ID_\alpha)$



and $A_\alpha = K_\alpha + C_\alpha$ and returns $\{A_\alpha, d_\alpha, Pub\}$.

- **Extract_{server}**(ID_β): When someone asks a server Extract-query for an identity ID_β ,

\mathcal{B} simulates $Hash(H, ID_\beta)$ query to get $H(ID_\beta)$, computes $K_\beta = x \cdot H(ID_\beta)$, and

returns $\{K_\beta, Pub\}$.

• **Send**: Assume that user instance Π_α^s and the server instance Π_β^t are partners.

There are the two kinds of send query in Protocol 1.

- **Send_{User}**(Π_α^s , Start): \mathcal{B} generates a timestamp T_α . If it is asked in the l -th session,

then \mathcal{B} lets $N_\alpha = A$; otherwise, \mathcal{B} chooses a random nonce n_α in Z_q^* , and computes

$N_\alpha = n_\alpha \cdot P$. \mathcal{B} simulates $Hash(H, ID_\alpha)$ query to get $H(ID_\alpha)$, and $Hash(H, ID_\beta)$

query to get $H(ID_\beta)$. \mathcal{B} computes $y_1 = \hat{e}(x \cdot N_\alpha, H(ID_\beta))$ and $y_2 = \hat{e}(x \cdot H(ID_\alpha),$

$H(ID_\beta))$. \mathcal{B} simulates $Hash(h_1, (y_1 || T_\alpha))$ query to get $h_1(y_1 || T_\alpha)$, and computes

$AID = ID_\alpha \oplus h_1(y_1 || T_\alpha)$. \mathcal{B} simulates $Hash(h_1, (ID_\alpha || N_\alpha || y_1 || y_2 || T_\alpha))$ query to get

v_U , and returns $\{ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha\}$.

- **Send_{Server}**(Π_β^t , $\langle ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha \rangle$): \mathcal{B} generates a timestamp T_β , and verifies

$(T_\beta - T_\alpha) \leq \Delta T$. If $(T_\beta - T_\alpha) > \Delta T$, \mathcal{B} returns “Reject” to \mathcal{A} . Otherwise, \mathcal{B}

continues the process. \mathcal{B} computes $y_1 = \hat{e}(N_\alpha, K_\beta)$, and simulates $Hash(h_1,$

$(y_1 || T_\alpha))$ query to get $h_1(y_1 || T_\alpha)$. \mathcal{B} computes $ID_\alpha^* = AID \oplus h_1(y_1 || T_\alpha)$, simulates



$Hash(H, ID_\alpha^*)$ query to get $H(ID_\alpha^*)$, and simulates $Hash(H, ID_\beta)$ query to get $H(ID_\beta)$. \mathcal{B} computes $y_2 = \hat{e}(H(ID_\alpha^*), x \cdot H(ID_\beta))$, and simulates $Hash(h_1, (ID_\alpha^* || N_\alpha || y_1 || y_2 || T_\alpha))$ query to get v_α^* . \mathcal{B} verifies v_α^* , returns “Reject” if $v_\alpha^* \neq v_\alpha$, and continues the process if $v_\alpha^* = v_\alpha$. If it is asked in the l -th session, \mathcal{B} lets $N_\beta = B$, randomly chooses v_β in $\{0, 1\}^l$, returns $\{AID, N_\beta, v_\beta\}$, computes $y_3 = \hat{e}((H(ID_\alpha^*), x \cdot N_\beta)$, and records $Tag = (ID_\alpha^* || ID_\beta || N_\alpha || N_\beta || y_1 || y_2 || y_3)$. Otherwise, \mathcal{B} chooses a random nonce n_β in Z_q^* , computes $N_\beta = n_\beta \cdot P$, $y_3 = \hat{e}(H(ID_\alpha^*), x \cdot N_\beta)$, and $y_4 = n_\beta \cdot N_\alpha$, simulates $Hash(h_2, (ID_\alpha^* || ID_\beta || N_\alpha || N_\beta || y_1 || y_2 || y_3 || y_4))$ query to get SK , simulates $Hash(h_1, SK)$ query to get $h_1(SK)$, simulates $Hash(h_1, (ID_\alpha^* || h_1(SK)))$ query to get v_β , and returns $\{AID, N_\beta, v_\beta\}$.

- **Execute** (U_α, S_β): When it is asked for (ID_α, ID_β) , \mathcal{B} runs the simulation of above Send queries: simulates $Send_{User}(\Pi_\alpha^S, Start)$ query to get $\{ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha\}$, and simulates $Send_{Server}(\Pi_\beta^t, \langle ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha \rangle)$ query to get $\{AID, N_\beta, v_\beta\}$. \mathcal{B} then returns the transmitted messages $\{ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha\}$ and $\{AID, N_\beta, v_\beta\}$.
- **Reveal**: The following are the two kinds of reveal query.
 - **Reveal_{SK}** (Π_α^S): \mathcal{B} runs the simulation of above Send queries for Π_α^S and returns SK if the instance Π_α^S has accepted the session, and responds a null value



otherwise.

- **Reveal_{ID}** (Π_α^S): \mathcal{B} returns ID_α .

• **Rot**: The following are the three types of Rot query. Note that at most two types of

Rot query can be asked for a user U_α .

- **Rot** (U_α, PW): \mathcal{B} returns PW_α .

- **Rot** (U_α, BI): \mathcal{B} returns B_α .

- **Rot** (U_α, SC): \mathcal{B} simulates **Extract_{user}** (ID_α, RPW_α) query to get $\{A_\alpha, d_\alpha, Pub\}$, and returns $\{A_\alpha, d_\alpha, Pub\}$.

• **Corrupt** (Π_α^S): When this query is asked for ID_α , then \mathcal{B} simulates **Extract** (ID_α, RPW_α) query to get $\{A_\alpha, d_\alpha, Pub\}$, and returns $\{PW_\alpha, B_\alpha, A_\alpha, d_\alpha, Pub\}$.

• **Test_{SK}**: The following are the two types of reveal query.

- **Test_{SK}** (Π_α^S): \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns the session key SK if $b = 1$, and returns a random value if $b = 0$.

- **Test_{ID}** (Π_α^S): \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns the real identity ID_α of α if $b = 1$, and returns a random value if $b = 0$.

\mathcal{A} may ask other queries during asking the **Test_{SK}** (Π_α^S) query whenever Π_α^S is fresh, and eventually outputs its guess b' for the unbiased bit b in **Test_{SK}** (Π_α^S) query. The game



is terminated.

If \mathcal{A} can break the session key secrecy or the perfect forward secrecy of Protocol 1, $y_4^* = abP$ should appear in the L_{h2} list. \mathcal{B} uses Tag to find $((Tag || y_4^*), SK)$ in the L_{h2} list to get y_4^* . \mathcal{B} then answers $abP = y_4^*$ to the ECCDH problem. The success probability of \mathcal{B} solving the ECCDH problem depends on the event that \mathcal{A} asking the $Test_{SK}$ query in the l -th session. The probability of \mathcal{A} asking the $Test_{SK}$ query in the l -th session is $1/q_n$ in above simulation. If \mathcal{A} correctly guesses the unbiased bit b in the $Test_{SK}$ query with a non-negligible advantage ε , \mathcal{B} solves the ECCDH problem with a non-negligible advantage ε/q_n . By Assumption 3, no probabilistic polynomial time algorithm can solve the ECCDH problem with a non-negligible advantage. It is a contradiction. Hence, no probabilistic polynomial time adversary can break the session key secrecy or the perfect forward secrecy of Protocol 1 with a non-negligible advantage. Therefore, Protocol 1 maintains session key secrecy and perfect forward secrecy by Definition 6. \square

Theorem 2 *The proposed Protocol 1 withstands user and server forgery attacks under the Decisional Bilinear Diffie-Hellman (DBDH) assumption and hash function assumption.*

Proof: Suppose that \mathcal{A} is a probabilistic polynomial time adversary, who can forge a



user or a server in Protocol 1 with a non-negligible advantage ε . Then we can construct an algorithm \mathcal{B} to solve the Decisional Bilinear Diffie-Hellman (DBDH) problem with a non-negligible advantage by using \mathcal{A} 's ability of breaking Protocol 1. \mathcal{B} is given an instance $(G_1, G_2, P, \hat{e}, q, A = aP, B = bP, C = cP, \hat{e}(P, P)^d)$ of the DBDH problem, and \mathcal{B} 's goal is to determine if $\hat{e}(P, P)^d = \hat{e}(P, P)^{abc}$. \mathcal{B} runs \mathcal{A} as a subroutine and simulates its attack circumstances. \mathcal{B} sets up the system by letting $P_{pub}=A$ and setting the public parameters $\mathcal{Pub} = \{G_1, G_2, P, \hat{e}, H, h_1, h_2, Gen, Rep, q, P_{pub}, \Delta T\}$. \mathcal{B} permeates the DBDH problem into the queries on user $U(ID_U)$ and server $S(ID_S)$, which are asked by \mathcal{A} in the l -th session. \mathcal{B} adds (ID_U, B, NULL) and (ID_S, C, NULL) into L_H list. Namely, \mathcal{B} lets $H(ID_U) = B$ and $H(ID_S) = C$. Let q_U and q_S denotes the numbers of users and servers, respectively. The probability of \mathcal{A} asking $Test_{SK}$ query in the l -th session for user $U(ID_U)$ and server $S(ID_S)$ is $1/q_U \cdot q_S$.

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, and the user instance Π_α^s and the server instance Π_β^t are partners. In order to avoid the collision and ensure the identical responses of the hash queries, \mathcal{B} maintains three Hash lists L_H, L_{h1} , and L_{h2} , which are initially empty. \mathcal{B} simulates the oracle queries of \mathcal{A} as follows:



The *Hash*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test* queries are identical to those queries in the proof of Theorem 1.

• **Extract**: There are two types of extract query in Protocol 1 as follows:

- **Extract_{user}** (ID_α, RPW_α): A user Extract-query can only be asked for $ID_\alpha \neq ID_U$. \mathcal{B}

simulates $Hash(H, ID_\alpha)$ query to get $H(ID_\alpha)$, finds $(ID_\alpha, H(ID_\alpha), w)$ in L_H list to

get w , and computes $K_\alpha = w \cdot A$. \mathcal{B} simulates $Hash(H, (ID_\alpha || RPW_\alpha))$ query to get

C_α , simulates $Hash(h_1, C_\alpha)$ query to get d_α , computes $A_\alpha = K_\alpha + C_\alpha$, and then

returns $\{A_\alpha, d_\alpha, Pub\}$.

- **Extract_{server}** (ID_β): When a server Extract-query is asked for $ID_\beta \neq ID_S$, \mathcal{B}

simulates $Hash(H, ID_\beta)$ query to get $H(ID_\beta)$. \mathcal{B} finds $(ID_\beta, H(ID_\beta), w)$ in L_H list

to get w , computes $K_\beta = w \cdot A$, and returns $\{K_\beta, Pub\}$.

• **Send**: There are two types of send query in Protocol 1 as follows:

- **Send_{User}** ($\Pi_\alpha^S, Start$): When a $Send_{User}$ query is asked for ID_α , whose partner is

ID_β , \mathcal{B} generates a timestamp T_α and a random nonce n_α in Z_q^* , and computes

$N_\alpha = n_\alpha \cdot P$.

1) If $ID_\alpha \neq ID_U$, \mathcal{B} simulates $Hash(H, ID_\alpha)$ query to get $H(ID_\alpha)$, finds $(ID_\alpha,$

$H(ID_\alpha), w_\alpha)$ in L_H list to get w_α , and computes $y_2 = \hat{e}(w_\alpha \cdot A, H(ID_\beta))$.



2) If $ID_\alpha = ID_U$ and $ID_\beta = ID_S$, \mathcal{B} lets $y_2 = \hat{e}(P, P)^d$.

3) If $ID_\alpha = ID_U$ and $ID_\beta \neq ID_S$, \mathcal{B} simulates $Hash(H, ID_\beta)$ query to get $H(ID_\beta)$,

finds $(ID_\beta, H(ID_\beta), w_\beta)$ in L_H list to get w_β , and computes $y_2 = \hat{e}(H(ID_\alpha),$

$w_\beta \cdot A)$.

Then, \mathcal{B} computes $y_1 = \hat{e}(n_\alpha \cdot A, H(ID_\beta))$, simulates $Hash(h_1, (y_1 || T_\alpha))$ query to

get $h_1(y_1 || T_\alpha)$, and computes $AID = ID_\alpha \oplus h_1(y_1 || T_\alpha)$. \mathcal{B} simulates $Hash(h_1,$

$(ID_\alpha || N_\alpha || y_1 || y_2 || T_\alpha)$ query to get v_U , and returns $\{ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha\}$.

- **Send_{Server}** ($\Pi_\beta^t, \langle ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha \rangle$): When a $Send_{Server}$ query is asked for ID_β ,

\mathcal{B} generates a timestamp T_β , and verifies $(T_\beta - T_\alpha)$. If $(T_\beta - T_\alpha) > \Delta T$, then \mathcal{B}

returns “Reject” to \mathcal{A} ; otherwise, \mathcal{B} continues the process.

1) If $ID_\beta \neq ID_S$, \mathcal{B} simulates $Hash(H, ID_\beta)$ query to get $H(ID_\beta)$, finds $(ID_\beta,$

$H(ID_\beta), w_\beta)$ in L_H list to get w_β , computes $K_\beta = w_\beta \cdot A$ and $y_1 = \hat{e}(N_\alpha, K_\beta)$,

simulates $Hash(h_1, (y_1 || T_\alpha))$ query to get $h_1(y_1 || T_\alpha)$, computes $ID_\alpha^* = AID \oplus$

$h_1(y_1 || T_\alpha)$, simulates $Hash(H, ID_\alpha^*)$ query to get $H(ID_\alpha^*)$, and computes $y_2 =$

$\hat{e}(H(ID_\alpha^*), K_\beta)$.

2) If $ID_\beta = ID_S$, \mathcal{B} uses v_α to find $(ID_\alpha^* || N_\alpha || y_1 || y_2 || T_\alpha)$ in L_{h1} list to get ID_α^*

and y_1 . If $ID_\beta = ID_S$ and $ID_\alpha^* = ID_U$, \mathcal{B} lets $y_2 = \hat{e}(P, P)^d$. If $ID_\beta = ID_S$ and



$ID_\alpha^* \neq ID_U$, \mathcal{B} simulates $Hash(H, ID_\alpha^*)$ query to get $H(ID_\alpha^*)$, simulates $Hash(H, ID_\beta)$ query to get $H(ID_\beta)$, finds $(ID_\alpha^*, H(ID_\alpha^*), w_\alpha)$ in L_H list to get w_α , and computes $K_\alpha = w_\alpha \cdot A$ and $y_2 = \hat{e}(K_\alpha, H(ID_\beta))$.

\mathcal{B} randomly chooses n_β in Z_q^* , and computes $N_\beta = n_\beta \cdot P$, $y_3 = \hat{e}(H(ID_\alpha^*), n_\beta \cdot A)$, and $y_4 = n_\beta \cdot N_\alpha$. \mathcal{B} simulates $Hash(h_2, (ID_\alpha^* || ID_\beta || N_\alpha || N_\beta || y_1 || y_2 || y_3 || y_4))$ query to get SK , and $Hash(h_1, SK)$ query to get $h_1(SK)$, and $Hash(h_1, (ID_\alpha^* || h_1(SK)))$ query to get v_β . \mathcal{B} returns $\{AID, N_\beta, v_\beta\}$.

If \mathcal{A} answers $b = 1$ to the $Test_{SK}$ query, \mathcal{B} answers $\hat{e}(P, P)^d = \hat{e}(P, P)^{abc}$ to the DBDH problem. If \mathcal{A} answers $b \neq 1$ to the $Test_{SK}$ query, \mathcal{B} answers $\hat{e}(P, P)^d \neq \hat{e}(P, P)^{abc}$ to the DBDH problem. The success probability of \mathcal{B} solving the DBDH problem depends on the event that \mathcal{A} asks the $Send_{AKA}(\Pi_\alpha^S, \text{Start})$ query for user $U (ID_U)$ to forge the server $S (ID_S)$, and asks the $Test_{SK}$ query in this session; or the event that \mathcal{A} asks the $Send_{AKA}(\Pi_\beta^t, \langle ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha \rangle)$ query for server $S (ID_S)$ to forge the user $U (ID_U)$, and asks the $Test_{SK}$ query in this session. The probability of \mathcal{A} asking the $Send_{AKA}(\Pi_\alpha^S, \text{Start})$ query for user $U (ID_U)$ or asking the $Send_{AKA}(\Pi_\beta^t, \langle ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha \rangle)$ query for server $S (ID_S)$, and asking the $Test_{SK}$ query in this session is $1/q_U \cdot q_S$ in the above simulation. If \mathcal{A} correctly guesses b in the $Test_{SK}$ query with a non-negligible advantage



ε , \mathcal{B} solves the DBDH problem with a non-negligible advantage $\varepsilon/q_U \cdot q_S$.

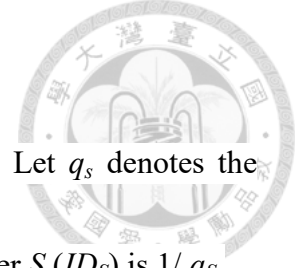
By Assumption 4, no probabilistic polynomial time algorithm can solve the DBDH problem with non-negligible advantage. It is a contradiction. Hence, no a probabilistic polynomial time adversary can forge a user or a server in Protocol 1 with a non-negligible advantage. Thus, Protocol 1 withstands user and server forgery attacks by

Definition 6. \square

Theorem 3 *The proposed Protocol 1 achieves user anonymity on the Decisional Bilinear Diffie-Hellman (DBDH) assumption and hash function assumption.*

Proof: Suppose that \mathcal{A} is a probabilistic polynomial time adversary, who can break the user anonymity of Protocol 1 (general AAKA) with a non-negligible advantage ε . By using \mathcal{A} 's ability of breaking Protocol 1, we can construct an algorithm \mathcal{B} to solve the Decisional Bilinear Diffie-Hellman (DBDH) problem with a non-negligible advantage.

\mathcal{B} is given an instance $(G_1, G_2, P, \hat{e}, q, A=aP, B=bP, C=cP, \hat{e}(P, P)^d)$ of the DBDH problem, and \mathcal{B} 's goal is to determine if $\hat{e}(P, P)^d = \hat{e}(P, P)^{abc}$. \mathcal{B} runs \mathcal{A} as a subroutine and simulates its attack circumstances. \mathcal{B} sets up the system by letting $P_{pub}=A$ and setting the public parameters $\mathcal{Pub} = \{G_1, G_2, P, \hat{e}, H, h_1, h_2, Gen, Rep, q, P_{pub}, \Delta T\}$. \mathcal{B} permeates the DBDH problem into Send queries of server $S (ID_S)$, which are asked by \mathcal{A} .



\mathcal{B} adds (ID_S, B, NULL) into L_H list. Namely, \mathcal{B} lets $H(ID_S) = B$. Let q_s denotes the number of servers. The probability of \mathcal{A} asking $Test_{SK}$ query for server $S (ID_S)$ is $1/q_s$.

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{ID}*, and no *Corrupt* (α) or *Reveal* (Π_α^S) query is asked before *Test_{ID}* (Π_α^S) query.

In order to avoid the collision and ensure the identical responses of the hash queries, \mathcal{B} maintains three hash lists L_H , L_{h1} , and L_{h2} , which are initially empty. \mathcal{B} simulates the oracle queries of \mathcal{A} as follows:

The *Hash*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test* queries are identical to those queries in the proof of Theorem 1. The *Extract* query is identical to the *Extract* query in the proof of Theorem 2.

• **Send**: There are two types of send query in Protocol 1 as follows:

- **Send_{User}** (Π_α^S , Start): When \mathcal{A} asks a user Send-query for ID_α , whose partner is ID_β , \mathcal{B} generates a timestamp T_α and a random nonce n_α in Z_q^* , simulates *Hash*(H, ID_β) query to get $H(ID_\beta)$, simulates *Hash*(H, ID_α) query to get $H(ID_\alpha)$, finds $(ID_\alpha, H(ID_\alpha), r_\alpha)$ in L_H list to get r_α , and computes $y_2 = \hat{e}(r_\alpha \cdot A, H(ID_\beta))$.



1) If $ID_\beta = ID_S$, \mathcal{B} computes $N_\alpha = n_\alpha \cdot C$ and $y_1 = (\hat{e}(P, P)^d)^{n_\alpha}$.

2) If $ID_\beta \neq ID_S$, \mathcal{B} computes $N_\alpha = n_\alpha \cdot P$ and $y_1 = \hat{e}(n_\alpha \cdot A, H(ID_\beta))$.

\mathcal{B} then simulates $Hash(h_1, (y_1 || T_\alpha))$ query to get $h_1(y_1 || T_\alpha)$, and computes AID

$= ID_\alpha \oplus h_1(y_1 || T_\alpha)$. \mathcal{B} simulates $Hash(h_1, (ID_\alpha || N_\alpha || y_1 || y_2 || T_\alpha))$ query to get v_α ,

and returns $\{ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha\}$.

- **Send_{server}** ($\Pi_\beta^\dagger, \langle ID_\beta, AID, N_\alpha, T_\alpha, v_\alpha \rangle$): When \mathcal{A} asks a user Send-query for ID_β ,

whose partner is ID_α , \mathcal{B} generates a timestamp T_β and verifies $(T_\beta - T_\alpha)$. If $(T_\beta$

$- T_\alpha) > \Delta T$, \mathcal{B} returns “Reject”. Otherwise, \mathcal{B} continues the process.

1) If $ID_\beta = ID_S$, \mathcal{B} uses v_α to find $(ID_\alpha || N_\alpha || y_1 || y_2 || T_\alpha)$ in L_{h_1} to get ID_α and y_1 .

2) If $ID_\beta \neq ID_S$, \mathcal{B} simulates $Hash(H, ID_\beta)$ query to get $H(ID_\beta)$, finds $(ID_\beta,$

$H(ID_\beta), r_\beta)$ in L_H to get r_β , computes $K_\beta = r_\beta \cdot A$ and $y_1 = \hat{e}(N_\alpha, K_\beta)$, simulates

$Hash(h_1, (y_1 || T_\alpha))$ query to get $h_1(y_1 || T_\alpha)$, and computes $ID_\alpha = AID \oplus h_1(y_1 || T_\alpha)$.

\mathcal{B} then simulates $Hash(H, ID_\alpha)$ query to get $H(ID_\alpha)$ to get $H(ID_\alpha)$, finds $(ID_\alpha,$

$H(ID_\alpha), r_\alpha)$ in L_H to get r_α , and computes $K_\alpha = r_\alpha \cdot A$ and $y_2 = \hat{e}(K_\alpha, H(ID_\beta))$. \mathcal{B}

randomly chooses n_β in Z_q^* , and computes $N_\beta = n_\beta \cdot P$, $y_3 = \hat{e}(H(ID_\alpha), n_\beta \cdot A)$, and

$y_4 = n_\beta \cdot N_\alpha$. \mathcal{B} simulates $Hash(h_2, (ID_\alpha || ID_\beta || N_\alpha || N_\beta || y_1 || y_2 || y_3 || y_4))$ query to get

SK , and $Hash(h_1, SK)$ query to get $h_1(SK)$, and $Hash(h_1, (ID_\alpha || h_1(SK)))$ query to



get v_β . \mathcal{B} returns $\{AID, N_\beta, v_\beta\}$.

If \mathcal{A} answers $b = 1$ to the $Test_{ID}$ query, \mathcal{B} answers $\hat{e}(P, P)^d = \hat{e}(P, P)^{abc}$ to the DBDH problem. If \mathcal{A} answers $b \neq 1$ to the $Test_{ID}$ query, \mathcal{B} answers $\hat{e}(P, P)^d \neq \hat{e}(P, P)^{abc}$ to the DBDH problem. The success probability of \mathcal{B} depends on the event that \mathcal{A} asks the $Send_{User}(\Pi_\alpha^S, \text{Start})$ query for server $S (ID_S)$, and asks the $Test_{ID}$ query for this session. In the above simulation, the probability that \mathcal{A} asking the $Send_{AKA}(\Pi_\alpha^S, \text{Start})$ query for server $S (ID_S)$ and asking the $Test_{SK}$ query for this session is $1/q_S$. If \mathcal{A} correctly guesses b in the $Test_{ID}$ query with a non-negligible advantage ε , \mathcal{B} can solve the DBDH problem with a non-negligible advantage ε/q_S . By Assumption 4, no probabilistic polynomial algorithm can solve DBDH problem with non-negligible advantage. It is a contradiction.

Thus, Protocol 1 withstands user anonymity attacks by Definition 7. \square



Chapter 6

Privacy Protection for TMIS with Multiple Servers Using a Biometric-based AKA Protocol

Telecare medical information systems (TMIS) allow patients remotely log in to medical service providers to acquire their medical information and track their health status through unsecured public networks. Hence, the privacy of patients is vulnerable to various types of security threats and attacks, such as the leakage of medical records or login footprints and the forgery attacks. Many three-factor based anonymous authentication and key agreement (AAKA) protocols have been proposed for TMIS with single server, but none of them is applicable to TMIS with multiple servers.

In this chapter, we will propose a biometric-based three-factor AAKA protocol in TMIS with multiple servers (our Protocol 2) [101], and we will give a formal security proof of the proposed protocol. The security of the proposed protocol is based on the elliptic curve decisional Diffie-Hellman (ECDDH) problem assumption and hash function assumption. We will show that the proposed Protocol 2 is efficient enough for low-power mobile devices in Section 9.1.2.



6.1 The Framework of an AAKA Protocol in TMIS

In a TMIS with multiple servers, there are one trusted registration center (RC), various medical service providers (Servers), and numerous patients (Users). RC is in charge of system initialization, the registration affairs, and keeping the master key of the system. Servers may be doctors, case managers, clinics, hospitals, health centers, and so on. Figure 6 illustrates the framework of an AAKA protocol in TMIS. To protect the privacy of users, servers are regarded as independent entities having distinct private keys. Any server cannot compromise the secrecy of the session between a user and another server. Each user has a low-power mobile device to communicate to RC and servers. Initially, RC establishes the system. Every server and user must register with RC through a trustworthy channel when join the system, and RC would generate the private key of the server/user and send it back through a trustworthy channel. After registration, each user makes online update through a public channel to get the necessary information before he/she logs in to an unfamiliar server. Then, a user can use his/her private key and the necessary information to remotely log in to a server, mutually authenticate with the server, and establish a session key for the upcoming secure communications in public channels.

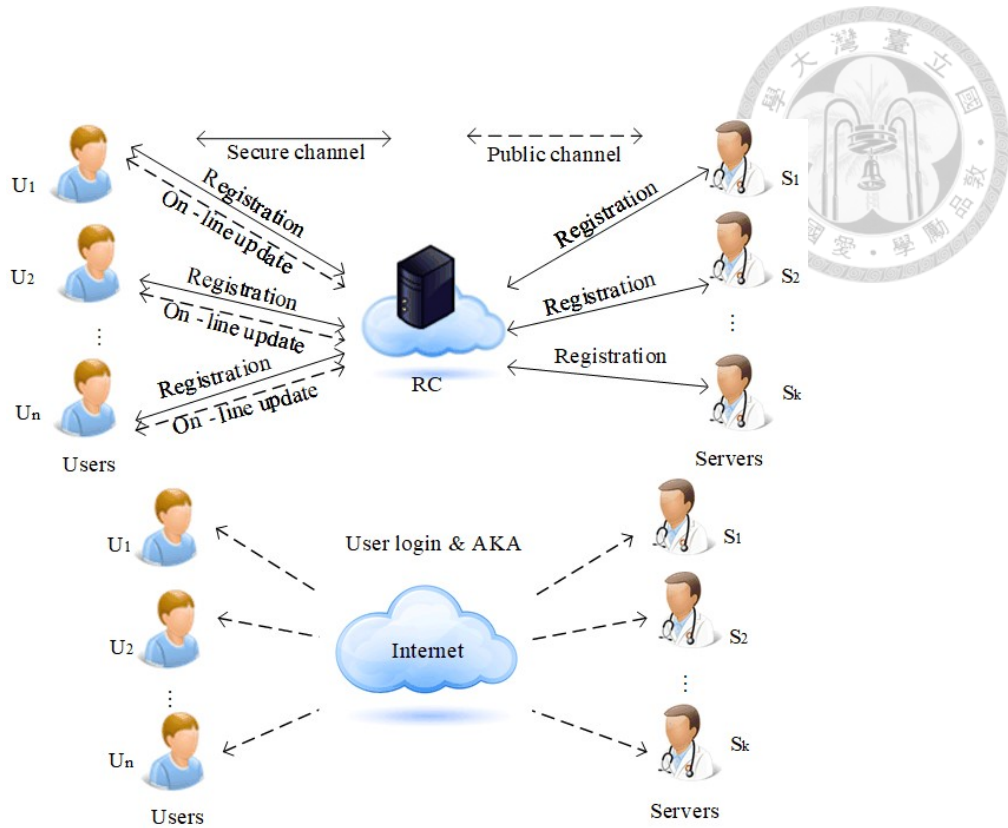


Figure 6. The framework of an AAKA protocol in TMIS

6.2 The Proposed Protocol (Protocol 2: AAKA for TMIS)

Our Protocol 2 consists of five phases: the initialization phase, the registration phase, the online update phase, the login and AKA phase, and the password and biometric change phase.

6.2.1 Initialization Phase

RC selects a large prime q , an elliptic curve $E_q(a, b)$ over a finite field F_q , a base point $P \in E_q(a, b)$, a one-way hash function $h: \{0, 1\}^* \rightarrow Z_q^*$. RC decides the maximum transmission delay ΔT and the generation function Gen and the reproduction function Rep of fuzzy extractor. RC chooses $x \in Z_q^*$, keeps x as the master key, and computes



$X=x \cdot P$. RC lets $\mathit{Pub} = \{X, h, \mathit{Gen}, \mathit{Rep}, p, E_p, P, \Delta T\}$ be public parameters.

6.2.2 Registration Phase

We are going to describe the server registration and the user registration phases in this subsection, and they are illustrated in Figure 7 and Figure 8, respectively. All the messages are transmitted in trusted channels in the registration phases.

[Server Registration]

When a new server S_j joins the system, S_j chooses an identity ID_{S_j} , and sends it to RC through a trusted channel. After receiving ID_{S_j} from the server, RC computes S_j 's secret key $k_{S_j} = h(ID_{S_j} || x)$, and sends $\{k_{S_j}, \mathit{Pub}\}$ to S_j through a trusted channel.

[User Registration]

When a patient U_i with a portable device wants to be a legal user in the TMIS, the following steps are performed.

Step 1: U_i freely chooses an identity ID_{U_i} and a password PW_i . Note that ID_{U_i} can be either the real identity of U_i or just a pseudonym, which enables U_i to achieve strong anonymity. U_i then imprints the biometric via a sensor to get B_i , and the device executes the generation function Gen on B_i to produce the secret string SS_i and the helper string HLP_i , i.e. $(SS_i, HLP_i) = \mathit{Gen}(B_i)$.

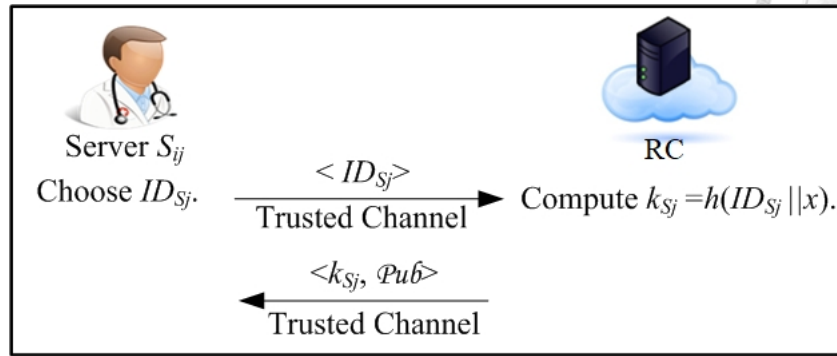


Figure 7. Server registration phase of our Protocol 2 (AAKA for TMIS)

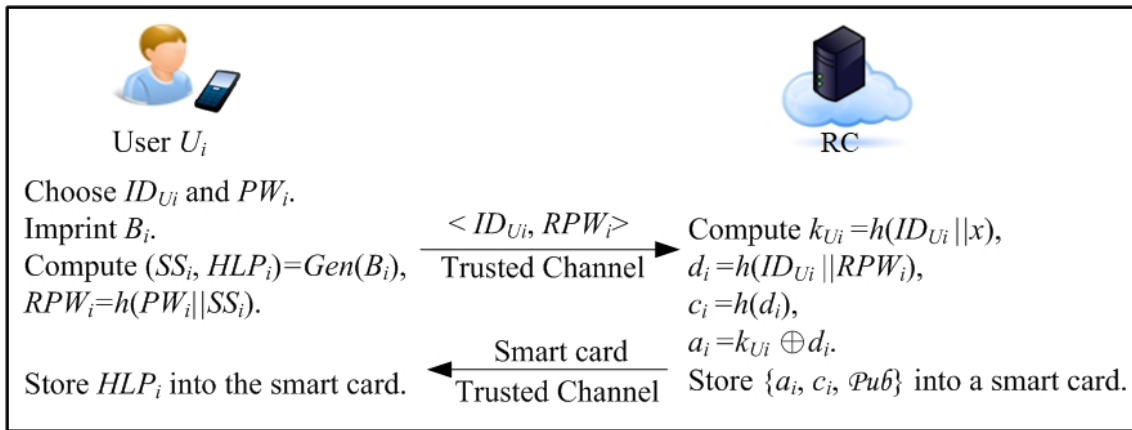


Figure 8. User registration phase of our Protocol 2 (AAKA for TMIS)

The device then computes the regenerated password $RPW_i = h(PW_i || SS_i)$ and sends $\langle ID_{U_i}, RPW_i \rangle$ to RC.

Step 2: After receiving a request message from the user, RC computes U_i 's secret key

k_{U_i} by the master key x , i.e. $k_{U_i} = h(ID_{U_i} || x)$, $d_i = h(ID_{U_i} || RPW_i)$, and the

validation value $c_i = h(d_i)$. RC encrypts k_{U_i} by d_i , i.e. $a_i = k_{U_i} \oplus d_i$, and stores $\{a_i,$

$c_i, Pub\}$ into a smart card, and sends it to U_i .

Step 3: After receiving the smart card from RC, U_i stores the helper string HLP_i into the



smart card. Finally, the smart card contains $\{HLP_i, a_i, c_i, Pub\}$.

[Member revocation phase]

When a legal user or a legal server wants to revoke the previous authorization or RC wants withdraw the authorization of a legal user or a legal server, RC adds the identity of the revoked member to the certificate revocation list (CRL) and issues the updated CRL to each member.

6.2.3 Online Update Phase

Before a patient U_i logs in to an unfamiliar server S_j , he/she has to run the online update phase once to get the public key PK_j of S_j and the common secret key C_{ij} between U_i and S_j . U_i can delete $\langle ID_{S_j}, PK_j, C_{ij} \rangle$, which are stored in the smart card, at any time after the online update phase. But after deleting it, U_i has to execute online update phase again to get $\langle ID_{S_j}, PK_j, C_{ij} \rangle$ before U_i logs into server S_j . U_i can ask a batch of online update phase for different servers, and can ask for the same server more than once. The online update phase is illustrated in Figure 9 and performed as following steps. All the messages are transmitted through public channels.

Step 1: U_i inputs the identity ID_{U_i} and the password PW_i to the smart card and imprints

the biometric impression at the sensor to get B_i .

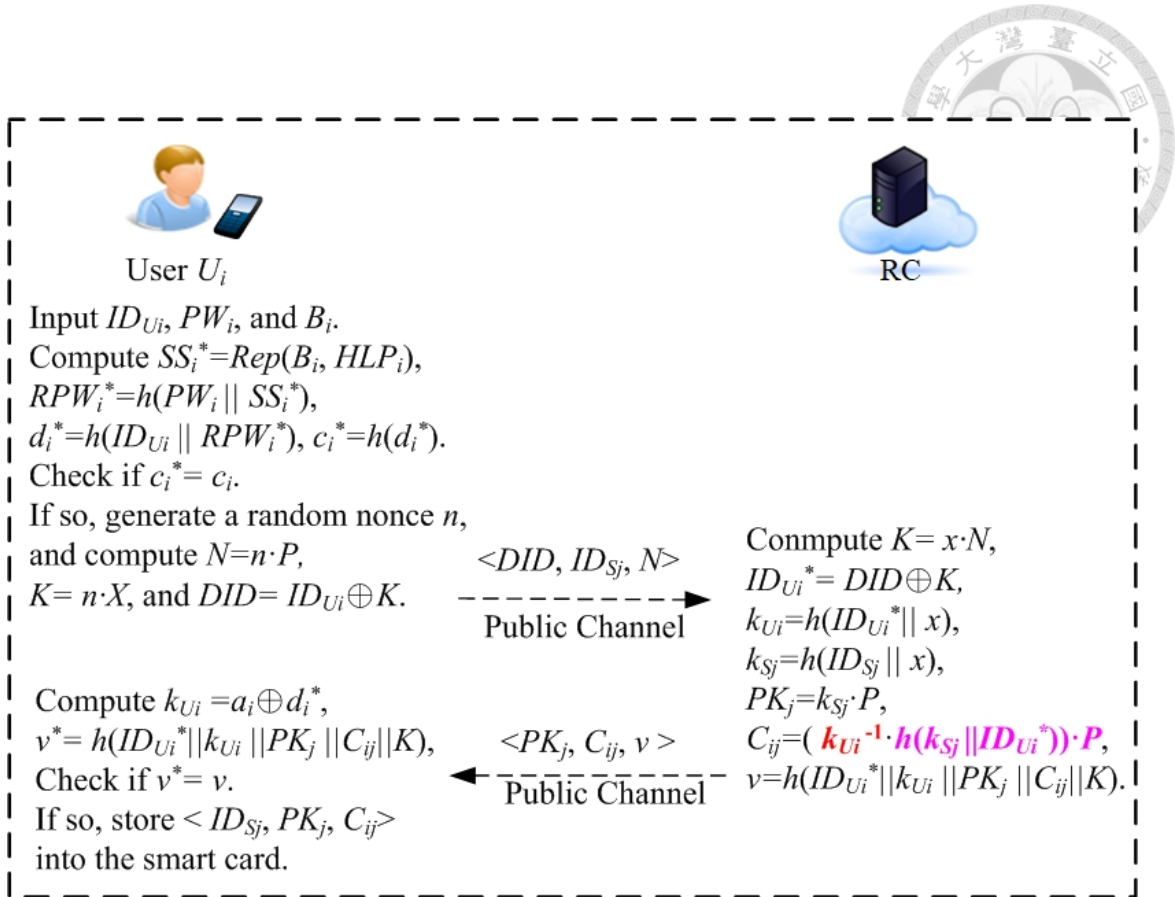


Figure 9. Online update phase of our Protocol 2 (AAKA for TMIS)

U_i 's smart card produces the secret string SS_i^* by executing the reproduction function Rep on the biometric B_i and the helper string HLP_i , i.e. $SS_i^* = Rep(B_i, HLP_i)$. U_i 's device computes $RPW_i^* = h(PW_i || SS_i^*)$, $d_i^* = h(ID_{U_i} || RPW_i^*)$, and $c_i^* = h(d_i^*)$, and checks if $c_i^* = c_i$. If so, the validity of U_i is confirmed, and U_i 's device continues the procedure. Otherwise, U_i 's device terminates it.

Step 2: U_i 's device generates a random nonce $n \in Z_p^*$ and computes $N = n \cdot P$, $K = n \cdot X$, and $DID = ID_{U_i} \oplus K$. U_i 's device then sends $\langle DID, ID_{S_j}, N \rangle$ to RC.

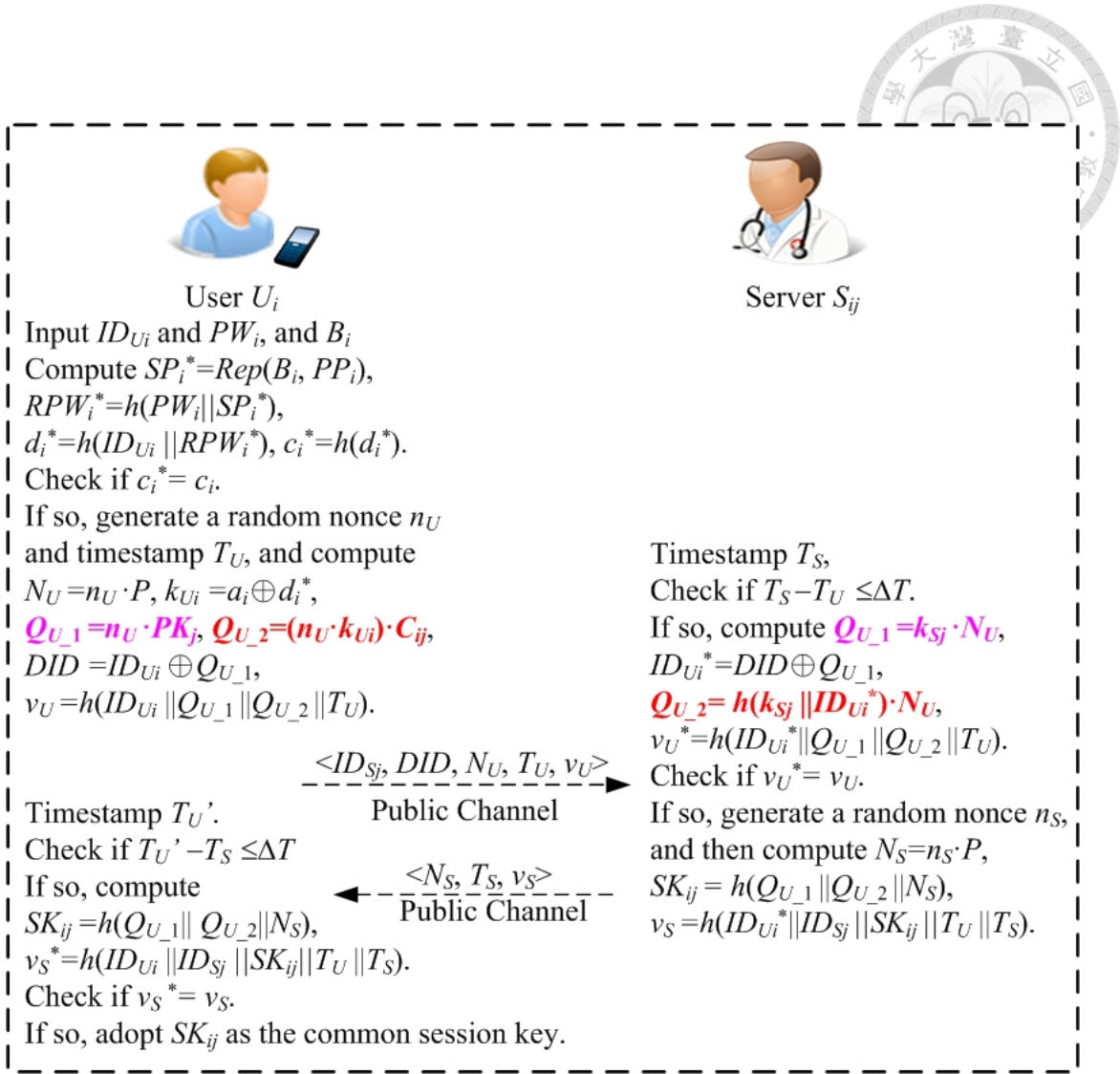


Figure 10. Login and AAKA phase of our Protocol 2 (AAKA for TMIS)

Step 3: After receiving a request message from the user, RC computes $K = x \cdot N$, $ID_{U_i}^* =$

$DID \oplus K$, U_i 's secret key $k_{U_i} = h(ID_{U_i}^* || x)$, $k_{S_j} = h(ID_{S_j} || x)$, S_j 's public key PK_j

$= k_{S_j} \cdot P$, and $C_{ij} = (k_{U_i}^{-1} \cdot h(k_{S_j} || ID_{U_i}^*)) \cdot P$. Finally, RC computes the validation

value $v = h(ID_{U_i}^* || k_{U_i} || PK_j || C_{ij} || K)$, and sends $\langle PK_j, C_{ij}, v \rangle$ to U_i .

Step 4: U_i 's device computes the secret key $k_{U_i} = a_i \oplus d_i^*$ and the validation value $v^* =$

$h(ID_{U_i}^* || k_{U_i} || PK_j || C_{ij} || K)$, and checks if $v^* = v$. If so, it stores $\langle ID_{S_j}, PK_j, C_{ij} \rangle$



into the smart card.

6.2.4 Login and AKA Phase

When a patient U_i wants to log in to a server S_j , the following steps are performed.

Figure 10 illustrates the login and AKA phase.

Step 1: It is the same as the procedure of Step 1 in the online update phase.

Step 2: U_i 's device generates a random nonce n_U and a timestamp T_U , computes $N_U =$

$n_U \cdot P$, and derives the secret key k_{U_i} by d_i^* and a_i , which are computed in Step 1

and stored in the smart card, respectively, i.e. $k_{U_i} = a_i \oplus d_i^*$. U_i 's device

searches S_j 's public key PK_j and C_{ij} in the smart card by S_j 's identity ID_{S_j} , and

computes $Q_{U_1} = n_U \cdot PK_j$ and $Q_{U_2} = (n_U \cdot k_{U_i}) \cdot C_{ij}$. U_i 's device masks the real

identity ID_{U_i} by Q_{U_1} , which only can be computed by the one who has n_U or

k_{S_j} , i.e. the dynamic identity $DID = ID_{U_i} \oplus Q_{U_1}$, and computes the validation

value $v_U = h(ID_{U_i} || Q_{U_1} || Q_{U_2} || T_U)$. The device then transmits the login

request $\langle ID_{S_j}, DID, N_U, T_U, v_U \rangle$ to S_j through an untrusted channel.

Step 3: When the server S_j receives the login request message from U_i , S_j generates a

timestamp T_S , and verifies $(T_S - T_U)$. If $(T_S - T_U) > \Delta T$, S_j rejects the login

request; otherwise, S_j continues the process. S_j computes $Q_{U_1} = k_{S_j} \cdot N_U$, derives



the real identity $ID_{U_i}^*$ of U_i by computing $DID \oplus Q_{U_{-1}}$, and then computes $Q_{U_{-2}}$
 $= h(k_{S_j} || ID_{U_i}^*) \cdot N_U$, and the validation value $v_U^* = h(ID_{U_i}^* || Q_{U_{-1}} || Q_{U_{-2}} || T_U)$. S_j
 verifies v_U^* , rejects the login request if $v_U^* \neq v_U$, and continues the process
 otherwise. S_j generates a random nonce n_S in Z_p^* and computes $N_S = n_S \cdot P$, the
 session key $SK_{ij} = h(Q_{U_{-1}} || Q_{U_{-2}} || N_S)$, and the validation value $v_S = h(ID_{U_i}^*$
 $|| ID_{S_j} || SK_{ij} || T_U || T_S)$. S_j then sends $\langle N_S, T_S, v_S \rangle$ to U_i through an untrusted
 channel.

Step 4: After receiving $\langle N_S, T_S, v_S \rangle$, U_i 's device generates a timestamp T_U' , and
 verifies $(T_U' - T_S)$. If $(T_U' - T_S) > \Delta T$, the device aborts the session; otherwise,
 the device continues the process. The device computes the session key $SK_{ij} =$
 $h(Q_{U_{-1}} || Q_{U_{-2}} || N_S)$ and the validation value $v_S^* = h(ID_{U_i} || ID_{S_j} || SK_{ij} || T_U || T_S)$,
 and verifies v_S^* , adopts SK_{ij} as the common session key if $v_U^* = v_U$, and sborts
 the session otherwise.

Correctness: The session keys computed by U_i and S_j are identical, since $Q_{U_{-1}} =$
 $n_U \cdot PK_j = n_U \cdot k_{S_j} \cdot P = k_{S_j} \cdot N_U$, and $Q_{U_{-2}} = (n_U \cdot k_{U_i}) \cdot C_{ij} = (n_U \cdot k_{U_i}) \cdot (k_{U_i}^{-1} \cdot h(k_{S_j} || ID_{U_i}^*)) \cdot P =$
 $(n_U \cdot h(k_{S_j} || ID_{U_i}^*)) \cdot P = h(k_{S_j} || ID_{U_i}^*) \cdot N_U$.



6.2.5 Password and Biometric Change Phase

When a user U_i wants to change the password or biometric impression, U_i can change them on his/her own by performing the following steps.

Step 1: It is the same as the procedure of Step 1 in the online update phase.

Step 2: U_i inputs the new password PW_i^{new} , and imprints new biometric impression B_i^{new} .

U_i 's device performs the generation function on B_i^{new} to get the new secret string SS_i^{new} and the new helper string HLP_i^{new} , i.e. $(SS_i^{new}, HLP_i^{new}) = Gen(B_i^{new})$, and computes the new regenerated password $RPW_i^{new} = h(PW_i^{new} || SS_i^{new})$, $d_i^{new} = h(ID_i || RPW_i^{new})$, $a_i^{new} = a_i \oplus d_i^* \oplus d_i^{new}$, and the new validation value $c_i^{new} = h(d_i^{new})$. U_i 's smart card then replaces a_i , c_i , and HLP_i with a_i^{new} , c_i^{new} , and HLP_i^{new} , respectively.

6.3 Characteristic Analysis

We analyze the properties of the proposed Protocol 2 point by point in the following.

- (P1) **Three-factor authentication:** We use biometric as the third factor of a user in Protocol 2. That is, a user uses three factors to login a server: the password, the smart card, and the biometric. Thus, Protocol 2 achieves three-factor authentication.



- (P2) **Applicability of multi-server environments:** Servers are regarded as independent entities and have distinct secret keys $k_{S_j} = h(ID_{S_j} || x)$ in Protocol 2, hence Protocol 2 is applicable to multi-server environments.
- (P3) **User anonymity** and (P4) **User untraceability (unlinkability):** A user U_i mask its identity ID_{U_i} by Q_{U-1} , where $Q_{U-1} = n_U \cdot PK_j$. Since only S_j and U_i can compute Q_{U-1} , no third party can obtain Q_{U-1} to get U_i 's real identity ID_{U_i} , thus Protocol 2 achieves user anonymity. Since n_U is different in each session, no third party can derive the relation between any two login transmissions. Moreover, we will formal prove that Protocol 2 achieves user anonymity on ECDDH assumption in Theorem 5. Moreover, a user can log in to a server under a pseudonym to achieve strong anonymity if the user chooses the pseudonym to be his/her identity in the user registration phase.
- (P5) **Perfect forward secrecy:** We will formal prove that Protocol 2 achieves perfect forward secrecy on ECDDH assumption in Theorem 4.
- (P6) **Member revocation:** Protocol 2 does not deliberate on the member revocation problem, but it can use the certificate revocation list (CRL) to deal with the member revocation problem.



- (P7) **Independent authentication** and (P8) **Table free**: There is a trade off between authenticate dependently and maintain tables. A user and a server in Protocol 2 can independently authenticate with each other without the help of any third party if they store the login information, otherwise, they need to perform online update phase again to get the login information to execute the authentication and key exchange.
- (P9) **Public key announcement free**: There are public keys of servers need to be issued to the users. Before a user wants to login a foreign server, the user has to make online update with RC to get the server's public key and the secret information.
- (P10) **Formal security proof**: We will give the formal proofs of Protocol 2 in Section 6.4, and the security of Protocol 2 is based on the Elliptic Curve Decisional Diffie-Hellman (ECDDH) problems.

6.4 Security Analysis

In this subsection, we analyze Protocol 2 in the random oracle model [7]. The random oracle model assumes that the hash function is actually a true random function and it produces a random value for each new query. In the random oracle model, the



security of the proposed protocol is based on the ECDDH problem. We formally prove that the proposed protocol offers unforgeability, session key secrecy, and perfect forward secrecy, and provides user anonymity. We will prove that Protocol 2 offers existential unforgeability, session key secrecy and perfect forward secrecy in Theorem 4, and prove that Protocol 2 achieves user anonymity in Theorem 5.

Theorem 4 *The proposed Protocol 2 offers existential unforgeability, session key secrecy and perfect forward secrecy against adaptive chosen ID attacks on Elliptic Curve Decisional Diffie-Hellman (ECDDH) assumption and hash function assumption.*

Proof: Suppose that there is a probabilistic polynomial time adversary \mathcal{A} who can break the unforgeability or session key secrecy or perfect forward secrecy of Protocol 2 (AAKA for TMIS) with non-negligible advantage ε , and given ID_U and ID_S . Then we can construct an algorithm \mathcal{B} to solve Elliptic Curve Decisional Diffie-Hellman (ECDDH) problem with non-negligible advantage. Let q_U and q_S denote the numbers of users and servers, respectively. \mathcal{B} is given an instance $(p, E_p, P, A=aP, B=bP, \text{ and } C=cP)$ of the ECDDH problem. Then \mathcal{B} 's goal is to determine whether $C = abP$. \mathcal{B} runs \mathcal{A} as a subroutine and simulates its attack environment. First, \mathcal{B} chooses x and sets the public system parameters $\mathcal{Pub} = \{X, h, Gen, Rep, p, E_p, P, \Delta T\}$ by letting $X = xP$. \mathcal{B} permeates

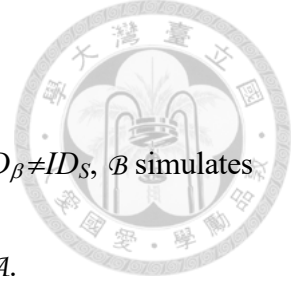


the *ECDDH* problem into the queries on user U (ID_U) and server S (ID_S), which are asked by \mathcal{A} .

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once. \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{SK}*, and no *Corrupt* (α) or *Reveal* (Π_α^S) query is asked before *Test_{SK}* (Π_α^S) query.

\mathcal{B} maintains the list L_H to ensure identical responses and avoid collision of the hash queries, and is L_H empty in the beginning. \mathcal{B} adds $((ID_V || x), k_V)$, $((k_V || ID_U), \text{NULL})$, and $((ID_U || x), \text{NULL})$ to L_H . \mathcal{B} lets $k_U^{-1} \cdot P = A$, $h(k_V || ID_U) \cdot P = B$, and $k_V = h(ID_V || x)$. \mathcal{B} simulates the oracle queries of \mathcal{A} as follows:

- **Hash** (D): When \mathcal{A} makes an *H*-query for D , \mathcal{B} returns w if $(D, w) \in L_H$. Otherwise, \mathcal{B} returns a random value w and adds (D, w) to L_H .
- **Extract** (ID): There are two types of extract query.
 - **Extract_{user}** (ID_α, RPW_α): When \mathcal{A} asks a user Extract-query for $ID_\alpha \neq ID_U$, \mathcal{B} makes *Hash*($ID_\alpha || x$) query to get k_α , simulates *Hash*($ID_\alpha || RPW_\alpha$) query to get d_α , simulates *Hash*(d_α) query to get c_α , computes $a_\alpha = k_\alpha \oplus d_\alpha$, and returns $\{a_\alpha, c_\alpha, \text{Pub}\}$ to \mathcal{A} .



- **Extract_{server}** (ID_β): When \mathcal{A} asks a server Extract-query for $ID_\beta \neq ID_S$, \mathcal{B} simulates

$Hash (ID_\beta || x)$ query to get k_β , and then returns $\{k_\beta, Pub\}$ to \mathcal{A} .

• **Send** (Π_α^S, m): There are four types of send query.

- **Send_{update}** ($\Pi_\alpha^S, Start$): When \mathcal{A} asks this query for ID_α , \mathcal{B} generates a random

nonce n and computes $N = n \cdot P$, $K = n \cdot X$, and $DID = ID_\alpha \oplus K$. \mathcal{B} returns $\langle DID,$

$ID_\beta, N \rangle$ to \mathcal{A} .

- **Send_{update}** ($RC, \langle DID, ID_\beta, N \rangle$): \mathcal{B} computes $K = x \cdot N$ and $ID_\alpha = DID \oplus K$, and

simulates $Hash (ID_\alpha || x)$ query to get k_α and $Hash (ID_\beta || x)$ query to get k_β .

1) If $ID_\alpha = ID_U$ and $ID_\beta = ID_S$, \mathcal{B} lets $C_{US} = C$.

2) If $ID_\alpha = ID_U$ and $ID_\beta \neq ID_S$, \mathcal{B} lets $C_{U\beta} = h(k_\beta || ID_U) \cdot A$.

3) If $ID_\alpha \neq ID_U$, \mathcal{B} computes $PK_\beta = k_\beta \cdot P$, and $C_{\alpha\beta} = (k_\alpha^{-1} \cdot h(k_\beta || ID_\alpha)) \cdot P$.

\mathcal{B} then simulates $Hash (ID_\alpha || k_\alpha || PK_\beta || C_{\alpha\beta} || K)$ query to get v and returns

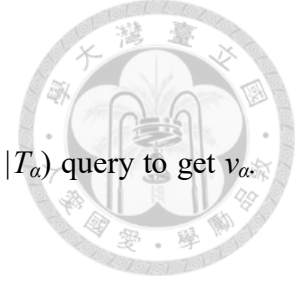
$\langle PK_\beta, C_{\alpha\beta}, v \rangle$ to \mathcal{A} .

- **Send_{AKA}** ($\Pi_\alpha^S, Start$): \mathcal{B} generates a random nonce n_α and a timestamp T_α , and

then simulates $Hash (ID_\beta || x)$ query to get k_β , and $Hash (k_\beta || ID_\alpha)$ query to get

$h(k_\beta || ID_\alpha)$. If $ID_\alpha = ID_U$ and $ID_\beta = ID_S$, \mathcal{B} lets $Q_{\alpha,2} = n_\alpha \cdot B$; otherwise, \mathcal{B}

computes $N_\alpha = n_\alpha \cdot P$ and $Q_{\alpha,2} = h(k_\beta || ID_\alpha) \cdot N_\alpha$. \mathcal{B} then computes $Q_{\alpha,1} = k_\beta \cdot N_\alpha$



and $DID = ID_\alpha \oplus Q_{\alpha_1}$. \mathcal{B} simulates $Hash (ID_\alpha || Q_{\alpha_1} || Q_{\alpha_2} || T_\alpha)$ query to get v_α .

\mathcal{B} returns $\langle ID_\beta, DID, N_\alpha, T_\alpha, v_\alpha \rangle$.

- **Send_{AKA}** ($\Pi_\beta^t, \langle ID_\beta, DID, N_\alpha, T_\alpha, v_\alpha \rangle$): \mathcal{B} generates a random nonce n_β and a timestamp T_β , and verifies $(T_\beta - T_\alpha)$. If $(T_\beta - T_\alpha) > \Delta T$, \mathcal{B} returns “Reject”. If $(T_\beta - T_\alpha) \leq \Delta T$, \mathcal{B} continues the process. \mathcal{B} simulates $Hash (ID_\beta || x)$ query to get k_β , and computes $N_\beta = n_\beta \cdot P$, $Q_{\alpha_1} = k_\beta \cdot N_\alpha$, $ID_\alpha = DID \oplus Q_{\alpha_1}$. If $ID_\alpha = ID_U$ and $ID_\beta = ID_S$, \mathcal{B} uses v_α to find $((ID_\alpha || Q_{\alpha_1} || Q_{\alpha_2} || T_\alpha), v_\alpha)$ in L_H to get Q_{α_2} ; otherwise, \mathcal{B} simulates $Hash (k_\beta || ID_\alpha)$ query to get $h(k_\beta || ID_\alpha)$, and computes $Q_{\alpha_2} = h(k_\beta || ID_\alpha) \cdot N_\alpha$. \mathcal{B} simulates $Hash (ID_\alpha || Q_{\alpha_1} || Q_{\alpha_2} || T_\alpha)$ query to get v_α^* , and verifies v_α^* . \mathcal{B} returns “Reject” if $v_\alpha^* \neq v_\alpha$; \mathcal{B} continues the process otherwise. \mathcal{B} simulates $Hash (Q_{\alpha_1} || Q_{\alpha_2} || N_\beta)$ query to get $SK_{\alpha\beta}$, and $Hash (ID_\alpha || ID_\beta || SK_{\alpha\beta} || T_\alpha || T_\beta)$ query to get v_β . \mathcal{B} returns $\langle N_\beta, T_\beta, v_\beta \rangle$.

- **Execute** (U_α, S_β): When \mathcal{A} asks an Execute (ID_α, ID_β) query, \mathcal{B} returns the transcript $\langle (DID, ID_\beta, N), (PK_\beta, C_{\alpha\beta}, v), (ID_\beta, DID, N_\alpha, T_\alpha, v_\alpha), (N_\beta, T_\beta, v_\beta) \rangle$ by simulating above Send queries.
- **Reveal** (Π_α^S): There are two types of reveal query as follows:

- **Reveal_{SK}** (Π_α^S): \mathcal{B} returns session key SK by simulating above Send queries if the



instance Π_α^S has accepted SK ; otherwise, \mathcal{B} returns a null value.

- **Reveal_{ID}** (Π_α^S): \mathcal{B} returns ID_α .

• **Rot** (U_α , *TYPE*): At most two types of Rot query can be asked for a user U_α . \mathcal{B}

responds by the following three types of Rot query.

- **Rot** (U_α , *PW*): \mathcal{B} returns PW_α .

- **Rot** (U_α , *BI*): \mathcal{B} returns B_α .

- **Rot** (U_α , *SC*): \mathcal{B} simulates **Extract**_{user} (ID_α , RPW_α) query to get $\{a_\alpha, c_\alpha, Pub\}$, and

then returns $\{a_\alpha, c_\alpha, Pub\}$.

• **Corrupt** (Π_α^S): \mathcal{B} simulates an **Extract** (ID_α , RPW_α) query to get $\{a_\alpha, c_\alpha, Pub\}$, and

then returns PW_α , B_α , and $\{a_\alpha, c_\alpha, Pub\}$ to \mathcal{A} .

• **Test_{SK}** (Π_α^S): \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns $SK_{\alpha\beta}$ if $b = 1$, and returns a

random value otherwise.

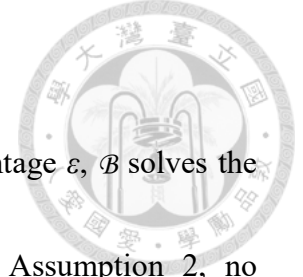
If \mathcal{A} answers $b=1$ to the **Test_{SK}** query, \mathcal{B} answers $C=abP$ to the ECDDH problem. If

\mathcal{A} answers $b \neq 1$ to the **Test_{SK}** query, \mathcal{B} answers $C \neq abP$ to the ECDDH problem. The

success probability of \mathcal{B} depends on the event that \mathcal{A} asks the **Test_{SK}** query for user U

(ID_U) and server S (ID_S) and correctly guesses b in the **Test_{SK}** query. In the above

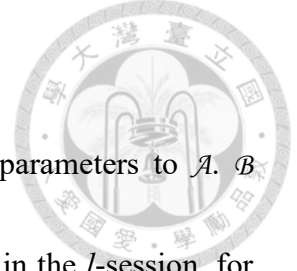
simulation, the probability that \mathcal{A} asks the **Test_{SK}** query in the l -th session is $1/q_U \cdot q_S$. If \mathcal{A}



correctly guesses b in the $Test_{SK}$ query with a non-negligible advantage ϵ , \mathcal{B} solves the ECDDH problem with a non-negligible advantage $\epsilon/q_U \cdot q_S$. By Assumption 2, no polynomial-time algorithm can solve ECDDH problem with non-negligible advantage, it is a contradiction. Hence, no probabilistic polynomial time adversary \mathcal{A} has a non-negligible advantage in the above game played between \mathcal{A} and \mathcal{B} . Then by Definition 6, Protocol 2 offers existential unforgeability, session key secrecy and perfect forward secrecy against adaptive chosen ID attacks. \square

Theorem 5 *The proposed Protocol 2 maintains user anonymity on Elliptic Curve Decisional Diffie-Hellman (ECDDH) and hash function assumptions.*

Proof: Suppose that there is a probabilistic polynomial time adversary \mathcal{A} who can break the anonymity of Protocol 2 (AAKA for TMIS) with running time T , advantage ϵ . Then we can construct an algorithm \mathcal{B} to solve Elliptic Curve Decisional Diffie-Hellman (ECDDH) problem with non-negligible advantage. Let q_U , q_S , and q_{ns} , respectively, denote the numbers of users, servers, and sessions. \mathcal{B} is given an instance $(p, E_p, P, A = aP, B = bP, \text{ and } C = cP)$ of the elliptic curve decision Diffie-Hellman problem. Then \mathcal{B} 's goal is to determine whether $C = abP$. \mathcal{B} runs \mathcal{A} as a subroutine and simulates its attack environment. First, \mathcal{B} chooses x and sets the public system parameters $Pub = \{X, h,$



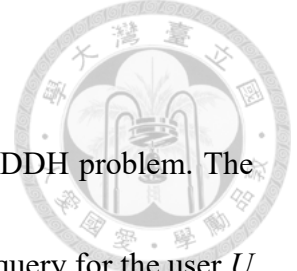
$Gen, Rep, p, E_p, P, \Delta T\}$ by letting $X = x \cdot P$. \mathcal{B} gives the public parameters to \mathcal{A} . \mathcal{B} permeates $ECDDH$ problem into the queries, which are asked by \mathcal{A} in the l -session, for user $U (ID_U)$ and server $S (ID_S)$. Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, and the user instance Π_α^S and the server instance Π_β^t are partners. \mathcal{B} maintains the list L_H to ensure identical responses and avoid collision of the hash queries. \mathcal{B} simulates the oracle queries of \mathcal{A} as follows:

- **Hash** (m), **Extract** (ID), **Send_{update}** (Π_α^S , Start), **Send_{AKA}** (Π_α^S , Start), **Send_{AKA}** (Π_β^t , $\langle ID_\beta, DID, N_\alpha, T_\alpha, v_\alpha \rangle$), **Execute** (U_α, S_β), **Reveal** (Π_α^S), **Rot** (U_α, M), and **Corrupt** (Π_α^S) are identical to those queries in the proof of Theorem 4.

- **Send_{update}** (Π_α^S , $\langle DID, ID_\beta, N \rangle$): \mathcal{B} computes $K = x \cdot N$ and $ID_\alpha = DID \oplus K$, simulates **Hash** ($ID_\alpha || x$) query to get k_α , simulates **Hash** ($ID_\beta || x$) query to get k_β , and computes $PK_\beta = k_\beta \cdot P$ and $C_{\alpha\beta} = (k_\alpha^{-1} \cdot h(k_\beta || ID_\alpha)) \cdot P$. If $ID_\alpha = ID_U$ and $ID_\beta = ID_S$, \mathcal{B} lets $PK_\beta = PK_S = B$. \mathcal{B} then simulates **Hash** ($ID_\alpha || k_\alpha || PK_\beta || C_{\alpha\beta} || K$) query to get v and returns $\langle PK_\beta, C_{\alpha\beta}, v \rangle$ to \mathcal{A} .

- **Test_{ID}** (Π_α^S): When \mathcal{A} makes a **Test** query, \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} then returns ID_α if $b = 1$, and else returns a random number.

If \mathcal{A} answers $b = 1$ to the **Test_{ID}** query, \mathcal{B} answers $C = abP$ to the $ECDDH$ problem.



If \mathcal{A} answers $b \neq 1$ to the $Test_{ID}$ query, \mathcal{B} answers $C \neq abP$ to the ECDDH problem. The success probability of \mathcal{B} depends on the event that \mathcal{A} asks the $Test_{ID}$ query for the user U (ID_U) and the server S (ID_S) in the l -session. In the above simulation, the probability of \mathcal{A} asking the $Test_{ID}$ query for ID_U is $1/q_U$, and asking the Send query for ID_S in the l -session is $1/q_S \cdot q_{ns}$. If \mathcal{A} correctly guesses b in the $Test_{ID}$ query with non-negligible advantage ε , \mathcal{B} solves $mECDDH$ problem with non-negligible advantage at least $\varepsilon/q_U \cdot q_S \cdot q_{ns}$. By Assumption 2, no polynomial-time algorithm can solve ECDDH problem with non-negligible advantage, it is a contradiction. Hence, no probabilistic polynomial time adversary \mathcal{A} has a non-negligible advantage in the above game played between \mathcal{A} and \mathcal{B} . Then by Definition 7, Protocol 2 offers existential user anonymity against adaptive chosen ID attacks. \square



Chapter 7

PUF Based AKE Protocol for IoT without Verifiers and Explicit CRPs

In the chapter, we will propose a PUF based AKE protocol for IoT (our Protocol 3), which has no verifier and no explicit challenge–response pair (CRP). Any two of the IoT nodes in the proposed Protocol 3 can freely mutual authenticate with each other without the help of a verifier as well as no explicit challenge–response pair (CRP) is stored. We will describe the framework and the system assumptions first, and then expatiate on our Protocol 3. We then formally prove that our Protocol 3 is secure on Elliptic Curve Computational Diffie-Hellman (ECCDH) problem, Decisional Bilinear Diffie-Hellman (DBDH) problem, and hash function assumptions.

7.1 The Framework of a PUF based AKE Protocol for IoT

The architecture of our Protocol 3 consists of one trusted security credential generator (SCG), some data providers (DP), and many IoT nodes. The SCG is responsible for setting up the system and generating security credentials for data providers and IoT nodes. Data providers are responsible for providing public data for IoT nodes during AKE and all the data providers store identical public helper data.

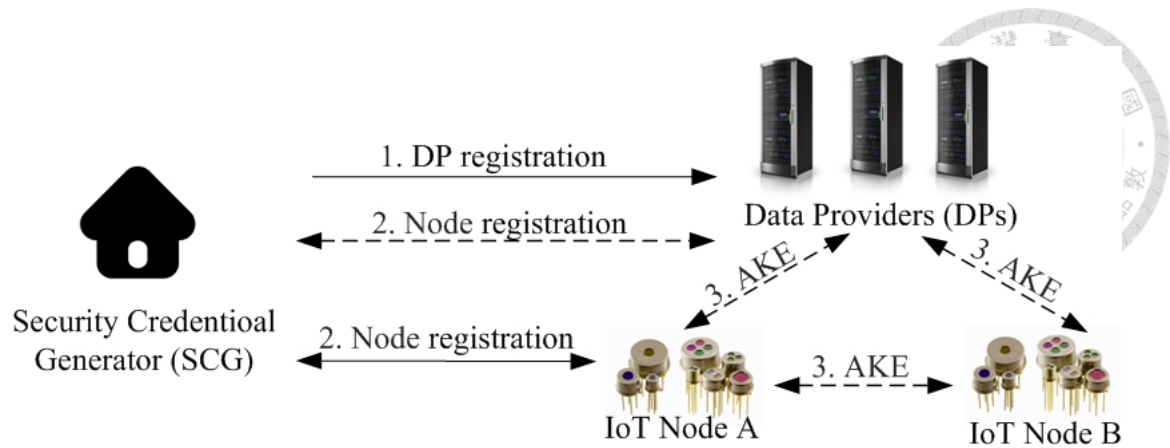


Figure 11. The framework of our Protocol 3 (AKE for IoT)

After an IoT node joining the system, all the data providers would store the public helper data of the IoT node. When a data provider is disabled, the IoT nodes may seek for another data provider to get the public helper data. IoT nodes are regarded as physical unclonable function (PUF) enabled provers and have limited computational abilities, and they prove their authenticity to another IoT node by using their embedded PUF instances. Figure 11 illustrates the framework of our Protocol 3.

7.2 The Proposed Protocol (Protocol 3: AKE for IoT)

We describe the system assumptions in an IoT environment, and propose our Protocol 3 in the section. Our Protocol 3 consists of three phases: initialization, registration, and IoT node the authentication and key exchange (AKE). The registration phase and the AKE phase are illustrated in Figure 12, Figure 13, respectively.

7.2.1 System Assumptions

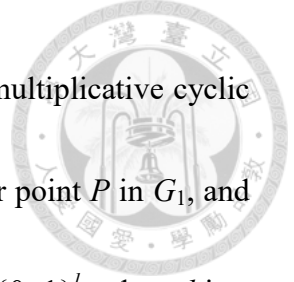


System Model: SCG is assumed to be a trusted entity that generates security credentials. Assume that PUF is embedded in each IoT nodes, and the IoT nodes have capability to perform cryptographic hash functions and elliptic curve operations, including pairing operation and scalar multiplication. DP is assumed to have stable storage, stable connection, and the capability to perform a cryptographic hash function.

Threat Assumptions: Assume that an adversary can eavesdrop, replay, insert, delete, or modify any message over an untrustworthy communication channel, and any legitimate IoT node can behave as an adversary. The challenge-response characteristics is an implicit property in a PUF instance embedded in an IoT node that adversaries cannot access it. The goal of an adversary is to impersonate an IoT node to authenticate with another IoT node.

7.2.2 Initialization Phase

SCG sets up the system in the initial. SCG selects a large prime q , lets F_q be a finite field of integers modulo a large prime number q , and lets an elliptic curve E over F_q be defined by an equation of the form $y^2 = x^3 + ax + b$, where $a, b \in F_q$ satisfy $4a^3 + 27b^2 \neq 0 \pmod{q}$. Let $E(F_q)$ denotes the set of all the points on E . SCG produces a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$, where G_1 is a subgroup of the additive group of points on



an elliptic curve over a finite field $E(F_q)$ with order q , and G_2 is a multiplicative cyclic subgroup with the same order q over F_q . SCG picks a base generator point P in G_1 , and chooses three one-way hash functions $h_1: \{0, 1\}^* \rightarrow Z_q^*$, $h_2: \{0, 1\}^* \rightarrow \{0, 1\}^l$, where l is a constant integer, and $H: \{0, 1\}^* \rightarrow G_1$. SCG generates the master key $MSK \in Z_q^*$. SCG chooses a constant $n \in \mathbb{N}$, which stands for the number of (Challenge, Response) pairs. For each $i \in \{1, 2, \dots, n\}$, SCG randomly generates a seed $t_i \in_R Z_q^*$, and computes the master private key $MRK_i = h_1(MSK || t_i)$ and the master public key $MPK_i = MRK_i \cdot P$, and then appends (i, t_i, MRK_i) to list L_{key} and keeps it in secret. SCG lets $\mathcal{Pub} = \{p, G_1, G_2, P, \hat{e}, h_1, h_2, H, \{(i, MPK_i)\}_{i=1}^n\}$ be public parameters.

7.2.3 Registration Phase

This subsection present the registration phases of the data provider and the IoT node, and the revocation phase of IoT node. The entire messages are transmitted through a trusted channel in the registration phases, and are transmitted through an unsecure channel in the revocation phase.

[Data provider]

SCG issues a new data provider DP_j by the following processe. SCG generates DP_j 's identity $ID_{DP_j} \in \{0, 1\}^*$, and computes DP_j 's secret key $K_j = H(MSK || ID_{DP_j})$.

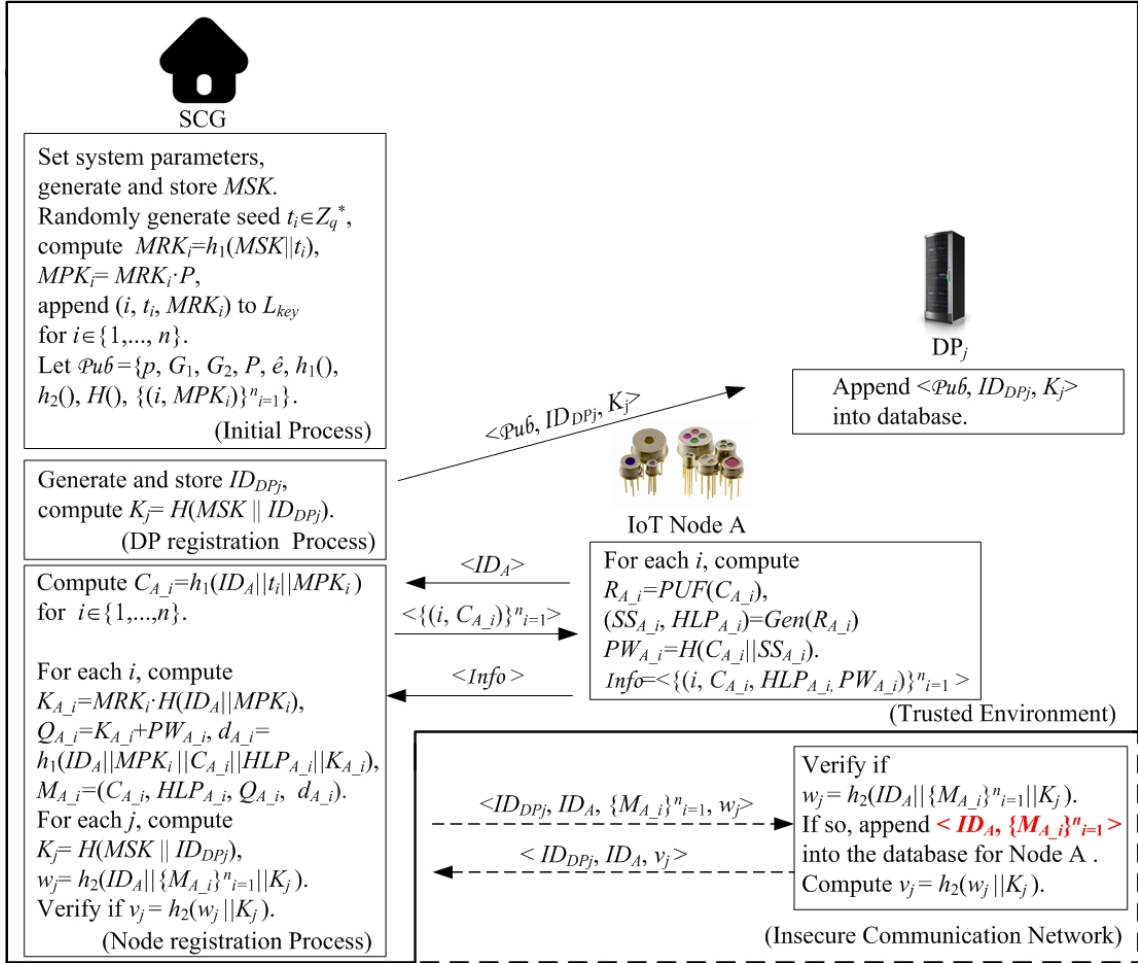


Figure 12. Registration phase of our Protocol 3 (AKE for IoT)

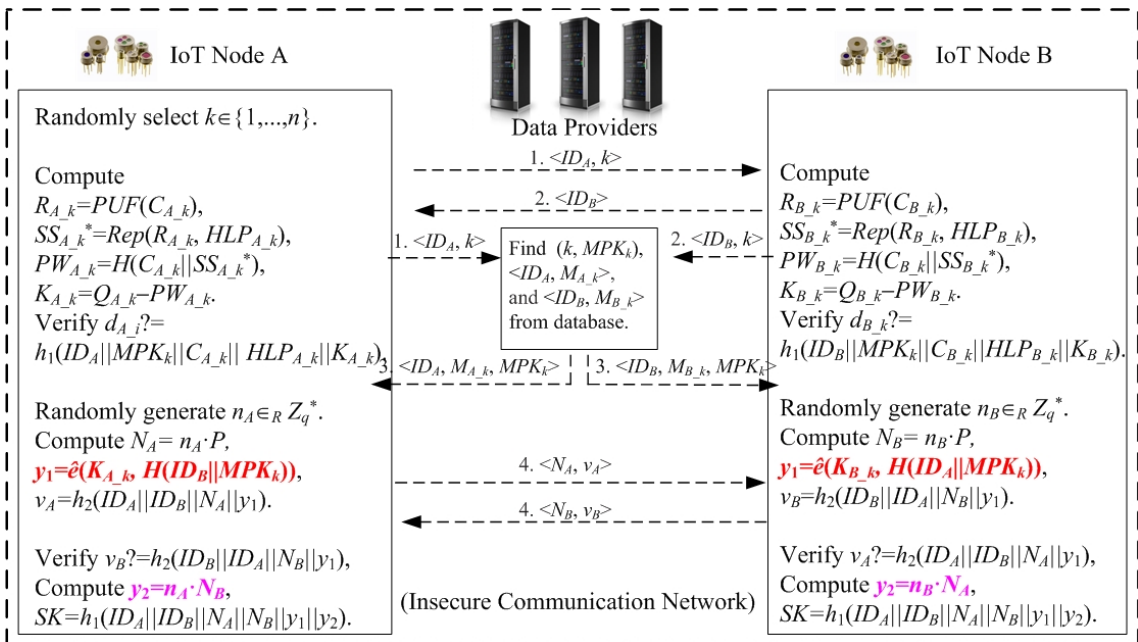
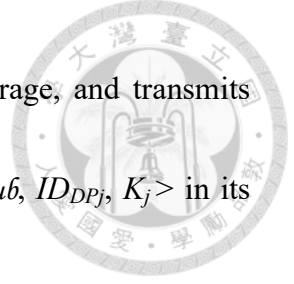


Figure 13. AKE phase of our Protocol 3 (AKE for IoT)



SCG appends ID_{DP_j} to list L_{DP} , which is stored in SCG's storage, and transmits $\langle Pub, ID_{DP_j}, K_j \rangle$ to DP_j via a trusted channel. DP_j then stores $\langle Pub, ID_{DP_j}, K_j \rangle$ in its database.

[IoT node]

This phase is executed for every IoT node in a secure and trusted environment before deploying these IoT nodes in the communication network. Messages in Step1, 2, and 3 are transmitted through trusted channels, and messages in Step 4 are transmitted through unsecure channels.

Step 1: IoT node A sends its identity ID_A to SCG. SCG computes $C_{A_i} = h_1(ID_A || t_i || MPK_i)$

for $i \in \{1, 2, \dots, n\}$, and sends $\langle \{(i, C_{A_i})\}_{i=1}^n \rangle$ back to A .

Step 2: IoT node A computes $R_{A_i} = PUF(C_{A_i})$, executes Gen function on R_{A_i} to

produce the secret key SS_{A_i} and the public key HLP_{A_i} , i.e. $Gen(R_{A_i}) = (SS_{A_i},$

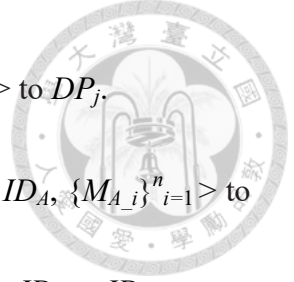
$HLP_i)$. IoT Node A then computes $PW_{A_i} = H(C_{A_i} || SS_{A_i})$ for $i \in \{1, 2, \dots, n\}$, and

returns $\{(i, C_{A_i}, HLP_{A_i}, PW_{A_i})\}_{i=1}^n$ to SCG. SCG then computes $K_{A_i} = MRK_i \cdot$

$H(ID_A || MPK_i)$, $Q_{A_i} = K_{A_i} + PW_{A_i}$, $d_{A_i} = h_1(ID_A || MPK_i || C_{A_i} || HLP_{A_i} || K_{A_i})$,

and $M_{A_i} = (C_{A_i}, HLP_{A_i}, Q_{A_i}, d_{A_i})$ for $i \in \{1, 2, \dots, n\}$.

Step 3: For each data provider DP_j , SCG computes $K_j = H(MSK || ID_{DP_j})$ and $w_j =$



$h_2(ID_A || \{M_{A_i}\}_{i=1}^n || K_j)$, and sends $\langle ID_{DP_j}, ID_A, \{M_{A_i}\}_{i=1}^n, w_j \rangle$ to DP_j .

Step 4: DP_j verifies if $w_j = h_2(ID_A || \{M_{A_i}\}_{i=1}^n || K_j)$. If so, appends $\langle ID_A, \{M_{A_i}\}_{i=1}^n \rangle$ to its database, and then computes $v_j = h_2(w_j || K_j)$ and returns $\langle ID_{DP_j}, ID_A, v_j \rangle$ to SCG. SCG then verifies v_j . If $v_j \neq h_2(w_j || K_j)$, then SCG resends $\langle ID_{DP_j}, ID_A, \{M_{A_i}\}_{i=1}^n, w_j \rangle$ to DP_j .

[IoT revocation phase]

When SCG wants withdraw the authorization of a legal IoT ID_A , SCG executes the following steps to delete ID_A 's data from data providers. The messages are transmitted through unsecure channels.

Step 1: SCG generates a timestamp T . For each data provider DP_j , SCG computes $K_j = H(MSK || ID_{DP_j})$ and $w_j = h_2(ID_A || T || K_j)$, and sends $\langle ID_{DP_j}, ID_A, \text{"Revocation"}, w_j \rangle$ to DP_j .

Step 2: DP_j checks the timestamp T and verifies w_j . If $w_j = h_2(ID_A || T || K_j)$, DP_j deletes $\langle ID_A, \{M_{A_i}\}_{i=1}^n \rangle$ from its database, and then computes $v_j = h_2(w_j || K_j)$ and returns $\langle ID_{DP_j}, ID_A, v_j \rangle$ to SCG. SCG then verifies if $v_j = h_2(w_j || K_j)$. If not, SCG executes Step 1 again.



7.2.4 IoT Node AKE Phase

When IoT Node A wants to mutually authenticate with IoT Node B through public channels and establish a common session key, they perform the following steps:

Step 1: IoT Node A randomly picks $k \in \{1, 2, \dots, n\}$, and sends $\langle ID_A, k \rangle$ to IoT Node B

and the nearest Data Provider, say DP_{j_1} .

Step 2: IoT Node B sends $\langle ID_B \rangle$ to IoT Node A and sends $\langle ID_B, k \rangle$ to the nearest data

provider, say DP_{j_2} .

Step 3: DP_{j_1} finds (k, MPK_k) and $\langle ID_A, M_{A_k} \rangle$ from its database, and sends $\langle ID_A, M_{A_k},$

$MPK_k \rangle$ back to IoT Node A via the same channel. DP_{j_2} finds (k, MPK_k) and

$\langle ID_B, M_{B_k} \rangle$ from its database, and sends $\langle ID_B, M_{B_k}, MPK_k \rangle$ back to IoT Node

B . Note that DP_{j_1} and DP_{j_2} may be the same Data Provider, i.e. $j_1 = j_2$.

Step 4: IoT Node A inputs C_{A_k} into its PUF to get R_{A_k} , i.e. $R_{A_k} = PUF(C_{A_k})$, and

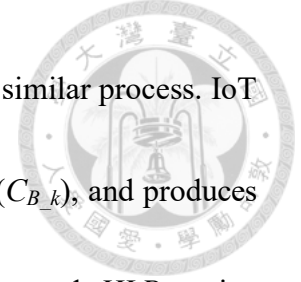
produces the secret key $SS_{A_k}^*$ by executing Rep function on R_{A_k} and HLP_{A_k} , i.e.

$SS_{A_k}^* = Rep(R_{A_k}, HLP_{A_k})$. IoT Node A then computes $PW_{A_k} = H(C_{A_k} || SS_{A_k}^*)$,

and $K_{A_k} = Q_{A_k} - PW_{A_k}$, and verifies if $d_{A_i} = h_1(ID_A || MPK_k || C_{A_k} || HLP_{A_k} || K_{A_k})$.

If so, IoT Node A randomly generates $n_{A \in R} Z_q^*$, and computes $N_A = n_A \cdot P$,

$y_1 = \hat{e}(K_{A_k}, H(ID_B || MPK_k))$, and $v_A = h_2(ID_A || ID_B || N_A || y_1)$, and then sends $\langle N_A,$



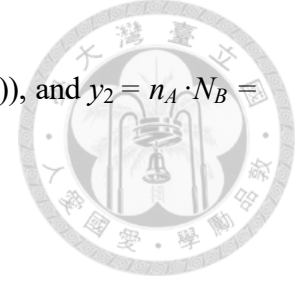
v_A to IoT Node B . IoT Node B simultaneously executes the similar process. IoT Node B inputs C_{B_k} into its PUF to get R_{B_k} , i.e. $R_{B_k} = PUF(C_{B_k})$, and produces the secret key $SS_{B_k}^*$ by executing Rep function on R_{B_k} and HLP_{B_k} , i.e. $SS_{B_k}^* = Rep(R_{B_k}, HLP_{B_k})$. IoT Node B computes $PW_{B_k} = H(C_{B_k} || SS_{B_k}^*)$, and $K_{B_k} = Q_{B_k} - PW_{B_k}$, and verifies if $d_{B_i} = h_1(ID_B || MPK_k || C_{B_k} || HLP_{B_k} || K_{B_k})$. If so, IoT Node B randomly generates $n_B \in \mathbb{Z}_q^*$, and computes $N_B = n_B \cdot P$, $y_1 = \hat{e}(K_{B_k}, H(ID_A || MPK_k))$, and $v_B = h_2(ID_B || ID_A || N_B || y_1)$, and then sends $\langle N_B, v_B \rangle$ to IoT Node A .

Step 5: IoT Node A verifies if $v_B = h_2(ID_B || ID_A || N_B || y_1)$. If so, the validity of IoT Node B is confirmed, and IoT Node A computes $y_2 = n_A \cdot N_B$ and the common session key $SK = h_1(ID_A || ID_B || N_A || N_B || y_1 || y_2)$. Otherwise, aborts the session. IoT Node B simultaneously executes the similar process. IoT Node B verifies if $v_A = h_2(ID_A || ID_B || N_A || y_1)$. If so, the validity of IoT Node A is confirmed, and IoT Node B computes $y_2 = n_B \cdot N_A$ and the session key $SK = h_1(ID_A || ID_B || N_A || N_B || y_1 || y_2)$. Otherwise, aborts the session.

Correctness: The session keys computed by IoT Node A and IoT Node B are identical, since $y_1 = \hat{e}(K_{A_k}, H(ID_B || MPK_k)) = \hat{e}(MRK_k \cdot H(ID_A || MPK_k), H(ID_B || MPK_k))$

$$= \hat{e}(H(ID_A || MPK_k), MRK_i \cdot H(ID_B || MPK_k)) = \hat{e}(K_{B_k}, H(ID_A || MPK_k)), \text{ and } y_2 = n_A \cdot N_B =$$

$$n_A \cdot n_B \cdot P = n_B \cdot n_A \cdot P = n_B \cdot N_A.$$



7.3 Characteristic Analysis

We analyze the properties of the proposed Protocol 3 point by point in the following.

- (P1) **Three-factor authentication**: Protocol 3 adopts PUF as the third authentication factor.
- (P2) **Applicability of multi-server environments**: Protocol 3 focus on the authentication and key exchange between two IoT nodes that this property is not applicable to Protocol 3.
- (P3) **User anonymity** and (P4) **User untraceability (unlinkability)**: Protocol 3 does provide the anonymity and the untraceability of an IoT.
- (P5) **Perfect forward secrecy**: We will prove that Protocol 3 achieves perfect forward secrecy on ECCDH assumption in Theorem 6.
- (P6) **Member revocation**: When an IoT node is revoked, its helper data stored in the data providers would be deleted; nevertheless, a malicious IoT node cannot be revoked if it keeps the helper data in the previous session.
- (P7) **Independent authentication**: When two IoT nodes want to authenticate with



each other, they need to ask for the helper data from the data providers.

- (P8) **Table free**: The helper data of IoT nodes are stored in data providers.
- (P9) **Public key announcement free**: Protocol 3 adopts the identity of an IoT to be its public key; hence no public key needs to be announced.
- (P10) **Formal security proof**: We will give the formal proofs of Protocol 3 in Section 7.4, and the security of Protocol 3 is based on the Elliptic Curve Computational Diffie-Hellman (ECCDH) and the Decisional Bilinear Diffie-Hellman (DBDH) assumptions.

7.4 Security Analysis

In this section, we will analyze the security of the proposed protocol in the random oracle model [7]. In the random oracle model, the hash functions are assumed to be a true random function that produces a random value for each new query. The security of the proposed protocol is based on Elliptic Curve Computational Diffie-Hellman (ECCDH) problem, Decisional Bilinear Diffie-Hellman (DBDH) problem, and hash function assumptions. We give formal proofs of the proposed protocol in Theorem 6 and Theorem 7 to demonstrate that Protocol 3 (AKE for IoT) achieves existential session key secrecy and perfect forward secrecy, and achieves existential



unforgeryability, respectively.

Theorem 6 *The proposed Protocol 3 achieves existential session key secrecy and perfect forward secrecy on the Elliptic Curve Computational Diffie-Hellman (ECCDH) assumption and hash function assumption.*

Proof: Suppose that \mathcal{A} is a polynomial-time adversary, who can break the session key secrecy or the perfect forward secrecy of Protocol 3 (AKE for IoT) with a non-negligible advantage ε . We can construct an algorithm \mathcal{B} to solve the ECCDH problem with a non-negligible advantage by using \mathcal{A} 's ability of breaking Protocol 3. \mathcal{B} is given an instance $(G_1, G_2, P, \hat{e}, q, A = aP, B = bP)$ of ECCDH problem, and \mathcal{B} 's goal is to output abP . \mathcal{B} maintains a hash list L_H to avoid the collision and ensure the identical responses. \mathcal{B} generates master secret key $MSK \in \mathbb{Z}_p^*$ and chooses a constant $n \in \mathbb{Z}$. For each $i \in \{1, 2, \dots, n\}$, \mathcal{B} randomly generates $t_i, MRK_i \in_R \mathbb{Z}_q^*$, appends $(h_1, (MSK || t_i), MRK_i)$ to list L_H , computes master public key $MPK_i = MRK_i \cdot P$, appends (i, t_i, MRK_i) to list L_{key} , and sets the public parameters $Pub = \{p, G_1, G_2, P, \hat{e}, h_1, h_2, H, \{(i, MPK_i)\}_{i=1}^n\}$.

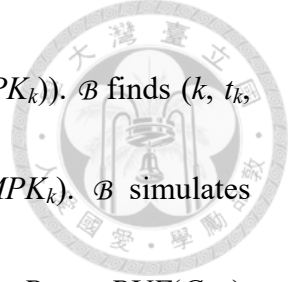
Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash, Extract, Send, Execute, Reveal, Rot, Corrupt*, and *Test_{SK}*, and no *Corrupt*



(α) or *Reveal* (Π_α^s) query is asked before *Test_{SK}* (Π_α^s) query.

\mathcal{B} permeates the ECCDH problem into the queries asked by \mathcal{A} in the l th session for the instance Π_U^s or Π_V^t , where Π_U^s and Π_V^t authenticate mutually and establish a common session key. \mathcal{B} simulates the oracle queries of \mathcal{A} as follows:

- **Hash** (type, M): There are three types of hash query in the proposed protocol. \mathcal{B} returns h if (type, M , h) is in L_H . Otherwise, \mathcal{B} randomly chooses $r_1 \in Z_q^*$ and $r_2 \in \{0,1\}^l$. If type = H , \mathcal{B} computes $R = r_1 \cdot P$, adds (H, M, R, r_1) to L_H , and returns R . If type = h_1 , \mathcal{B} adds (h_1, M, r_1) to L_H , and returns r_1 . If type = h_2 , \mathcal{B} adds (h_2, M, r_2) to L_H , and returns r_2 .
- **Extract** (ID_α): When \mathcal{A} asks the Extract query for ID_α , \mathcal{B} simulates *Hash*(h_1 , ($ID_\alpha || t_i || MPK_i$)) query to get C_{α_i} for $i \in \{1, 2, \dots, n\}$, returns $\langle \{(i, C_{\alpha_i})\}_{i=1}^n \rangle$, and waits for the input. After \mathcal{A} inputting $\{(i, C_{\alpha_i}, PW_{\alpha_i})\}_{i=1}^n$, \mathcal{B} simulates *Hash*(H , ($ID_\alpha || MPK_i$)) query to get $H(ID_\alpha || MPK_i)$, computes $K_{\alpha_i} = MRK_i \cdot H(ID_\alpha || MPK_i)$ and $Q_{\alpha_i} = K_{\alpha_i} + PW_{\alpha_i}$, simulates *Hash*(h_1 , ($ID_\alpha || MPK_i || C_{\alpha_i} || K_{\alpha_i}$)) query to get d_{α_i} , and lets $M_{\alpha_i} = (C_{\alpha_i}, Q_{\alpha_i}, d_{\alpha_i})$ for $i \in \{1, 2, \dots, n\}$. For each data provider DP_j , \mathcal{B} simulates *Hash*(H , ($MSK || ID_{DP_j}$)) query to get K_j and *Hash*(h_2 , ($ID_A || \{M_{\alpha_i}\}_{i=1}^n || K_j$)) query to get w_j , and returns $\langle ID_{DP_j}, ID_\alpha, \{M_{\alpha_i}\}_{i=1}^n, w_j \rangle$.



- **Send_{DB}** ($\langle ID_\alpha, k \rangle$): \mathcal{B} simulates **Hash**-query for $(H, (ID_\alpha || MPK_k))$. \mathcal{B} finds (k, t_k, MRK_k) in list L_{key} , and computes $K_{\alpha_k} = MRK_k \cdot H(ID_\alpha || MPK_k)$. \mathcal{B} simulates **Hash**-query for $(h_1, (ID_\alpha || t_k || MPK_k))$ to get C_{α_k} , and computes $P_{\alpha_k} = PUF(C_{\alpha_k})$, $PW_{\alpha_k} = H(C_{\alpha_k} || P_{\alpha_k})$, and $Q_{\alpha_k} = K_{\alpha_k} + PW_{\alpha_k}$. \mathcal{B} simulates **Hash**-query for $(h_1, (ID_\alpha || MPK_k || C_{\alpha_k} || K_{\alpha_k}))$ to get d_{α_k} , lets $M_{\alpha_k} = (C_{\alpha_k}, Q_{\alpha_k}, d_{\alpha_k})$, and returns $\langle (k, MPK_k), ID_\alpha, M_{\alpha_k} \rangle$.
- **Send_{IoT}** ($\Pi_\alpha^S, \langle ID_\beta, k, M_{\alpha_k} = (C_{\alpha_k}, Q_{\alpha_k}, d_{\alpha_k}), MPK_k \rangle$): \mathcal{B} computes $P_{\alpha_k} = PUF(C_{\alpha_k})$, $PW_{\alpha_k} = H(C_{\alpha_k} || P_{\alpha_k})$, $K_{\alpha_k} = Q_{\alpha_k} - PW_{\alpha_k}$. \mathcal{B} verifies if $d_{\alpha_k} = h_1(ID_\alpha || MPK_k || C_{\alpha_k} || K_{\alpha_k})$, and aborts the session if it is not. Otherwise, \mathcal{B} computes $y_1 = \hat{e}(K_{\alpha_k}, H(ID_\beta || MPK_k))$, $v_A = h_2(ID_\alpha || ID_\beta || N_\alpha || y_1)$, and returns $\langle N_\alpha, v_A \rangle$.
- **Execute** (α, β): If it is asked for l th session, \mathcal{B} lets $N_\alpha = A$ and $N_\beta = B$. Otherwise, \mathcal{B} randomly generates $n_\alpha, n_\beta \in_R Z_q^*$, and computes $N_\alpha = n_\alpha \cdot P$ and $N_\beta = n_\beta \cdot P$. \mathcal{B} then computes $P_{\alpha_k} = PUF(C_{\alpha_k})$, $PW_{\alpha_k} = H(C_{\alpha_k} || P_{\alpha_k})$, $K_{\alpha_k} = Q_{\alpha_k} - PW_{\alpha_k}$, $y_1 = \hat{e}(K_{\alpha_k}, H(ID_\beta || MPK_k))$, $v_\alpha = h_2(ID_\alpha || ID_\beta || N_\alpha || y_1)$, and $v_\beta = h_2(ID_\beta || ID_\alpha || N_\beta || y_1)$. \mathcal{B} lets $Tag = (ID_A || ID_B || N_A || N_B || y_1)$, and returns $\langle k, ID_\alpha, M_{\alpha_k} = (C_{\alpha_k}, Q_{\alpha_k}, d_{\alpha_k}), MPK_k, ID_\beta, M_{\beta_k} = (C_{\beta_k}, Q_{\beta_k}, d_{\beta_k}), N_\alpha, v_\alpha, N_\beta, v_\beta \rangle$.
- **Reveal** (Π_α^S): \mathcal{B} runs the simulation of above **Send** queries, and returns SK if the



instance Π_α^S has accepted the session and responds a null value otherwise.

- **Corrupt** (α, M): \mathcal{B} computes and returns $PUF(M)$.
- **Test_{SK}** (Π_α^S): \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns the session key SK of the instance Π_α^S if $b=1$, and returns a random value if $b=0$.

If \mathcal{A} can break the session key secrecy or the perfect forward secrecy of the proposed protocol, abP should appear in the L_H list. \mathcal{B} uses Tag to find $(h_1, M = (Tag||y_2), r_1)$ in L_H , where $y_2=abP$. The success probability of \mathcal{B} solving the ECCDH problem depends on the event that \mathcal{A} asks the *Test* query in the l th session. Let q_n denotes the number of sessions, then the probability of \mathcal{A} asking the *Test* query in the l th session is $1/q_n$. If \mathcal{A} correctly guesses b in the *Test* query with a non-negligible advantage ε , \mathcal{B} can solve the ECCDH problem with a non-negligible advantage ε/q_n . By Assumption 3, no polynomial-time algorithm can solve the ECCDH problem with a non-negligible advantage. It is a contradiction. Thus, no polynomial-time adversary can break the session key secrecy or the perfect forward secrecy of Protocol 3 with a non-negligible advantage by Definition 6. \square

Theorem 7 *The proposed Protocol 3 achieves existential unforgeability of IoT nodes on the Decisional Bilinear Diffie-Hellman (DBDH) assumption and hash function*



assumption.

Proof: Suppose that \mathcal{A} is a polynomial-time adversary, who can forge an IoT node in Protocol 3 with a non-negligible advantage ε . Then we can construct an algorithm \mathcal{B} to solve the DBDH problem with a non-negligible advantage by using \mathcal{A} 's ability of breaking the protocol. Suppose that \mathcal{B} is given an instance $(G_1, G_2, P, \hat{e}, q, A = aP, B = bP, C = cP, g = \hat{e}(P, P)^d)$ of DBDH problem, and \mathcal{B} 's goal is to determine if $g = \hat{e}(P, P)^{abc}$. \mathcal{B} permeates the DBDH problem into queries, and chooses target IoT nodes U and V . \mathcal{B} maintains a hash list L_H to avoid the collision and ensure the identical responses. \mathcal{B} chooses a constant $n \in \mathbb{Z}$. For each $i \in \{1, 2, \dots, n\}$, \mathcal{B} randomly generates $a_i, t_i, r_i \in_R \mathbb{Z}_q^*$, computes $MPK_i = a_i \cdot A$, appends $(i, t_i, (MPK_i, a_i))$ to list L_{key} , and appends $(H, (ID_U || MPK_i), r_i \cdot B, r_i)$ and $(H, (ID_V || MPK_i), r_i \cdot C, r_i)$ to list L_H . \mathcal{B} sets the public parameters $\mathcal{Pub} = \{p, G_1, G_2, P, \hat{e}, \{(i, MPK_i)\}_{i=1}^n\}$.

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{SK}*, and no *Corrupt* (α) or *Reveal* (Π_α^S) query is asked before *Test_{SK}* (Π_α^S) query. \mathcal{B} simulates the oracle queries of \mathcal{A} as follows:



- The *Hash*, *Extract*, *Reveal*, *Corrupt*, and *Test* queries are identical to those queries

in the proof of Theorem 6.

- *Send_{DB}* ($\langle ID_\alpha, k \rangle$): \mathcal{B} finds $(k, t_k, (MPK_k, a_k))$ in list L_{key} . If $\alpha = U$ or $\alpha = V$, \mathcal{B}

randomly chooses $K_{\alpha_k} \in_R G_1$. If $\alpha \neq U$ and $\alpha \neq V$, \mathcal{B} simulates *Hash*-query for $(H,$

$ID_\alpha || MPK_k)$, finds $(H, (ID_\alpha || MPK_k), H(ID_\alpha || MPK_k), r_1)$ in L_H to get r_1 , and

computes $K_{\alpha_k} = a_k \cdot r_1 \cdot A$, where $a_k \cdot r_1 \cdot A = MRK_k \cdot H(ID_\alpha || MPK_k) = K_{\alpha_k}$. \mathcal{B}

then simulates *Hash*-query for $(h_1, (ID_\alpha || t_k || MPK_k))$ to get C_{α_k} , and computes P_{α_k}

$= PUF(C_{\alpha_k})$, $PW_{\alpha_k} = H(C_{\alpha_k} || P_{\alpha_k})$, and $Q_{\alpha_k} = K_{\alpha_k} + PW_{\alpha_k}$. \mathcal{B} simulates

Hash-query for $(h_1, (ID_\alpha || MPK_k || C_{\alpha_k} || K_{\alpha_k}))$ to get d_{α_k} , lets $M_{\alpha_k} = (C_{\alpha_k}, Q_{\alpha_k},$

$d_{\alpha_k})$, and returns $\langle (k, MPK_k), ID_\alpha, M_{\alpha_k} \rangle$.

- *Send_{IoT}* ($\Pi_\alpha^S, \langle ID_\beta, M_{\alpha_k} = (C_{\alpha_k}, Q_{\alpha_k}, d_{\alpha_k}), MPK_k \rangle$): \mathcal{B} computes $P_{\alpha_k} =$

$PUF(C_{\alpha_k})$, simulates *Hash*-query for $(H, (C_{\alpha_k} || P_{\alpha_k}))$ to get PW_{α_k} , and computes

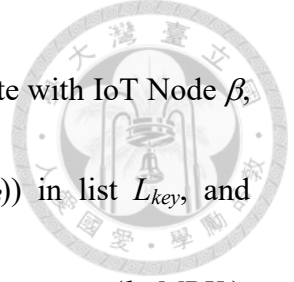
$K_{\alpha_k} = Q_{\alpha_k} - PW_{\alpha_k}$. \mathcal{B} simulates *Hash*-query for $(h_1, (ID_\alpha || MPK_k || C_{\alpha_k} || K_{\alpha_k}))$,

verifies if $d_{\alpha_k} = h_1(ID_\alpha || MPK_k || C_{\alpha_k} || K_{\alpha_k})$, and returns NULL if it is not true.

Otherwise, \mathcal{B} randomly generates $n_\alpha \in_R Z_q^*$, computes $N_\alpha = n_\alpha \cdot P$ and $y_1 = \hat{e}(K_{\alpha_k},$

$H(ID_\beta || MPK_k)$), simulates *Hash*-query for $(h_2, (ID_\alpha || ID_\beta || N_\alpha || y_1))$ to get v_α , and

returns $\langle ID_\alpha, v_\alpha \rangle$.




• **Execute** (Π_α^S): When it is asked for IoT Node α , who authenticate with IoT Node β , \mathcal{B} randomly selects $k \in \{1, 2, \dots, n\}$, finds $(k, t_k, (MPK_k, a_\beta))$ in list L_{key} , and simulates **Send**_{DB} ($\langle ID_\alpha, k \rangle$) and **Send**_{DB} ($\langle ID_\beta, k \rangle$) queries to get $\langle (k, MPK_k), ID_\alpha, M_{\alpha_k} \rangle$ and $\langle (k, MPK_k), ID_\beta, M_{\beta_k} \rangle$.

If $\alpha = U$ and $\beta = V$, \mathcal{B} finds $(H, (ID_U || MPK_k), r_k \cdot B, r_k)$ and $(H, (ID_V || MPK_k), r_k \cdot C, r_k)$ in L_H , and computes $y_1 = g^{a_k \cdot r_k \cdot r_k}$. (Note that $y_1 = \hat{e}(K_{U_k}, H(ID_V || MPK_k)) = \hat{e}(MRK_k \cdot H(ID_U || MPK_k), r_k \cdot C) = \hat{e}(a_k \cdot a \cdot r_k \cdot B, r_k \cdot C) = \hat{e}(a_k \cdot r_k \cdot P, r_k \cdot P)^{abc} = g^{a_k \cdot r_k \cdot r_k}$).

If $\alpha \neq U$, \mathcal{B} finds $(H, (ID_\alpha || MPK_k), H(ID_\alpha || MPK_k), r)$ and $(H, (ID_\beta || MPK_k), H(ID_\beta || MPK_k), r^*)$ in L_H , and computes $K_{\alpha_k} = a_k \cdot r \cdot A$ and $y_1 = \hat{e}(K_{\alpha_k}, H(ID_\beta || MPK_k))$. (Note that $K_{\alpha_k} = MRK_k \cdot H(ID_\alpha || MPK_k) = a_k \cdot a \cdot r \cdot P = a_k \cdot r \cdot A$).

If $\beta \neq V$, \mathcal{B} finds $(H, (ID_\alpha || MPK_k), H(ID_\alpha || MPK_k), r)$ and $(H, (ID_\beta || MPK_k), H(ID_\beta || MPK_k), r^*)$ in L_H , and computes $K_{\beta_k} = a_k \cdot r^* \cdot A$ and $y_1 = \hat{e}(K_{\beta_k}, H(ID_\alpha || MPK_k))$. (Note that $K_{\beta_k} = MRK_k \cdot H(ID_\beta || MPK_k) = a_k \cdot a \cdot r^* \cdot P = a_k \cdot r^* \cdot A$).

\mathcal{B} then randomly generates $n_\alpha, n_\beta \in_R Z_q^*$, computes $N_\alpha = n_\alpha \cdot P$ and $N_\beta = n_\beta \cdot P$, simulates **Hash**-query for $(h_2, (ID_\alpha || ID_\beta || N_\alpha || y_1))$ to get v_α and for $(h_2, (ID_\beta || ID_\alpha || N_\beta || y_1))$ to get v_β , and returns $\{\langle ID_\alpha, k \rangle, \langle ID_\beta, k \rangle, \langle ID_\alpha, M_{\alpha_k}, MPK_k \rangle, \langle ID_\beta, M_{\beta_k}, MPK_k \rangle, \langle N_\alpha, v_\alpha \rangle, \langle N_\beta, v_\beta \rangle\}$.



If \mathcal{A} answers $b = 1$ to *Test* query, \mathcal{B} answers $g = \hat{e}(P, P)^{abc}$ to the DBDH problem. If \mathcal{A} answers $b \neq 1$ to *Test* query, \mathcal{B} answers $g \neq \hat{e}(P, P)^{abc}$ to the DBDH problem. The success probability of \mathcal{B} solving the DBDH problem depends on the event that \mathcal{A} asks the *Test* query for IoT node U and V . Let q_I denotes the number of IoT nodes, then the probability of \mathcal{A} asking *Execute* and *Test* query for IoT node U (who authenticates with IoT node V) or and IoT node V (who authenticates with IoT node U) is $2/(q_I \cdot (q_I - 1))$. If \mathcal{A} correctly guesses b in the *Test* query with a non-negligible advantage ε , \mathcal{B} can solve the ECCDH problem with a non-negligible advantage $2\varepsilon/(q_I \cdot (q_I - 1))$. By Assumption 4, no polynomial-time algorithm can solve the DBDH problem with a non-negligible advantage. It is a contradiction; hence, Protocol 3 achieves existential unforgeability by Definition 7. \square



Chapter 8

CAKE: Compatible Authentication and Key Exchange Protocol – Taking a Smart City for Example in 5G Networks

In this section, we take a smart city for example to propose a compatible authentication and key exchange (CAKE) protocol and extend it to an anonymous CAKE (ACAKE). The benefits of 5G technology, such as high speed, high capacity, extremely low latency, and good quality of service of servers, realize the rapid and frequent authentication and communication between members in the communication networks. In the proposed CAKE/ACAKE protocol, there are four kinds of components: one registration center (RC), numerous service providers (Server), plenty of natural persons (NP), and a large number of IoT devices. Figure 14 depicts the components of a smart city. RC is a trusted third party, which is responsible for system initialization and performing the registration, revocation, and key updating for members. Servers can have various kinds of functions, such as providing services for NPs, monitor and controlling IoT devices, gathering and analyzing data from IoT devices, etc.

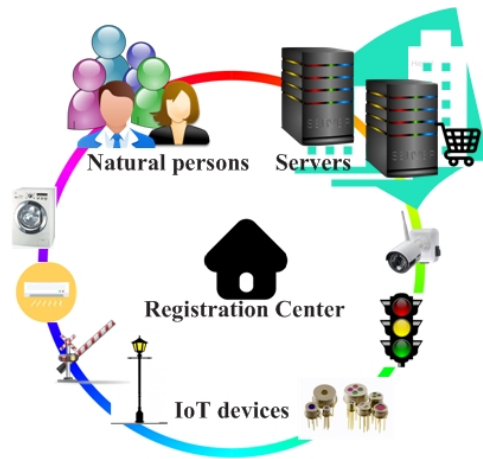


Figure 14. Components of a smart city

A server may be a cashier, an Automated Teller Machine (ATM), a housekeeper, a web service provider, a cloud service provider, a medical service provider, a library, a school, a company, a bank, a city control center, a government department, etc. An IoT device is assumed to be embedded Physical Unclonable Functions (PUFs); it may be a sensor, an electrical appliance, a traffic signal, a street light, a surveillance camera, a vehicular, etc.

8.1 The Framework of a CAKE/ACAKE Protocol for a Smart City

Here, we present the framework of a CAKE/ACAKE protocol, which is depicted in Figure 15. The ACAKE protocol is an extended CAKE protocol, which additionally provides an anonymity option for natural persons to protect their privacy. The framework of a CAKE/ACAKE protocol consists of four phases: initialization phase,

registration phase, online update phase, and CAKE/ACAKE phase.



In both CAKE and ACAKE protocols, every member has to register to the registration center (RC) in a secure environment when it joins the system. In the CAKE phase, each member can mutually authenticate with the other member and establish a secure session key through a public channel without the help of any third party, even if they are different kinds of members. In the ACAKE protocol, NP has an anonymity option to run the ACAKE phase with the other NP, Server or IoT device; meanwhile, the real identity of NP would not be revealed in the transmitted channel, and the relation between NP's sessions is untraceable. Temporary Mobile Subscriber Identity (TMSI) in mobility management is randomly assigned by the visitor location register (VLR) to every mobile in the area and the moment it is switched on, on the contrary the anonymous identity in our ACAKE protocol is decided by each member off-line.

For the member revocation problem, the proposed CAKE/ACAKE protocol employs a key update method to refresh the current secret key for current time period. In such case, every member must run the online update phase to update its secret key. In particular, members need not run the online update phase in each time period, and the member can only run the online update phase in the specific time period.

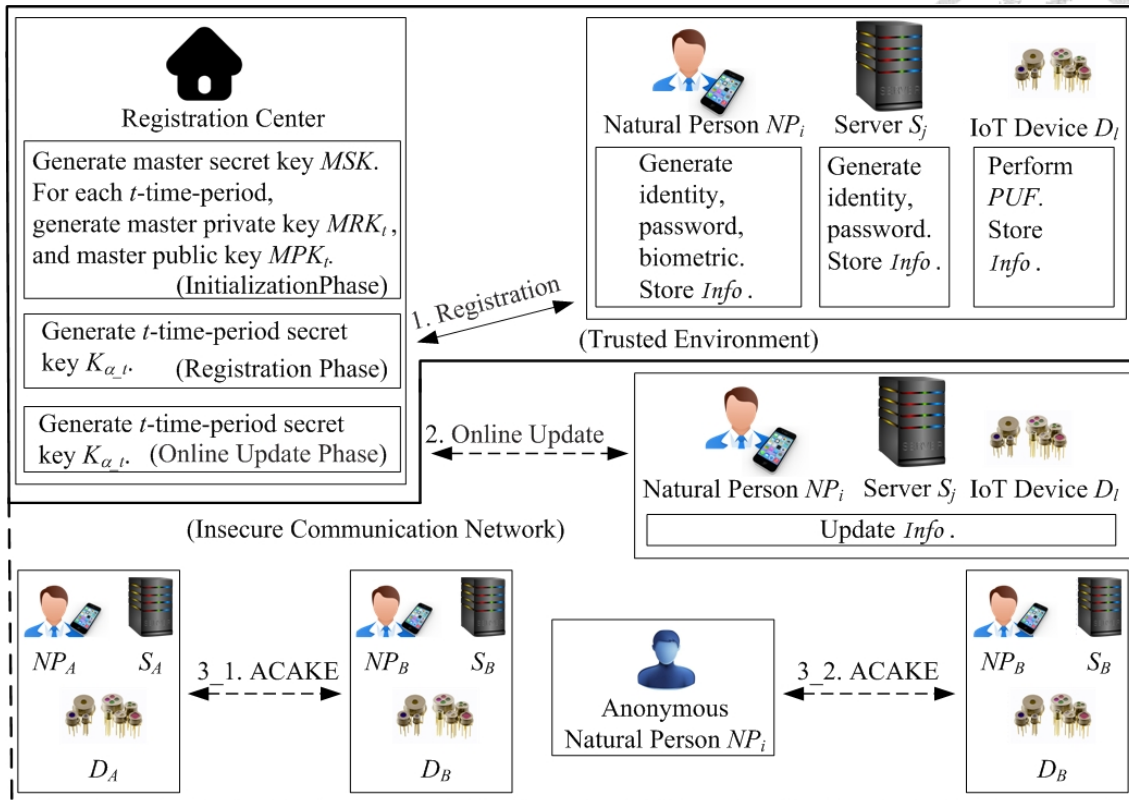


Figure 15. The framework of the proposed CAKE/ACAKE protocol

Before a member executing the compatible authentication and key exchange with another member, both of them have to update their secret keys with the help of RC through an open channel in the current time period. Once a member is revoked or disabled, it will not be able to receive the new updated keys in the next time period; that is, the disabled member would be revoked in the next time period. The length of a time period, which is decided by the supervisor, may be hours, days, weeks, or months. If a member wants to revoke the previous authorization immediately in the current time period, RC revokes the member by the certificate revocation list (CRL).



8.2 The Proposed Protocol (Protocol 4: CAKE)

The proposed CAKE protocol is comprised of four phases: initialization, registration, online update, compatible authentication and key exchange (CAKE), and password and biometric change phases. We expatiate on these four phases in the following.

8.2.1 Initialization Phase

RC chooses a large prime q , and defines a finite field F_q and an elliptic curve $E(F_q)$ by an equation of the form $y^2 = x^3 + mx + n$, where $m, n \in F_q$ satisfy $4m^3 + 27n^2 \neq 0 \pmod{q}$. RC produces a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$, where G_1 is a subgroup of the additive cyclic group of points on E with order q , and G_2 is a multiplicative cyclic subgroup over F_q with order q . RC picks a generator point P in G_1 , and chooses four distinct one-way hash functions, namely, $H_1: \{0, 1\}^* \rightarrow G_1$, $H_2: \{0, 1\}^* \rightarrow G_1$, $h_1: \{0, 1\}^* \rightarrow \{0, 1\}^l$, and $h_2: \{0, 1\}^* \rightarrow \{0, 1\}^l$, where l is the fix length of output. RC also generates the master secret key $MSK \in Z_p^*$. For each t -time-period, RC randomly generates a seed $s_t \in Z_q^*$, computes the master private key $MRK_t = h_1(MSK || s_t)$ and master public key $MPK_t = MRK_t \cdot P$, and appends (t, s_t, MRK_t, MPK_t) to list L_{seed} and keeps it secret. RC sets the public parameters $Pub = \{q, G_1, G_2, P, \hat{e}, H_1, H_2, h_1, h_2\}$.



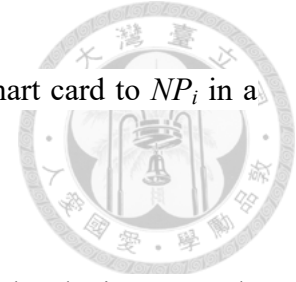
8.2.2 Registration Phase

[Natural Person]

When a natural person NP_i with a portable device wants to join to the system, the NP_i interacts with the RC as the following steps, which are depicted in Figure 16, in a trusted environment.

Step 1: NP_i chooses his/her identity ID_i and password pw_i . Note that NP_i can choose either the real identity or a pseudonym to be ID_i , where the pseudonym enables NP_i to achieve strong anonymity. NP_i imprints his/her biometric to the device to get B_i . The device performs the generation procedure Gen of fuzzy extractor on B_i to get secret string SS_i and helper string HLP_i , i.e. $(SS_i, HLP_i) = Gen(B_i)$. The device computes the regenerating password $RPW_i = H_2(ID_i || pw_i || SS_i)$, and sends $\langle ID_i, RPW_i, \text{"NP"} \rangle$ to RC through a trusted channel.

Step 2: RC checks the current time period index t , and verifies the validity of NP_i . If RC wants to authorize NP_i , RC computes NP_i 's secret key $K_{i,t}$ by using RC's secret key MRK_t , encrypts the secret key $K_{i,t}$ by the regenerating password RPW_i , i.e. computes $K_{i,t} = MRK_t \cdot H_1(ID_i || MPK_t)$ and $Q_{i,t} = K_{i,t} + RPW_i$. RC then computes the confirmation parameter $c_i = H_2(ID_j || RPW_i)$, lets $Info \leftarrow \{ID_i, t, Q_{i,t}, c_i$,



MPK_t }, issues a new smart card with $Info$, and sends the smart card to NP_i in a trusted environment.

Step 3: NP_i inserts the receiving smart card into the device, and the device appends HLP_i to the smart card.

[Server]

When a new server S_j wants to join the system, it has to register to RC and perform the following steps, which are depicted in Figure 17, in a trusted environment.

Step 1: S_j chooses its identity ID_j and password pw_j , and gives $\langle ID_j, \text{“Server”} \rangle$ to RC.

Step 2: RC checks the current time period index t , and verifies the validity of S_j . If RC wants to authorize S_j , RC computes S_j 's secret key K_{j_t} by using RC's secret key MRK_t , i.e. $K_{j_t} = MRK_t \cdot H_1(ID_j || MPK_t)$, and sends $\langle t, MPK_t, K_{j_t} \rangle$ to S_j .

Step 3: After receiving $\langle K_{j_t} \rangle$, S_j computes its regenerating password $RPW_j = H_2(ID_j || pw_j)$, $Q_{j_t} = K_{j_t} + RPW_j$, and $c_j = H_2(ID_j || RPW_j)$. S_j then lets $Info \leftarrow \{ID_j, t, Q_{j_t}, c_j, MPK_t\}$ and stores $Info$.

[IoT Device]

The following steps, which are depicted in Figure 18, are performed in a trusted environment before adding an IoT device D_l into the system.

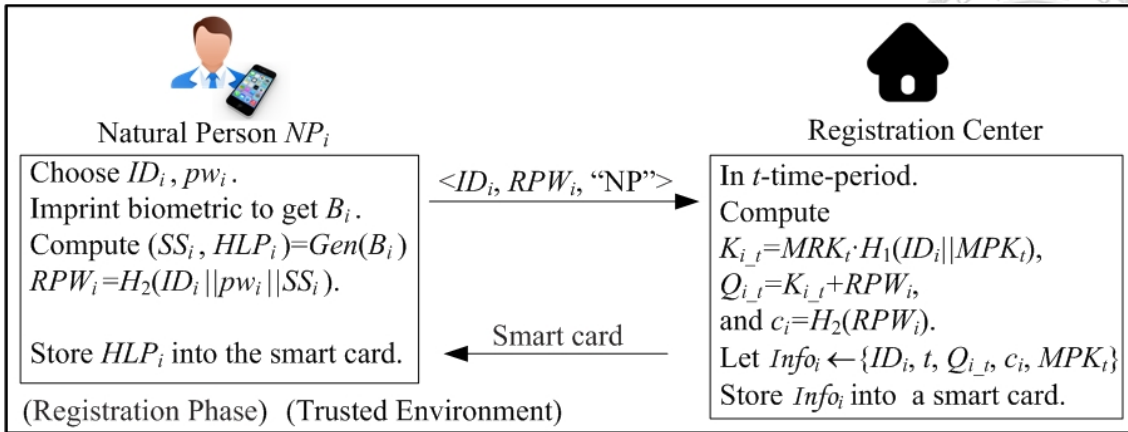


Figure 16. Registration phase for a natural person in our Protocol 4

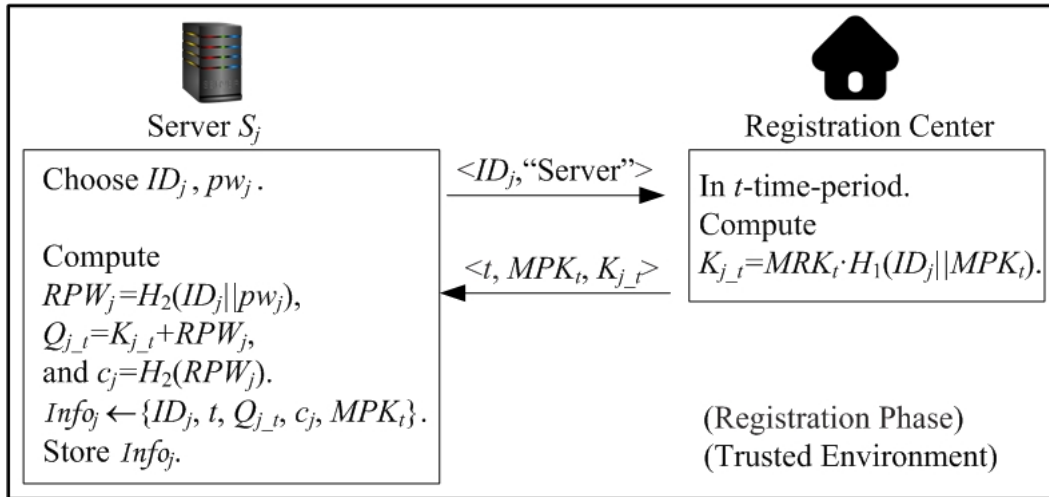


Figure 17. Registration phase for a server in our Protocol 4

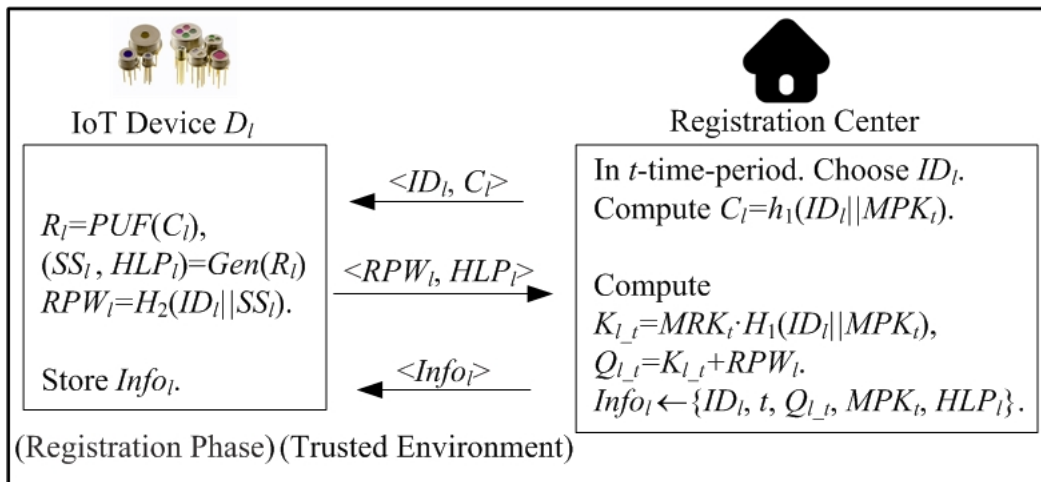
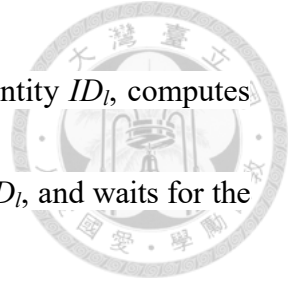


Figure 18. Registration phase for an IoT device in our Protocol 4



Step 1: RC checks the current time period index t , chooses D_l 's identity ID_l , computes the challenge string $C_l = h_1(ID_l || MPK_t)$, gives $\langle ID_l, C_l \rangle$ to D_l , and waits for the response.

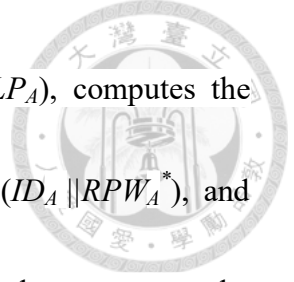
Step 2: D_l performs PUF on the challenge string C_l to get the output response R_l , performs the generation procedure Gen of fuzzy extractor with BCH encoder [110] on R_l to get secret string SS_l and helper string HLP_l , i.e. $R_l = PUF(C_l)$ and $(SS_l, HLP_l) = Gen(R_l)$. D_l computes the regenerating password $RPW_l = H_2(ID_l || SS_l)$, and sends $\langle RPW_l, HLP_l \rangle$ to RC.

Step 3: RC computes D_l 's secret key $K_{l,t}$ by using RC's secret key MRK_t and encrypts the secret key $K_{l,t}$ by the regenerating password RPW_l , i.e. $K_{l,t} = MRK_t \cdot H_1(ID_l || MPK_t)$ and $Q_{l,t} = K_{l,t} + RPW_l$. RC then lets $Info \leftarrow \{ID_l, t, Q_{l,t}, MPK_t, HLP_l\}$ and sends $\langle Info_l \rangle$ to D_l . D_l then stores $Info_l$.

For convenient, we define an algorithm GETKEY(A) in online update, CAKE, and ACAKE phases to be a procedure of getting the secret key of member A.

[GETKEY(A) Algorithm]

1) If A is a NP, A inputs ID_A and pW_A^* , imprints biometric to get B_A , the smart card performs the deterministic reproduction procedure Rep on B_A^* to recover SS_A^* from



the corresponding helper string HLP_A , i.e. $SS_A^* = Rep(B_A^*, HLP_A)$, computes the regenerating password $RPW_A^* = H_2(ID_A || pw_A^* || SS_A^*)$, $c_A^* = H_2(ID_A || RPW_A^*)$, and checks if $c_A^* = c_A$. If so, then the validity of A is confirmed, and then computes the secret key $K_A = Q_{A_t} - RPW_A^*$ and outputs K_A . Otherwise, the smart card returns a random number.

2) If A is a server, A inputs ID_A and pw_A^* , computes the regenerating password $RPW_A^* = H_2(ID_A || pw_A^*)$, derives its secret key K_A from Q_{A_t} by using RPW_A^* , i.e. $K_{A_t} = Q_{A_t} - RPW_A^*$, and outputs K_A .

3) If A is an IoT, A computes the challenge string $C_A = h_1(ID_A || MPK)$, performs PUF on C_A to get the response string R_A , performs deterministic reproduction procedure Rep on R_A to recover SS_A from the corresponding helper string HLP_A , i.e. $R_A = PUF(C_A)$ and $SS_A = Rep(R_A, HLP_A)$, computes the regenerating password $RPW_A = H_2(ID_A || SS_A)$, derives its secret key K_{A_t} from Q_{A_t} by using RPW_A^* , i.e. $K_{A_t} = Q_{A_t} - RPW_A$, and outputs K_{A_t} .

8.2.3 Online Update Phase

Each NP, server, and IoT device needs to perform the following steps through public channels to online update its secret key before it authenticates with another



member in the current t -time-period. This phase is depicted in Figure 19.

Step 1: The entity, say Entity A , executes GETKEY(A) algorithm to get A 's secret key

K_{A_t} , generates a random number $n \in_R Z_q^*$, computes $N_1 = n \cdot P$, $R = n \cdot MPK_t$, $N_2 =$

$h_2(ID_A || t || N_1 || R || K_{A_t})$, and sends $\langle ID_A, t, N_1, N_2, \text{"Update"} \rangle$ to RC through an

untrusted channel.

Step 2: RC checks the index t^* of current time period, verifies the validity of ID_A . RC

computes A 's newer secret key $K_{A_{t^*}} = MRK_{t^*} \cdot H_1(ID_A || MPK_{t^*})$, A 's older secret

key $K_{A_t} = MRK_t \cdot H_1(ID_A || MPK_t)$, and $R = MRK_t \cdot N_1$. RC then verifies if $N_2 =$

$h_2(ID_A || t || N_1 || R || K_{A_t})$. If not, RC aborts it. Otherwise, RC computes $U_{A_{t^*}} = K_{A_{t^*}}$

$+ H_2(K_{A_t} || R)$, $V_{A_{t^*}} = MPK_{t^*} + H_2(K_{A_t} || N_1)$, and $d_{A_t} = h_2(MPK_{t^*} || K_{A_{t^*}} || K_{A_t} || R)$,

and sends $\langle ID_A, t^*, U_{A_{t^*}}, V_{A_{t^*}}, d_{A_{t^*}} \rangle$ to A .

Step 3: A derives A 's newer secret key secret key $K_{A_{t^*}}$ from $U_{A_{t^*}}$ and derives the newer

master public key MPK_{t^*} by using A 's older secret key K_{A_t} , i.e. $K_{A_{t^*}} = U_{A_{t^*}}$

$- H_2(K_{A_t} || R)$ and $MPK_{t^*} = V_{A_{t^*}} - H_2(K_{A_t} || N_1)$. A then computes $Q = Q_A - K_{A_t}$

$+ K_{A_{t^*}}$, and verifies if $d_{A_{t^*}} = h_2(MPK_{t^*} || K_{A_{t^*}} || K_{A_t} || R)$. If so, A rewrites $Q_{A_t} \leftarrow Q$,

$t \leftarrow t^*$, and $MPK_t \leftarrow MPK_{t^*}$.



8.2.4 CAKE Phase

When two entities, say Entity A and Entity B , want to mutually authenticate with each other in t -time-period and create a secure session key, they perform the following steps, which are depicted in Figure 20, through a public channel. Note that the entity may be one of natural person, server, and IoT device.

Step 1: A generates a random number $n_A \in_R Z_q^*$, computes $N_A = n_A \cdot P$, and sends $\langle ID_A,$

$N_A, \text{“AKE”} \rangle$ to B through an untrusted channel.

Step 2: B executes GETKEY(B) algorithm to get B 's secret key K_{B_t} , generates a random

number $n_B \in_R Z_q^*$, computes $N_B = n_B \cdot P$, $y_1 = n_B \cdot N_A$, $y_2 = \hat{e}(K_{B_t}, H_1(ID_A || MPK_t))$,

and $v_B = h_2(ID_A || ID_B || MPK_t || y_1 || y_2)$, and sends $\langle ID_B, N_B, v_B \rangle$ to A through an

untrusted channel.

Step 3: A performs GETKEY(A) algorithm to get A 's secret key K_{A_t} , computes $y_1 =$

$n_A \cdot N_B$ and $y_2 = \hat{e}(K_{A_t}, H_1(ID_B || MPK_t))$, and verifies if $v_B = h_2(ID_A || ID_B || MPK_t$

$|| y_1 || y_2)$. If so, the authority of B is confirmed, and A computes the session key

$SK = h_1(ID_A || ID_B || N_A || N_B || y_1 || y_2)$ and $v_A = h_2(ID_A || ID_B || h_2(SK))$, and sends \langle

$v_A \rangle$ to B .

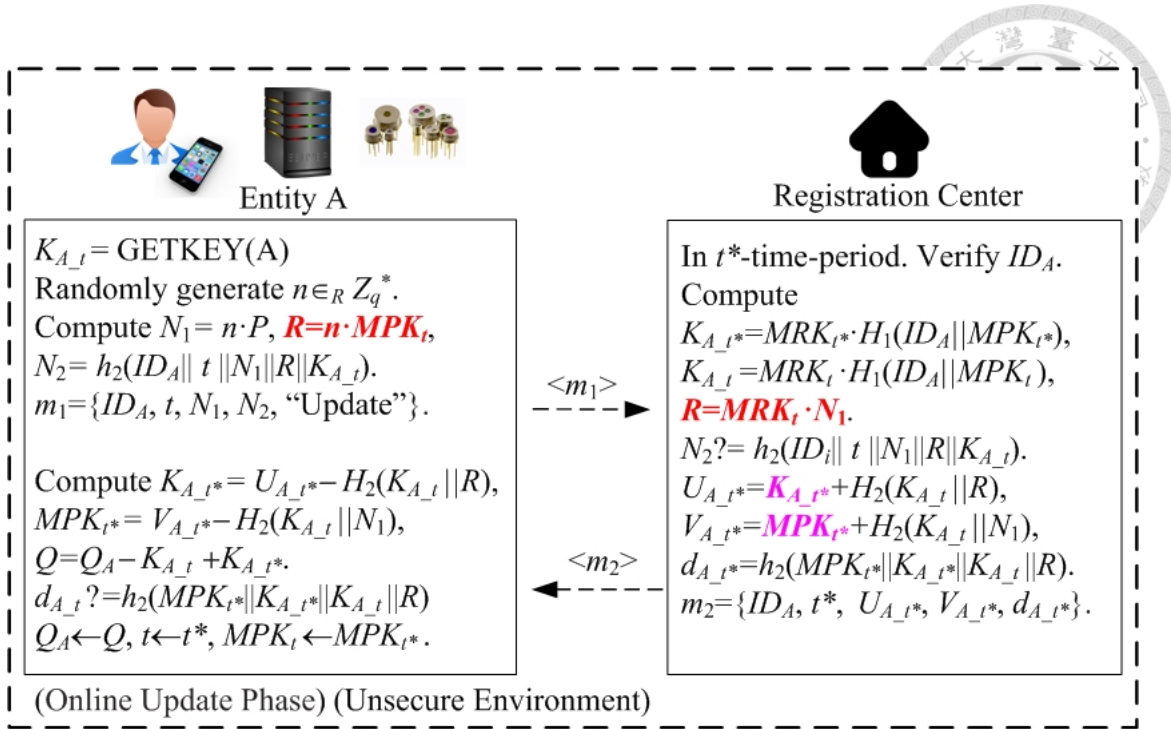


Figure 19. Online update phase of our Protocol 4

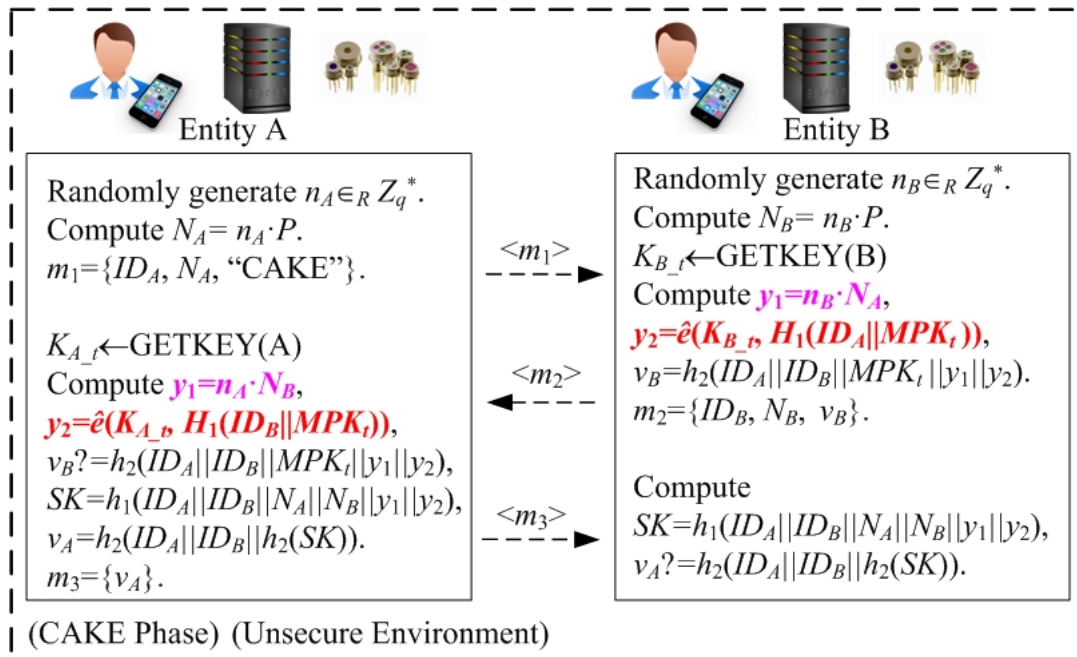


Figure 20. CAKE phase of our Protocol 4

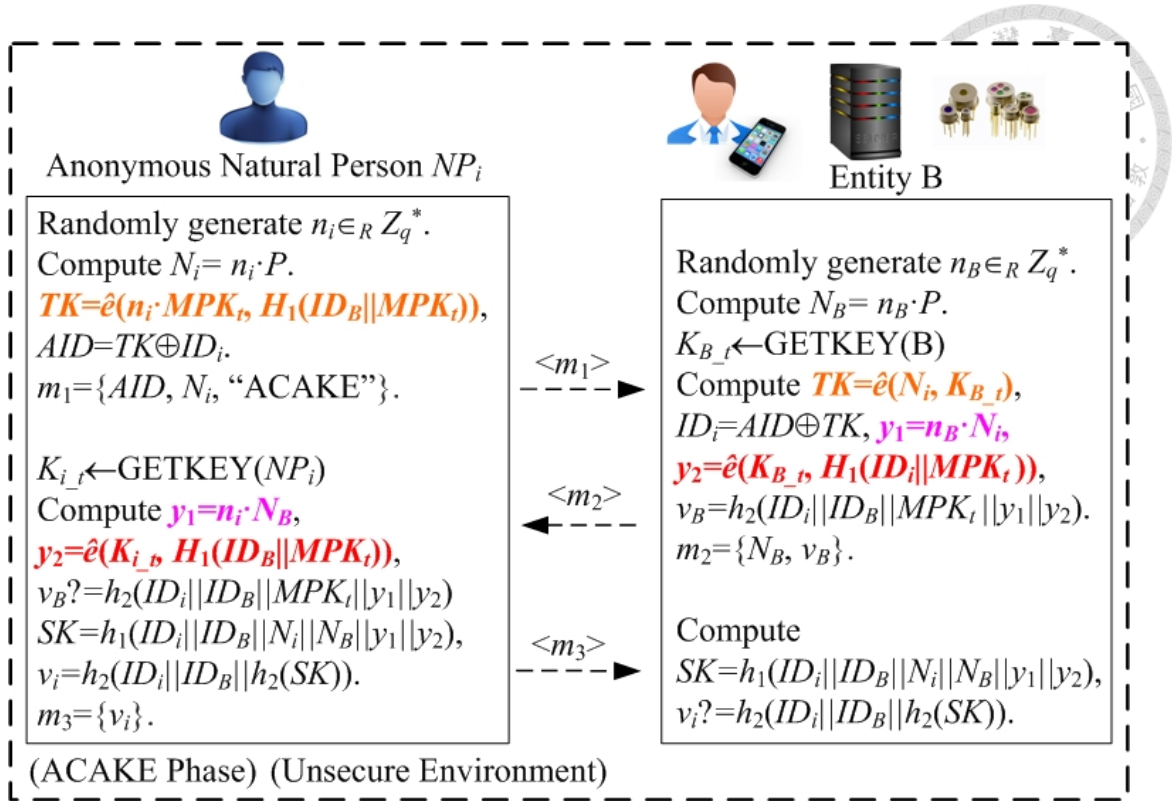


Figure 21. ACAKE phase of our Protocol 4

Step 4: B computes the session key $SK = h_1(ID_A || ID_B || N_A || N_B || y_1 || y_2)$ and verifies if v_A

$= h_2(ID_A || ID_B || h_2(SK))$. If so, the authority of A is confirmed and B accepts SK

as the session key.

Correctness: The session keys $SK = h_1(ID_A || ID_B || N_A || N_B || y_1 || y_2)$ computed by

IoT Node A and IoT Node B are identical, since the following equations hold.

$$y_1 = n_B \cdot N_A = n_B \cdot n_A \cdot P = n_A \cdot n_B \cdot P = n_A \cdot N_B.$$

$$y_2 = \hat{e}(K_{B_t}, H_1(ID_A || MPK_t)) = \hat{e}(MRK_t \cdot H_1(ID_B || MPK_t), H_1(ID_A || MPK_t))$$

$$= \hat{e}(H_1(ID_B || MPK_t), MRK_t \cdot H_1(ID_A || MPK_t)) = \hat{e}(H_1(ID_B || MPK_t), K_{A_t})$$

$$= \hat{e}(K_{A_t}, H_1(ID_B || MPK_t)).$$



8.2.5 Password and Biometric Change Phase

[Nature Person]

When a natural person NP_i wants to change the password or biometric impression, NP_i can change them on his/her own by performing the following steps.

Step 1: NP_i inputs his/her identity ID_i and password pw_i^* , imprints his/her biometric on the device to get B_i^* , and the device performs the deterministic reproduction procedure Rep on B_i^* to recover SS_i^* from the corresponding helper string HLP_i , which is stored in the smart card, i.e. $SS_i^* = Rep(B_i^*, HLP_i)$. The device computes the old regeneration password $RPW_i^* = H_2(ID_i || pw_i^* || SS_i^*)$, the old confirmation parameter $c_i^* = H_2(ID_i || RPW_i^*)$, and checks if $c_i^* = c_i$. If so, the validity of NP_i is confirmed, and the smart card continues the process. Otherwise, the smart card terminated it.

Step 2: NP_i inputs the new password PW_i^{new} , and imprints the new biometric impression B_i^{new} . NP_i 's device performs the generation procedure Gen on B_i^{new} to get secret string SS_i^{new} and helper string HLP_i^{new} , i.e. $(SS_i^{new}, HLP_i^{new}) = Gen(B_i^{new})$. The device then computes the new regenerating password $RPW_i^{new} = H_2(ID_i || pw_i^{new} || SS_i^{new})$, $Q_{i-t}^{new} = Q_{i-t} - RPW_i^* + RPW_i^{new}$, and $c_i^{new} = H_2(ID_i || RPW_i^{new})$. NP_i 's

device then replaces Q_{i_t} , c_i , and HLP_i with $Q_{i_t}^{new}$, c_i^{new} , and HLP_i^{new} in the smart card, respectively.



[Server]

When a server S_j wants to change the password, S_j can change them on his/her own by performing the following steps.

Step 1: S_j inputs the identity ID_j , the old password pw_j^* , and the new password pw_j^{new} . S_j

computes the old regenerating password $RPW_j^* = H_2(ID_j || pw_j^*)$ and the old

confirm parameter $c_j^* = H_2(ID_j || RPW_j^*)$, and checks if $c_j^* = c_j$. If so, the validity

of the old password pw_j is confirmed and S_j continues the process. Otherwise,

S_j terminated the process.

Step 2: S_j computes the new regenerating password $RPW_j^{new} = H_2(ID_j || pw_j^{new})$, $Q_{j_t}^{new} =$

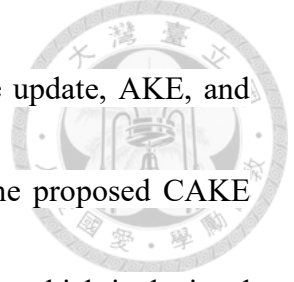
$Q_{j_t} - RPW_j^* + RPW_j^{new}$, and the new confirm parameter $c_j^{new} = H_2(ID_j || RPW_j^{new})$.

S_j then replaces Q_{j_t} and c_j with $Q_{j_t}^{new}$ and c_j^{new} , respectively.

8.3 The Proposed Protocol (Protocol 4: ACAKE)

The proposed ACAKE protocol is an extension of the CAKE protocol, which has extra anonymous authentication & key exchange (ACAKE) phase for natural persons.

We let the identity be the public key to avoid public key announcement problem.



The GETKEY algorithm and initialization, registration, online update, AKE, and password and biometric change phases are the same as those in the proposed CAKE protocol. NP can additionally perform the ACAKE phase as follows, which is depicted in Figure 21, to make AKE with another member, while keeping secrecy of his/her real identity.

8.3.1 ACAKE Phase

When a natural person, say NP_i , wants to anonymously authenticate with Entity B , whose identity is known by NP_i , they perform the following steps to mutually authenticate with each other and establish a secure common session key through a public channel.

Step 1: NP_i randomly generates $n_i \in_R Z_q^*$, computes $N_i = n_i \cdot P$, $TK = \hat{e}(n_i \cdot MPK_t,$

$H_1(ID_B || MPK_t))$, and anonymous identity $AID = TK \oplus ID_i$, and sends $\langle AID, N_i,$

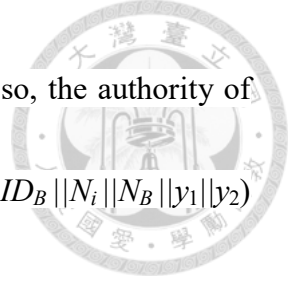
“ACAKE” \rangle to B .

Step 2: B randomly generates $n_B \in_R Z_q^*$, computes $N_B = n_B \cdot P$, $K_{B_t} \leftarrow \text{GETKEY}(B),$

$TK = \hat{e}(N_i, K_{B_t}), ID_i = AID \oplus TK, y_1 = n_B \cdot N_i, y_2 = \hat{e}(K_{B_t}, H_1(ID_i || MPK_t))$, and $v_B =$

$h_2(ID_i || ID_B || MPK_t || y_1 || y_2)$. B then sends $\langle N_B, v_B \rangle$ to NP_i .

Step 3: NP_i performs $K_{i_t} \leftarrow \text{GETKEY}(NP_i)$, computes $y_1 = n_i \cdot N_B$ and $y_2 = \hat{e}(K_{i_t}, H_1(ID_B$



$||MPK_t)$), and verifies if $v_B = h_2(ID_i || ID_B || MPK_t || y_1 || y_2)$. If so, the authority of

B is confirmed, and NP_i computes session key $SK = h_1(ID_i || ID_B || N_i || N_B || y_1 || y_2)$

and $v_i = h_2(ID_i || ID_B || h_2(SK))$, and sends $\langle v_i \rangle$ to B .

Step 4: B computes $SK = h_1(ID_i || ID_B || N_i || N_B || y_1 || y_2)$, and verifies if $v_i = h_2(ID_i || ID_B ||$

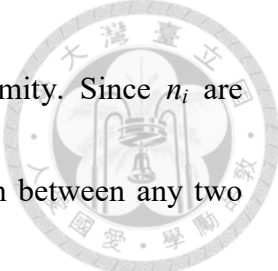
$h_2(SK))$. If so, the authority of NP_i is confirmed and SK is accepted as the session

key.

8.4 Characteristic Analysis

We analyze the properties of the proposed Protocol 4 point by point in the following.

- (P1) **Three-factor authentication**: We use biometric as the third authentication factor of nature persons, and use PUF as the third authentication factor of IoT devices.
- (P2) **Applicability of multi-server environments**: Servers are regarded as independent entities and have distinct secret keys $K_{j_t} = MRK_t \cdot H_1(ID_j || MPK_t)$ in Protocol 4, hence Protocol 4 is applicable to multi-server environments.
- (P3) **User anonymity** and (P4) **User untraceability (unlinkability)**: An anonymous nature person NP_i mask its identity ID_i by $TK = \hat{e}(n_i \cdot MPK_t, H_1(ID_B || MPK_t))$. Since only S_j and the login entity can compute TK , no third party can obtain TK to get



NP_i 's real identity ID_i , thus Protocol 4 achieves user anonymity. Since n_i are different in each session, no third party can derive the relation between any two login transmissions. Moreover, we will formal prove that Protocol 4 achieves user anonymity on DBDH assumption in Theorem 10. Moreover, a nature person can perform AAKA with another entity under a pseudonym to achieve strong anonymity if the nature person chooses the pseudonym to be his/her identity in the registration phase.

- (P5) **Perfect forward secrecy**: We will prove that Protocol 4 achieves perfect forward secrecy on ECCDH assumption in Theorem 8.
- (P6) **Member revocation**: Protocol 4 uses online update to deal with member revocation problem that the revoked member cannot execute the online update phase to get its newer secret keys in the next time period. The time period may be hours, days, or weeks, and it is decided by RC.
- (P7) **Independent authentication**: Any two valid entities in Protocol 4 can independently authenticate with each other without the help of any third party.
- (P8) **Table free**: No table needs to be stored or maintained in Protocol 4.
- (P9) **Public key announcement free**: Protocol 4 adopts the identities to be the



public keys, hence no public key needs to be announced.

- (P10) *Formal security proof*: We will give the formal proofs of Protocol 4 in Section 8.5, and the security of Protocol 4 is based on the Elliptic Curve Computational Diffie-Hellman (ECCDH) and the Decisional Bilinear Diffie-Hellman (DBDH) problems.

8.5 Security Analysis

We propose threat assumptions and construct an adversarial model for the CAKE/ACAKE protocol in this section. We then formally prove that our Protocol 4 (CAKE/ACAKE for a smart city) achieve existential session key secrecy and perfect forward secrecy under ECCDH and hash function assumptions, and achieve existential unforgeability under the DBDH and hash function assumptions in Theorem 8 and Theorem 9, respectively. We also formally prove that our Protocol 4 (ACAKE for a smart city) achieves existential anonymity under DBDH and hash function assumptions in Theorem 10.

Theorem 8 *The proposed Protocol 4 (CAKE/ACAKE for a smart city) achieve existential perfect forward secrecy and secrecy of session key on the Elliptic Curve Computational Diffie-Hellman (ECCDH) and hash function assumptions.*

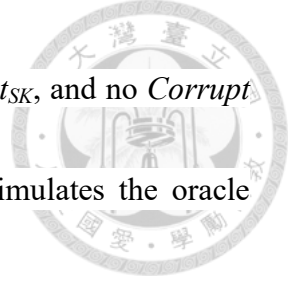


Proof: Suppose that there are one trusted third party RC, m Server, n Natural Persons (NP), and d IoT Devices in the smart city system. Each entity has a unique identity.

Suppose that there is an adversary \mathcal{A} , who can break the perfect forward secrecy or the secrecy of session key of Protocol 4 (CAKE/ACAKE for a smart city) with a non-negligible advantage in probabilistic polynomial time. Then we can construct a Challenge Algorithm \mathcal{B} to solve ECCDH problem with a non-negligible advantage by using \mathcal{A} 's ability of breaking the protocol. An instance $(G_1, G_2, P, \hat{e}, q, X = xP, Y = yP)$ of ECCDH problem is given to \mathcal{B} , and \mathcal{B} 's goal is to output xyP . \mathcal{B} permeates the ECCDH problem into the queries, which are asked by \mathcal{A} in l -session for Π_U^S and its partner Π_V^T . \mathcal{B} manages one time seed list L_{seed} , and three hash lists L_{H1}, L_{H2}, L_{h1} , and L_h , which are initially empty. \mathcal{B} generates master secret key $MSK \in Z_p^*$.

For each t -time-period, \mathcal{B} randomly generates $s_t, r_t \in Z_q^*$, sets master private key $MRK_t = h_1(MSK || s_t)$ and master public key $MPK_t = MRK_t \cdot P$, appends $((MSK || s_t), r_t, r_t \cdot P)$ to list L_{h1} and (t, s_t, MRK_t, MPK_t) to list L_{seed} . \mathcal{B} sets $Pub = \{q, G_1, G_2, P, \hat{e}\}$ to be public parameters.

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, \mathcal{A} may ask the following queries and obtain the corresponding



results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{SK}*, and no *Corrupt*

(α) or *Reveal* (Π_α^S) query is asked before *Test_{SK}* (Π_α^S) query. \mathcal{B} simulates the oracle

queries:

- **Hash** (D): \mathcal{B} checks the hash lists, and returns the corresponding value if it is in the list. Otherwise, \mathcal{B} randomly generates $w \in Z_q^*$ and $d \in \{0,1\}^l$, returns the value by the following three different types of hash queries.

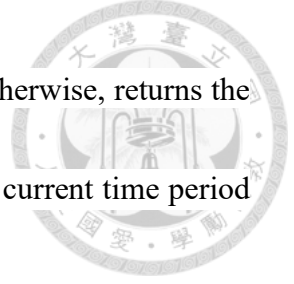
- **Hash_{H1}** (D): Compute $W = w \cdot P$, append (D, w, W) to list L_{H1} , and return W .

- **Hash_{H2}** (D): Compute $W = w \cdot P$, append (D, w, W) to list L_{H2} , and return W .

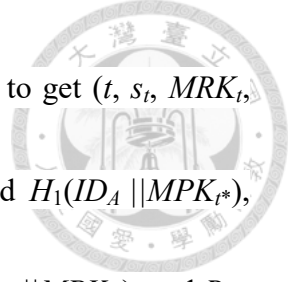
- **Hash_{h1}** (D): Append (D, d) to list L_{h1} , and return d .

- **Hash_{h2}** (D): Append (D, d) to list L_{h2} , and return d .

- **GETKEY(A)** Algorithm: If A is a NP, inputs ID_A , pw_A , and B_i , computes $SS_A = \text{Rep}(B_A, HLP_A)$, simulates **Hash_{h1}** query to get $RPW_A = H_2(ID_A || pw_A || SS_A)$, computes $K_A = Q_A - RPW_A$, and returns K_A . If A is a server, A inputs ID_A and pw_A , simulates **Hash_H** query to get $RPW_A = H_2(ID_A || pw_A)$, computes $K_{A,t} = Q_A - RPW_A$, and returns K_A . If A is an IoT, A computes $C_A = h_1(ID_A || MPK)$, performs $R_A = \text{PUF}(C_A)$, computes $SS_A = \text{Rep}(R_A, HLP_A)$, makes **Hash_{h1}** query to get $RPW_A = H_2(ID_A || SS_A)$, computes $K_A = Q_A - RPW_A$, and returns K_A .



- **Extract** (α, M): If α is one of the members, returns “Reject”. Otherwise, returns the corresponding registration parameters as follows. \mathcal{B} checks the current time period t , and list L_{seed} to get (t, s_t, MRK_t, MPK_t) .
 - $M = \langle ID_\alpha, RPW_\alpha, \text{“NP”} \rangle$: \mathcal{B} simulates **Hash**_{H1} query to get $H_1(ID_\alpha || MPK_t)$, computes $K_{\alpha_t} = MRK_t \cdot H_1(ID_\alpha || MPK_t)$ and $Q_{\alpha_t} = K_{\alpha_t} + RPW_\alpha$, lets $Info \leftarrow \{ID_\alpha, t, Q_\alpha, MPK_t\}$, and returns $Info$.
 - $M = \langle ID_\alpha, \text{“Server”} \rangle$: \mathcal{B} simulates **Hash**_{H1} query to get $H_1(ID_\alpha || MPK_t)$, computes $K_{\alpha_t} = MRK_t \cdot H_1(ID_\alpha || MPK_t)$, and returns $\langle t, MPK_t, K_{\alpha_t} \rangle$.
 - $M = \langle \text{“IoT”} \rangle$: \mathcal{B} chooses ID_α , simulates **Hash**_{h1} query to get $C_\alpha = h_1(ID_\alpha || MPK_t)$, returns $\langle C_\alpha \rangle$, and waits for the input $\langle RPW_\alpha, HLP_\alpha \rangle$. \mathcal{B} simulates **Hash**_{H1} query to get $H_1(ID_\alpha || MPK_t)$, computes $K_{\alpha_t} = MRK_t \cdot H_1(ID_\alpha || MPK_t)$ and $Q_{\alpha_t} = K_{\alpha_t} + RPW_\alpha$, and lets $Info \leftarrow \{ID_\alpha, t, Q_\alpha, MPK_t, HLP_\alpha\}$, and returns $Info$.
- **Send** (Π_β^n, M): There are three different types of **Send** query in the proposed protocols: Update, CAKE, and ACAKE. Assume that current time is in t^* -time-period. \mathcal{B} checks list L_{seed} to get $(t^*, s_{t^*}, MRK_{t^*}, MPK_{t^*})$ and responds according to the type of **Send** query.

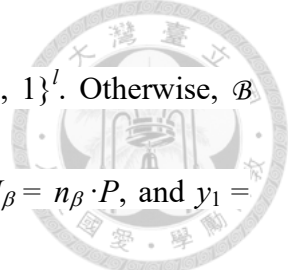


- **Send** (RC, $\langle ID_A, t, N_1, N_2, \text{"Update"} \rangle$): \mathcal{B} checks list L_{seed} to get (t, s_t, MRK_t, MPK_t) , simulates **Hash_{H1}** query to get $H_1(ID_A || MPK_t)$ and $H_1(ID_A || MPK_{t^*})$, computes $K_{A_t} = MRK_t \cdot H_1(ID_A || MPK_t)$, $K_{A_{t^*}} = MRK_{t^*} \cdot H_1(ID_A || MPK_{t^*})$, and $R = MRK_t \cdot N_1$. \mathcal{B} simulates **Hash_{H2}** query to get $H_2(K_{A_t} || R)$, computes $U_{A_{t^*}} = K_{A_{t^*}} + H_2(K_{A_t} || R)$ and $V_{A_{t^*}} = MPK_{t^*} + H_2(K_{A_t} || N_1)$, and simulates **Hash_{h2}** query to get $d_{A_{t^*}} = h_2(MPK_{t^*} || K_{A_{t^*}} || K_{A_t} || R)$. \mathcal{B} then returns $\langle ID_A, t^*, U_{A_{t^*}}, V_{A_{t^*}}, d_{A_{t^*}} \rangle$.

- **Send** (Π_β^n , $\langle ID_A, N_A, \text{"CAKE"} \rangle$): \mathcal{B} randomly generates $n_{\beta \in R} Z_q^*$, computes $N_\beta = n_\beta \cdot P$, makes GETKEY(β) algorithm to get K_{β_t} , computes $y_1 = n_\beta \cdot N_A$. \mathcal{B} simulates **Hash_{H1}** query to get $H_1(ID_A || MPK_t)$, computes $y_2 = \hat{e}(K_{\beta_t}, H_1(ID_A || MPK_t))$, simulates **Hash_{h2}** query to get $v_\beta = h_2(ID_A || ID_\beta || MPK_t || y_1 || y_2)$. \mathcal{B} returns $\langle ID_\beta, N_\beta, v_\beta \rangle$.

- **Send** (Π_β^n , $\langle AID, N_A, \text{"ACAKE"} \rangle$): \mathcal{B} randomly generates $n_{\beta \in R} Z_q^*$, computes $N_\beta = n_\beta \cdot P$, makes GETKEY(β) algorithm to get K_{β_t} , computes $TK = \hat{e}(N_A, K_{\beta_t})$, $ID_A = AID \oplus TK$, and $y_1 = n_\beta \cdot N_A$. \mathcal{B} simulates **Hash_{H1}** query to get $H_1(ID_A || MPK_t)$, computes $y_2 = \hat{e}(K_{\beta_t}, H_1(ID_A || MPK_t))$, and simulates **Hash_{h2}** query to get $v_\beta = h_2(ID_A || ID_\beta || MPK_t || y_1 || y_2)$. \mathcal{B} then returns $\langle ID_\beta, N_\beta, v_\beta \rangle$.

• **Execute** ($\Pi_\alpha^m, \Pi_\beta^n$): If $\Pi_\alpha^m = \Pi_U^S$ and $\Pi_\beta^n = \Pi_V^T$, \mathcal{B} lets $N_\alpha = X, N_\beta = Y, y_1 = NULL$,

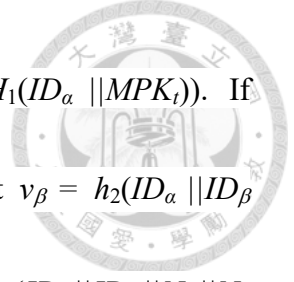


and $Tag = (ID_U || ID_V || X || Y)$, and randomly chooses $v_\beta \in_R \{0, 1\}^l$. Otherwise, \mathcal{B} randomly generates $n_\alpha, n_\beta \in_R Z_q^*$, and computes $N_\alpha = n_\alpha \cdot P$, $N_\beta = n_\beta \cdot P$, and $y_1 = n_\beta \cdot N_\alpha$. Suppose that current time is in t -time-period, \mathcal{B} checks list L_{seed} to get (t, s_t, MRK_t, MPK_t) .

- **Execute_{UPDATE}** (α, RC): \mathcal{B} chooses t^{OLD} -time-period, lets $N_1 = N_\alpha$, computes $R = n_\alpha \cdot MPK_{(t^{OLD})}$, simulates **Hash_{h2}** query to get $N_2 = h_2(ID_A || t^{OLD} || N_1 || R || K_{A_{(t^{OLD})}})$, and simulates **Send** ($RC, \langle ID_\alpha, t^{OLD}, N_1, N_2, \text{"Update"} \rangle$) query to get $\langle ID_\alpha, t, U_{\alpha_t}, V_{\alpha_t}, d_{\alpha_t} \rangle$. \mathcal{B} then returns $\langle ID_\alpha, t^{OLD}, N_1, N_2, \text{"Update"} \rangle$ and $\langle ID_\alpha, t, U_{\alpha_t}, V_{\alpha_t}, d_{\alpha_t} \rangle$.

- **Execute_{CAKE}** ($\Pi_\alpha^m, \Pi_\beta^n$): \mathcal{B} makes GETKEY(β) algorithm to get K_{β_t} , simulates **Hash_H** query to get $H(ID_\alpha || MPK_t)$, and computes $y_2 = \hat{e}(K_{\beta_t}, H_1(ID_\alpha || MPK_t))$. If $\Pi_\alpha^m \neq \Pi_U^S$ or $\Pi_\beta^n \neq \Pi_V^T$, \mathcal{B} simulates **Hash_{h2}** query to get $v_\beta = h_2(ID_\alpha || ID_\beta || MPK_t || y_1 || y_2)$, and simulates **Hash_{h1}** query to get $SK = h_1(ID_\alpha || ID_\beta || N_\alpha || N_\beta || y_1 || y_2)$. \mathcal{B} then simulates **Hash_{h2}** query to get $h_2(SK)$ and $v_\alpha = h_2(ID_\alpha || ID_\beta || h_2(SK))$, and returns $\langle ID_\alpha, N_\alpha, \text{"CAKE"} \rangle, \langle ID_\beta, N_\beta, v_\beta \rangle, \langle v_\alpha \rangle$.

- **Execute_{ACAKE}** ($\Pi_\alpha^m, \Pi_\beta^n$): \mathcal{B} computes $TK = \hat{e}(n_\alpha \cdot MPK_t, H_1(ID_\beta || MPK_t))$ and $AID = TK \oplus ID_\alpha$. \mathcal{B} makes GETKEY(β) algorithm to get K_{β_t} , simulates **Hash_{H1}**



query to get $H_1(ID_\alpha || MPK_t)$, computes $y_2 = \hat{e}(K_{\beta,t}, H_1(ID_\alpha || MPK_t))$. If

$\Pi_\alpha^m \neq \Pi_U^S$ or $\Pi_\beta^n \neq \Pi_V^T$, \mathcal{B} simulates **Hash_{h2}** query to get $v_\beta = h_2(ID_\alpha || ID_\beta$

$|| MPK_t || y_1 || y_2)$, and simulates **Hash_{h1}** query to get $SK = h_1(ID_\alpha || ID_\beta || N_\alpha || N_\beta$

$|| y_1 || y_2)$. \mathcal{B} then simulates **Hash_{h2}** query to get $v_\alpha = h_2(ID_\alpha || ID_\beta || h_2(SK))$, and

returns $\langle AID, N_\alpha, \text{"ACAKE"} \rangle, \langle ID_\beta, N_\beta, v_\beta \rangle$, and $\langle v_\alpha \rangle$.

- **Reveal** (Π_α^S): If entity α has accepted a session key, say SK , in its s -session, \mathcal{B} returns SK . Otherwise, \mathcal{B} returns "NULL".

- **Rot**: The following are the three types of Rot query. Note that at most two types of Rot query can be asked for a natural person NP_α .

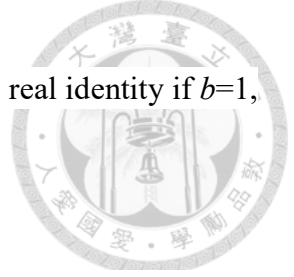
- **Rot** (NP_α, PW): \mathcal{B} returns pw_α .

- **Rot** (NP_α, BI): \mathcal{B} returns B_α .

- **Rot** (NP_α, SC): \mathcal{B} simulates **Extract_{user}** ($NP_\alpha, \langle ID_\alpha, RPW_\alpha, \text{"NP"} \rangle$) query to get $\{ID_\alpha, t, Q_\alpha, MPK_t, Pub\}$ and returns $\{ID_\alpha, t, Q_\alpha, MPK_t, Pub\}$.

- **Corrupt** (α): If entity α is one of the system members, \mathcal{B} runs the same procedures of **Extract** (α, M) query, and returns the corresponding result.

- **Test_{SK}** (Π_α^S): \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns entity α 's real session key in its s -session if $b = 1$, and returns a random value if $b = 0$.



- $Test_{ID}(\Pi_\alpha^S)$: \mathcal{B} flips an unbiased bit $b \in \{0,1\}$. \mathcal{B} returns entity α 's real identity if $b=1$, and returns a random value if $b = 0$.

If \mathcal{A} can successfully break the perfect forward secrecy or the secrecy of session key of Protocol 4, xyP should appear in the L_{h1} list. \mathcal{B} uses Tag to find $(M=(Tag||y_1||y_2), r, R)$ in L_{h1} , where $y_1 = xyP$ and $R = SK$. Then \mathcal{B} answers xyP to ECCDH problem. The successful probability of \mathcal{B} solving ECCDH problem depends on the successful probability of \mathcal{A} asking $Test_{SK}$ query in l -session for Π_U^S or its partner Π_U^S and breaking the perfect forward secrecy or the secrecy of session key of Protocol 4. The successful probability of \mathcal{B} solving ECCDH problem depends on the event that \mathcal{A} asking the $Test_{SK}$ query in the l -session for Π_U^S or its partner Π_U^S and breaking the session key secrecy or the perfect forward secrecy of Protocol 4. The probability of \mathcal{A} asking the $Test_{SK}$ query in the l -session for Π_U^S or its partner Π_U^S is $1/q_n$, where q_n is the total number of sessions. If \mathcal{A} successfully guesses b in $Test_{SK}$ query with advantage δ , which is non-negligible, then \mathcal{B} can solve ECCDH problem with advantage δ/q_n , which is also non-negligible. It contradicts ECCDH assumption. Thus, no polynomial-time adversary can break the perfect forward secrecy or the secrecy of session key of Protocol 4 with a non-negligible advantage. \square

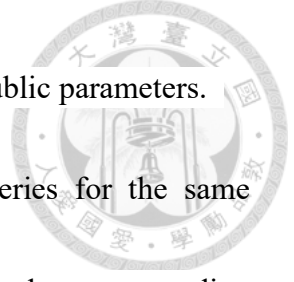


Theorem 9 *The proposed Protocol 4 (CAKE/ACAKE for a smart city) achieve existential unforgeability on Decisional Bilinear Diffie-Hellman (DBDH) and hash function assumptions.*

Proof: Suppose that there are one trusted third party RC, m Server, n Natural Persons (NP), and d IoT Devices in the smart city system, and each entity has a unique identity.

Suppose that there is an adversary \mathcal{A} , who can forge a member to cheat another member in Protocol 4 (CAKE/ACAKE for a smart city) with a non-negligible advantage in probabilistic polynomial time. Then we can construct a Challenge Algorithm \mathcal{B} to solve DBDH problem with a non-negligible advantage by using \mathcal{A} 's ability of forging a member. An instance $(G_1, G_2, P, \hat{e}, q, X = xP, Y = yP, Z = zP, g)$ of DBDH problem is given to \mathcal{B} , and \mathcal{B} 's goal is to determine if $g = \hat{e}(P, P)^{xyz}$.

\mathcal{B} guesses that \mathcal{A} tends to forge member U to cheat member V , and permeates the DBDH problem into the queries. \mathcal{A} cannot ask Corrupt query for U or for V . \mathcal{B} manages a time seed list L_{seed} , and three hash lists L_{H1} , L_{H2} , L_{h1} , and L_{h2} , which are initially empty. For each t -time-period, \mathcal{B} randomly generates $a_t, r_t \in_R Z_q^*$, computes $MPK_t = a_t \cdot X$, appends (t, a_t, MPK_t) to list L_{seed} , and appends $((ID_U || MPK_t), r_t, r_t \cdot Y)$ and $((ID_V || MPK_t), r_t, r_t \cdot Z)$ to list L_{H1} ; i.e. let $H_1(ID_U || MPK_t) = r_t \cdot Y$ and $H_1(ID_V || MPK_t) = r_t \cdot Z$. Note that



$MRK_t = a_t \cdot x$ is unknown here. \mathcal{B} sets $\mathcal{Pub} = \{q, G_1, G_2, P, \hat{e}\}$ to be public parameters.

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{SK}*, and no *Corrupt* (α) or *Reveal* (Π_α^S) query is asked before *Test_{SK}* (Π_α^S) query. \mathcal{B} simulates the oracle queries:

- The *Hash*, *Reveal*, *Rot*, *Corrupt*, and *Test* queries are identical to those in the proof of Theorem 8.

- *Extract* (α, M): If α is one of the members, returns “Reject”. Otherwise, returns the corresponding registration parameters as follows. \mathcal{B} checks the current time period t and list L_{seed} to get (t, a_t, MPK_t) .

- $M = \langle ID_\alpha, RPW_\alpha, \text{“NP”} \rangle$: \mathcal{B} simulates *Hash_{H1}* query for $(ID_\alpha || MPK_t)$, finds $((ID_\alpha || MPK_t), r, H_1(ID_\alpha || MPK_t))$ in L_H to get r , and computes $K_{\alpha_t} = a_t \cdot r \cdot X$.

Note that $K_{\alpha_t} = MRK_t \cdot H_1(ID_\alpha || MPK_t) = a_t \cdot r \cdot X$. \mathcal{B} computes $Q_{\alpha_t} = K_{\alpha_t} + RPW_\alpha$

lets $Info \leftarrow \{ID_\alpha, t, Q_\alpha, MPK_t\}$, and returns *Info*.

- $M = \langle ID_\alpha, \text{“Server”} \rangle$: \mathcal{B} simulates *Hash_{H1}* query for $(ID_\alpha || MPK_t)$, finds $((ID_\alpha || MPK_t), r, H_1(ID_\alpha || MPK_t))$ in L_{H1} to get r , and computes $K_{\alpha_t} = a_t \cdot r \cdot X$. \mathcal{B}



returns $\langle t, MPK_t, K_{\alpha_t} \rangle$.

- $M = \langle \text{“IoT”} \rangle$: \mathcal{B} chooses ID_α , simulates **Hash_{H1}** query to get $C_\alpha = h_1(ID_\alpha || MPK_t)$, returns $\langle C_\alpha \rangle$, and waits for the input $\langle RPW_\alpha, HLP_\alpha \rangle$. \mathcal{B} simulates **Hash_{H1}** query for $(ID_\alpha || MPK_t)$, finds $((ID_\alpha || MPK_t), r, H_1(ID_\alpha || MPK_t))$ in L_{H1} to get r , computes $K_{\alpha_t} = a_t \cdot r \cdot X$ and $Q_{\alpha_t} = K_{\alpha_t} + PW_l$, and lets $Info \leftarrow \{ ID_\alpha, t, Q_\alpha, MPK_t, HLP_\alpha \}$, and returns $Info$.

- **Send** (Π_B^n, M): There are three different types of **Send** query in the proposed protocols: Update, CAKE, and ACAKE. Assume that current time is in t -time-period. \mathcal{B} checks list L_{seed} to get (t, a_t, MPK_t) and responds according to the type of **Send** query.

- **Send** (RC, $\langle ID_\alpha, t^{OLD}, N_1, N_2, \text{“Update”} \rangle$): \mathcal{B} checks list L_{seed} to get $(t^{OLD}, a_{t^{OLD}}, MPK_t^{OLD})$, simulates **Hash_{H1}** query for $(ID_\alpha || MPK_t)$, finds $((ID_\alpha || MPK_t), r_t, H_1(ID_\alpha || MPK_t))$ and $((ID_\alpha || MPK_t^{OLD}), r_{t^{OLD}}, H_1(ID_\alpha || MPK_t^{OLD}))$ in L_{H1} to get r_t and $r_{t^{OLD}}$. If $\alpha \in \{U, V\}$, \mathcal{B} finds $((ID_A || t^{OLD} || N_1 || R || K_{\alpha_{(t^{OLD})}}), N_2)$ in list L_{h2} to get R and $K_{\alpha_{(t^{OLD})}}$, and computes $K_{\alpha_t} = a_t \cdot a_{t^{OLD}}^{-1} \cdot r_t \cdot r_{t^{OLD}}^{-1} \cdot K_{\alpha_{(t^{OLD})}}$. If $\alpha \notin \{U, V\}$, \mathcal{B} computes $K_{\alpha_{(t^{OLD})}} = a_t \cdot r_{t^{OLD}} \cdot X$ and $K_{\alpha_t} = a_t \cdot r_t \cdot X$. \mathcal{B} then simulates **Hash_{H2}** query to get $H_2(K_{\alpha_{(t^{OLD})}} || R)$ and $d_{\alpha_t} = H_2(MPK_t || K_{\alpha_t} || K_{\alpha_{(t^{OLD})}} || R)$,



computes $U_{\alpha_t} = K_{\alpha_t} + H_2(K_{\alpha_t(\text{OLD})} || R)$ and $V_{\alpha_t} = MPK_t + H_2(K_{\alpha_t(\text{OLD})} || N_1)$,

and returns $\langle ID_\alpha, t, U_{\alpha_t}, V_{\alpha_t}, d_{\alpha_t} \rangle$.

- **Send** ($\Pi_\beta^n, \langle ID_\alpha, N_\alpha, \text{"CAKE"} \rangle$): \mathcal{B} randomly generates $n_\beta \in_R Z_q^*$, computes N_β

$= n_\beta \cdot P$ and $y_1 = n_\beta \cdot N_\alpha$, and simulates **Hash_H** query to get $H_1(ID_\alpha || MPK_t)$ and

$H_1(ID_\beta || MPK_t)$. If $\alpha \in \{U, V\}$ and $\beta \in \{U, V\}$, \mathcal{B} randomly chooses $y_2 = g$. If

$\alpha \notin \{U, V\}$ and $\beta \in \{U, V\}$, \mathcal{B} finds $((ID_\alpha || MPK_t), r, H_1(ID_\alpha || MPK_t))$ in L_{H1} to

get r , and computes $K_{\alpha_t} = a_t \cdot r \cdot X$ and $y_2 = \hat{e}(K_{\alpha_t}, H_1(ID_\beta || MPK_t))$. If $\beta \notin \{U, V\}$,

\mathcal{B} finds $((ID_\beta || MPK_t), r, H_1(ID_\beta || MPK_t))$ in L_{H1} to get r , and computes $K_{\beta_t} =$

$a_t \cdot r \cdot X$ and $y_2 = \hat{e}(K_{\beta_t}, H_1(ID_\alpha || MPK_t))$. \mathcal{B} then simulates **Hash_{h2}** query to get v_β

$= h_2(ID_\alpha || ID_\beta || MPK_t || y_1 || y_2)$, and returns $\langle ID_\beta, N_\beta, v_\beta \rangle$.

- **Send** ($\Pi_\beta^n, \langle AID, N_\alpha, \text{"ACAKE"} \rangle$): \mathcal{B} randomly generates $n_\beta \in_R Z_q^*$, computes

$N_\beta = n_\beta \cdot P$ and $y_1 = n_\beta \cdot N_\alpha$, and simulates **Hash_{H1}** query to get $H_1(ID_\alpha || MPK_t)$

and $H_1(ID_\beta || MPK_t)$. If $\beta \in \{U, V\}$, \mathcal{B} randomly chooses $y_2 = g$. If $\beta \notin \{U, V\}$, \mathcal{B}

finds $((ID_\alpha || MPK_t), r, H_1(ID_\alpha || MPK_t))$ in L_{H1} to get r , and computes $K_{\beta_t} =$

$a_t \cdot r \cdot X$, $TK = \hat{e}(N_\alpha, K_{\beta_t})$, $ID_\alpha = AID \oplus TK$, $y_2 = \hat{e}(K_{\beta_t}, H_1(ID_\alpha || MPK_t))$. \mathcal{B} then

simulates **Hash_{h2}** query to get $v_\beta = h_2(ID_\alpha || ID_\beta || MPK_t || y_1 || y_2)$, and returns \langle

$N_\beta, v_\beta \rangle$.

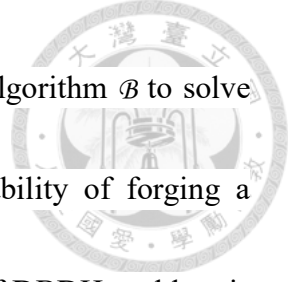


- **Execute** $(\Pi_\alpha^m, \Pi_\beta^n)$: \mathcal{B} simulates corresponding **Send** query to get the result and return it.

\mathcal{B} answers $g = \hat{e}(P, P)^{xyz}$ to DBDH problem if \mathcal{A} answers $b = 1$ to $Test_{SK}$ query, and answers $g \neq \hat{e}(P, P)^{xyz}$ if \mathcal{A} answers $b = 0$. Whether \mathcal{B} can successfully solve DBDH problem depends on whether \mathcal{A} asks **Send** and $Test_{SK}$ query for member U and V . The probability of \mathcal{A} asking **Send** and $Test_{SK}$ query for member U , whose partner is V , or for member V (whose partner is U) is $2 / (m+n+d)$. If \mathcal{A} successfully guesses b in $Test_{SK}$ query with advantage δ , which is non-negligible, \mathcal{B} can solve ECCDH problem with advantage $2\delta / (m+n+d)$, which is also non-negligible. It contradicts DBDH assumption, and thus Protocol 4 achieve existential unforgeability \square

Theorem 10 *The proposed Protocol 4 (ACAKE for a smart city) achieves existential anonymity on Decisional Bilinear Diffie-Hellman (DBDH) and hash function assumption.*

Proof: Suppose that there are one trusted third party Registration Center (RC), m Server, n Natural Persons (NP), and d IoT Devices in the smart city system, and each entity has a unique identity. Suppose that there is an adversary \mathcal{A} , who has ability to break the anonymity of a NP in the proposed ACAKE protocol with a non-negligible advantage in

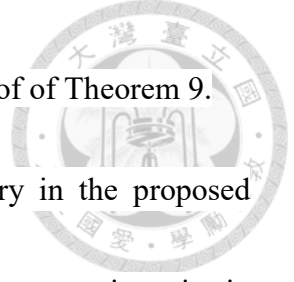


probabilistic polynomial time. Then we can construct a challenge algorithm \mathcal{B} to solve DBDH problem with a non-negligible advantage by using \mathcal{A} 's ability of forging a member. An instance $(G_1, G_2, P, \hat{e}, q, X = xP, Y = yP, Z = zP, g)$ of DBDH problem is given to \mathcal{B} , and \mathcal{B} 's goal is to determine if $g = \hat{e}(P, P)^{xyz}$.

\mathcal{B} guesses that \mathcal{A} tends to break the anonymity of a natural person, who makes an ACAKE with entity U . \mathcal{B} permeates the DBDH problem into the queries. \mathcal{B} manages one time seed list L_{seed} , and three hash lists L_{H1} , L_{H2} , L_{h1} , and L_h , which are initially empty. For each t -time-period, \mathcal{B} randomly generates $a_t, r_t \in_R Z_q^*$, computes $MPK_t = a_t \cdot X$, appends (t, a_t, MPK_t) to list L_{seed} , and appends $((ID_U || MPK_t), r_t, r_t \cdot Y)$ to list L_{H1} , i.e. let $H_1(ID_U || MPK_t) = r_t \cdot Y$. Note that $MRK_t = a_t \cdot x$ is unknown here. \mathcal{B} sets public parameters $Pub = \{q, G_1, G_2, P, \hat{e}\}$,

Without loss of generality, assume that \mathcal{A} does not ask queries for the same message more than once, \mathcal{A} may ask the following queries and obtain the corresponding results: *Hash*, *Extract*, *Send*, *Execute*, *Reveal*, *Rot*, *Corrupt*, and *Test_{SK}*, and no *Corrupt* (α) or *Reveal* (Π_α^S) query is asked before *Test_{ID}* (Π_α^S) query. \mathcal{B} simulates the oracle queries:

- The *Hash*, *Reveal*, *Rot*, *Corrupt*, and *Test* queries are identical to those in the proof

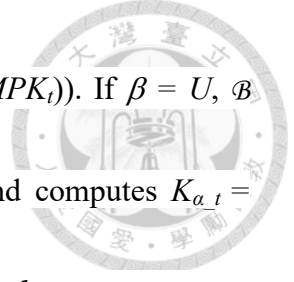


of Theorem 8, and **Extract** query is identical to those in the proof of Theorem 9.

- **Send** (Π_β^n, M): There are three different types of **Send** query in the proposed protocols: Update, CAKE, and ACAKE. Assume that current time is in t -time-period. \mathcal{B} checks list L_{seed} to get (t, a_t, MPK_t) , simulates **Hash_H** query to get $H_1(ID_\beta || MPK_t)$, finds $((ID_\beta || MPK_t), r_\beta, H_1(ID_\beta || MPK_t))$ in L_{H1} to get r_β , and responds according to the type of **Send** query.

- **Send** (RC, $\langle ID_\alpha, t^{OLD}, N_1, N_2, \text{"Update"} \rangle$): \mathcal{B} checks list L_{seed} to get $(t^{OLD}, a_{t^{OLD}}, MPK_t^{OLD})$, simulates **Hash_H** query for $(ID_\alpha || MPK_t)$, finds $((ID_\alpha || MPK_t), r_t, H_1(ID_\alpha || MPK_t))$ and $((ID_\alpha || MPK_t^{OLD}), r_{t^{OLD}}, H_1(ID_\alpha || MPK_t^{OLD}))$ in L_{H1} to get r_t and $r_{t^{OLD}}$. If $\alpha = U$, \mathcal{B} finds $((ID_A || t^{OLD} || N_1 || R || K_{\alpha_{(t^{OLD})}}), N_2)$ in list L_{h2} to get R and $K_{\alpha_{(t^{OLD})}}$, and computes $K_{\alpha_t} = a_t \cdot a_{t^{OLD}}^{-1} \cdot r_t \cdot r_{t^{OLD}}^{-1} \cdot K_{\alpha_{(t^{OLD})}}$. If $\alpha \neq U$, \mathcal{B} computes $K_{\alpha_{(t^{OLD})}} = a_t \cdot r_{t^{OLD}} \cdot X$ and $K_{\alpha_t} = a_t \cdot r_t \cdot X$. \mathcal{B} then simulates **Hash_{H2}** query to get $H_2(K_{\alpha_{(t^{OLD})}} || R)$, simulates **Hash_{h2}** query to get $d_{\alpha_t} = h_2(MPK_t || K_{\alpha_t} || K_{\alpha_{(t^{OLD})}} || R)$, computes $U_{\alpha_t} = K_{\alpha_t} + H_2(K_{\alpha_{(t^{OLD})}} || R)$ and $V_{\alpha_t} = MPK_t + H_2(K_{\alpha_{(t^{OLD})}} || N_1)$, and returns $\langle ID_\alpha, t, U_{\alpha_t}, V_{\alpha_t}, d_{\alpha_t} \rangle$.

- **Send** ($\Pi_\beta^n, \langle ID_\alpha, N_\alpha, \text{"CAKE"} \rangle$): \mathcal{B} randomly generates $n_\beta \in_R Z_q^*$, computes $N_\beta = n_\beta \cdot P$ and $y_1 = n_\beta \cdot N_\alpha$, and simulates **Hash_H** query to get $H(ID_\alpha || MPK_t)$. If $\beta \neq$

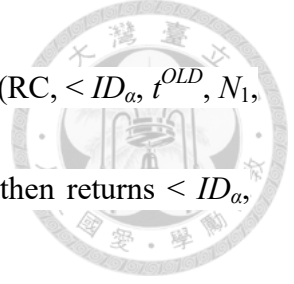


U , \mathcal{B} computes $K_{\beta_t} = a_t \cdot r_\beta \cdot X$ and $y_2 = \hat{e}(K_{\beta_t}, H_1(ID_\alpha || MPK_t))$. If $\beta = U$, \mathcal{B} finds $((ID_\alpha || MPK_t), r, H_1(ID_\alpha || MPK_t))$ in L_{H1} to get r , and computes $K_{\alpha_t} = a_t \cdot r \cdot X$ and $y_2 = \hat{e}(K_{\alpha_t}, H_1(ID_\beta || MPK_t))$. \mathcal{B} then simulates **Hash_{h2}** query to get $v_\beta = h_2(ID_\alpha || ID_\beta || MPK_t || y_1 || y_2)$, and returns $\langle ID_\beta, N_\beta, v_\beta \rangle$.

- **Send** ($\Pi_\beta^n, \langle AID, N_\alpha, \text{"ACAKE"} \rangle$): \mathcal{B} randomly generates $n_{\beta \in R} Z_q^*$, computes $N_\beta = n_\beta \cdot P$ and $y_1 = n_\beta \cdot N_\alpha$. If $\beta = U$, \mathcal{B} randomly chooses $v_\beta \in_R \{0, 1\}^l$. If $\beta \neq U$, \mathcal{B} computes $K_{\beta_t} = a_t \cdot r_\beta \cdot X$, $TK = \hat{e}(N_\alpha, K_{\beta_t})$, and $ID_\alpha = AID \oplus TK$, simulates **Hash_{H1}** query to get $H_1(ID_\alpha || MPK_t)$, computes $y_2 = \hat{e}(K_{\beta_t}, H_1(ID_\alpha || MPK_t))$, and simulates **Hash_{h2}** query to get $v_\beta = h_2(ID_\alpha || ID_\beta || MPK_t || y_1 || y_2)$. \mathcal{B} then returns $\langle N_\beta, v_\beta \rangle$.

- **Execute** ($\Pi_\alpha^m, \Pi_\beta^n$): Suppose that current time is in t -time-period, \mathcal{B} checks list L_{seed} to get (t, a_t, MPK_t) . \mathcal{B} simulates **Hash_{H1}** query to get $H_1(ID_\alpha || MPK_t)$ and $H_1(ID_\beta || MPK_t)$, and finds $((ID_\alpha || MPK_t), H_1(ID_\alpha || MPK_t), r_\alpha)$ and $((ID_\beta || MPK_t), H_1(ID_\beta || MPK_t), r_\beta)$ in L_{H1} to get r_α and r_β .

- **Execute_{UD}** (α, RC): \mathcal{B} chooses t^{OLD} , checks list L_{seed} to get $(t^{OLD}, a_{t^{OLD}}, MPK_t^{OLD})$, randomly generates $n \in_R Z_q^*$, and computes $N_1 = n \cdot P$ and $R = n \cdot MPK_t^{OLD}$. If $\beta = U$, \mathcal{B} randomly chooses $N_2 \in_R \{0, 1\}^l$. If $\beta \neq U$, \mathcal{B} simulates **Hash_{h2}** query to get



$N_2 = h_2(ID_A || t^{OLD} || N_1 || R || a_{tOLD} \cdot r_a \cdot X)$. \mathcal{B} simulates **Send** (RC, $\langle ID_a, t^{OLD}, N_1, N_2, \text{"Update"} \rangle$) query to get $\langle ID_a, t, U_{a,t}, V_{a,t}, d_{a,t} \rangle$. \mathcal{B} then returns $\langle ID_a, t^{OLD}, N_a, \text{"Update"} \rangle$ and $\langle ID_a, t, U_{a,t}, V_{a,t}, d_{a,t} \rangle$.

- **Execute**_{CAKE} ($\Pi_\alpha^m, \Pi_\beta^n$): There must be one of $\alpha \neq U$ and $\beta \neq U$ is true. If $\alpha \neq U$,

\mathcal{B} computes $K_{a,t} = a_t \cdot r_a \cdot X$, simulates **Hash**_{H1} query to get $H_1(ID_\beta || MPK_t)$, and

computes $y_2 = \hat{e}(K_{a,t}, H_1(ID_\beta || MPK_t))$. If $\beta \neq U$, \mathcal{B} computes $K_{\beta,t} = a_t \cdot r_\beta \cdot X$,

makes **Hash**_H query to get $H_1(ID_a || MPK_t)$, and computes $y_2 = \hat{e}(K_{\beta,t}, H_1(ID_a ||$

$MPK_t))$. \mathcal{B} randomly generates $n_\alpha, n_\beta \in_R Z_q^*$, computes $N_\alpha = n_\alpha \cdot P, N_\beta = n_\beta \cdot P$, and

$y_1 = n_\alpha \cdot n_\beta \cdot P$. \mathcal{B} simulates **Hash**_{h1} query to get $SK = h_1(ID_a || ID_\beta || N_\alpha || N_\beta || y_1$

$|| y_2)$, and simulates **Hash**_{h2} query to get $h_2(SK), v_\alpha = h_2(ID_a || ID_\beta || h_2(SK)), v_\beta =$

$h_2(ID_a || ID_\beta || MPK_t || y_1 || y_2)$.

- **Execute**_{ACAKE} ($\Pi_\alpha^m, \Pi_\beta^n$): If $\beta = U$, \mathcal{B} lets $N_\alpha = Z, TK = g$. If $\beta \neq U$, \mathcal{B} randomly

generates $n_\alpha \in_R Z_q^*$, computes $N_\alpha = n_\alpha \cdot P$ and $TK = \hat{e}(N_\alpha, a_t \cdot r_\beta \cdot X)$. \mathcal{B} computes


$AID = TK \oplus ID_U$, \mathcal{B} randomly generates $n_\beta \in_R Z_q^*$, computes $N_\beta = n_\beta \cdot P, y_1 = n_\beta \cdot$

N_U , and $y_2 = \hat{e}(a_t \cdot r_a \cdot X, H_1(ID_\beta || MPK_t))$. \mathcal{B} simulates **Hash**_{h2} query to get $v_\beta =$

$h_2(ID_a || ID_\beta || MPK_t || y_1 || y_2)$, and simulates **Hash**_{h1} query to get $SK = h_1(ID_a || ID_\beta$

$|| N_U || N_\beta || y_1 || y_2)$. \mathcal{B} then simulates **Hash**_{h2} query to get $v_\alpha = h_2(ID_a || ID_\beta ||$

$h_2(SK)$), and returns $\{\langle AID, N_U, \text{“ACAKE”} \rangle, \langle N_\beta, v_\beta \rangle, \langle v_a \rangle\}$.



\mathcal{B} answers $g = \hat{e}(P, P)^{xyz}$ to DBDH problem if \mathcal{A} answers $b = 1$ to $Test_{ID}$ query, and answers $g \neq \hat{e}(P, P)^{xyz}$ if \mathcal{A} answers $b=0$. Whether \mathcal{B} can successfully solve DBDH problem depends on whether \mathcal{A} asks $Send$ and $Test$ query for member U and V . The probability of \mathcal{A} asking $Send_{ACAKE}$ and $Test_{ID}$ query on any one of natural member, whose partner is U , is $n / (m+n+d)$. If \mathcal{A} successfully guesses b in the $Test$ query with advantage δ , which is non-negligible, \mathcal{B} can solve the ECCDH problem with advantage $\delta n / (m+n+d)$, which is also non-negligible. It contradicts DBDH assumption; hence, Protocol 4 achieves existential unforgeability. \square

Chapter 9

Performance Analysis and Comparisons



In this chapter, we analyze the performance of the proposed protocols and estimate the execution times by referring to some implementations to show that the execution times of each proposed protocol are rational and acceptable. We also compare the properties of the proposed protocols and relevant secure protocols to show the benefits of the proposed protocols, and point out the applications of each protocol.

9.1 Performance Analysis of the Proposed Protocols

We refer to some implementations [111][112] of elliptic curve cryptographic primitives and pairings [113][114][115] on microprocessors to estimate the execution times of performing the proposed protocols on low-power computing devices (i.e., smartcards) and low power devices.

Vliegen et al. [111] implemented elliptic curve cryptography over prime fields on the Xilinx VirtexII-Pro XC2VP30 FPGA device with maximal clock frequency 25.51 MHz, the execution times of $T_{G_{mul}}$, T_{inv} , $T_{G_{add}}$, and T_{mul} are 17.71 milliseconds (ms), 1.24 ms, 0.06276 ms, 0.00286 ms, respectively. Cavalieri and Cutuli [112] implemented hashing algorithm on MSP430 family, assuming a frequency of 8 MHz, and the



execution time is 0.065 milliseconds, which is negligible.

Scott et al. [113] implemented the standard Tate pairing ($k = 2$) over a large 512-bit field on a 32-bit Philips HiPerSmart™ smart card with a maximum clock speed of 36MHz, an instantiation of the MIPS-32 based SmartMIPS™ architecture, and the execution times of $TG_{\hat{e}}$ and TG_{mul} are 290 milliseconds (ms) and 270ms, respectively.

Cao et al. [114] implemented the standard Tate pairing over a large 512-bit field and the 160-bit hash in group G on a PIV 3.0 GHz processor with 512-MB memory and a Windows XP operation system, and on a Linux personal digital assistant equipped with a 206-MHz Strong ARM processor. The execution times of $TG_{\hat{e}}$, TG_{mul} , and T_H on the 3-GHz processor are 20.04 milliseconds (ms), 6.38 ms, and 3.04 ms, respectively. The execution times of TG_{mul} and T_H on a 206-MHz processor are 92.91 ms and 44.27 ms, respectively. The estimated execution time of $TG_{\hat{e}}$ on a 206-MHz processor, which is estimated from the result of 3-GHz processor, is $20.04 \times 3000 / 206 = 291.84$ milliseconds.

Xiong and Qin [115] implemented the same experiment on an Intel PXA270 processor at 624-MHz installed on the Linux personal digital assistant, and the execution times of $TG_{\hat{e}}$ and TG_{mul} are 96.2 milliseconds (ms) and 30.67, respectively.

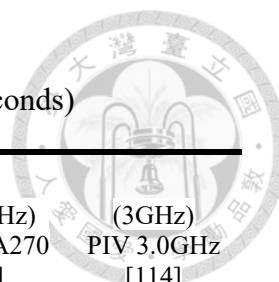


Table 5. Execution times of operations (in milliseconds)

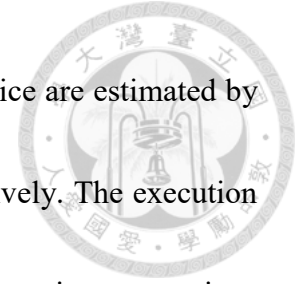
Platform	(25.51 MHz) Xilinx VirtexII-Pro XC2VP30 FPGA [111]	(36-MHz) MIPS-32 based SmartMIPS™ [113]	(206-MHz) StrongARM [114]	(624-MHz) Intel PXA270 [115]	(3GHz) PIV 3.0GHz [114]
TG_{mul}	17.71				
T_{inv}	1.24				
TG_{add}	0.06276	-	-	-	-
T_{mul}	0.00286				
$TG_{\hat{e}}$	-	290	291.84 (estimated)	96.2	20.04
TG_{mul}	-	270	92.91	30.67	6.38
T_H	-	253 (estimated)	44.27	14.62 (estimated)	3.04

Table 6. Estimated execution times on the user side in our Protocol 1

Platform	36-MHz device MIPS-32 based SmartMIPS™ [113]	206-MHz device StrongARM [114]	624-MHz device Intel PXA270 [115]
AAKA phase			
Computational cost		$3 TG_{\hat{e}} + 3 TG_{mul} + 9T_h$	
Execution time	1680 ms	1150 ms	380 ms

Table 7. Estimated execution times on the user side in our Protocol 2

Phase	Computational cost	Estimated execution time	Platform
User registration	$1T_h$	0.065 ms	(25.51 MHz) [111]
Online update	$2TG_{mul} + 4T_h$	35.68 ms	Xilinx VirtexII-Pro
Login and AAKA	$3TG_{mul} + T_{mul} + 6T_h$	53.523 ms	XC2VP30 FPGA




The time costs of TH on a 36-MHz device and on a 3GHz device are estimated by $3.04 \times 3000 / 624 = 14.62$ and $3.04 \times 3000 / 36 = 253$ milliseconds, respectively. The execution times of performing $TG_{\hat{e}}$, TG_{mul} , T_h , TG_{mul} , T_{inv} , TG_{add} , and T_{mul} operations on various mobile devices are summarized in Table 5. We ignore T_h , TG_{add} , T_{mul} , and the computational costs of generating a random number, XOR, and concatenation operations in this dissertation, since they are extremely low.

9.1.1 Our Protocol 1 (General AKA)

In our Protocol 1, which is proposed in Chapter 5, the computational cost of a user is $3 TG_{\hat{e}} + 3 TG_{mul} + 9T_h$ during the login and AKA phase, and the approximate execution times on a 36-MHz device, 206-MHz device, and 624-MHz device are about $3 \times 290 + 3 \times 270 = 1680$ milliseconds (ms), $3 \times 291.84 + 3 \times 92.91 = 1150$ ms, and $3 \times 96.2 + 3 \times 30.67 = 380$ ms, respectively. The execution times are summarized in Table 6. Thus, the proposed Protocol 1 is applicable to the low-power mobile devices.

9.1.2 Our Protocol 2 (AKA for TMIS)

Table 7 shows the estimated executing times on the user side in our Protocol 2, which is proposed in Chapter 6. The computational costs of a user in the registration phase, online update phase, and login and AKA phase are $1T_h$, $2TG_{mul} + 4T_h$, and



$3TG_{mul} + T_{mul} + 6T_h$, respectively. The estimated execution time of a user executing registration phase, online update phase, and login and AKA phase on a 25.52 MHz device are only 0.065 ms, $2 \times 17.71 + 4 \times 0.065 = 35.68$ ms, and $3 \times 17.71 + 0.00286 + 6 \times 0.065 = 53.523$ ms, respectively. Obviously, our Protocol 2 is well applicable to the low-power mobile devices.

9.1.3 Our Protocol 3 (AKE for IoT)

When an IoT node mutually authenticates with another IoT node and establishes a new session key in our Protocol 3, which is proposed in Chapter 7, the computational cost of the IoT node is $1TG_{\hat{e}} + 2TG_{mul} + 1TG_{add} + 6T_h$, and the estimated execution time is less than 0.6504 seconds.

Chatterjee et. al. [13] implemented their protocol on an Intel Edison board, with a Digilent Nexys-4 FPGA board consisting of an Artix-7 FPGA. The scenario is that a video camera transmits unencrypted captured video over a network. The camera, Edison board, and Artix-7 FPGA together form an IoT node, and the receiver PC acts as the verifier. The latency of running the end-to-end authentication protocol is 480.11 ms on average. The computational costs of each component in Chatterjee et. al.'s [13] protocol and our protocol are listed in Table 8.

Table 8. Computational costs of our Protocol 3 and relevant protocols

	Chatterjee et. al's protocol [13]	Our Protocol 3 (IoT)
IoT Node	$1TG_{\epsilon} + 2TG_{mul} + 2TG_{add} + 7T_H + T_{BCH}$	$1TG_{\epsilon} + 2TG_{mul} + 1TG_{add} + 6T_H$
Verifier	$1TG_{\epsilon} + 2TG_{mul} + 6TG_{add} + 9T_H$	None
Latency ^a	480.11 ms	< 480.11 ms

^a Latency of end-to-end authentication.

Table 9. Estimated execution times of our Protocol 4

	IoT Device (36-MHz) [113]	Natural Person (206-MHz) [114]	(624-MHz) [115]	Server (3GHz) [114]	RC
TG_{ϵ}	290	291.84 (estimated)	96.2	20.04	
TG_{mul}	270	92.91	30.67	6.38	
T_H	253 (estimated)	44.27	14.62 (estimated)	3.04	
Online Updating	1299	318.63	105.2	21.88	31.3
Executing CAKA	1336	566.2	186.78	38.88	-
Executing AKA	1626	950.95	292.93	58.92	-

Table 10. Computational costs of the proposed protocols

Entity	Protocol 1 (Chapter 5)	Protocol 2 (Chapter 6)	Protocol 3 (Chapter 7)	Protocol 4 (Chapter 8)
RC	0	0	0	0
User	$3TG_{\epsilon} + 3TG_{mul} + 9T_h$	$3TG_{mul} + T_{mul} + 6T_h$	$1TG_{\epsilon} + 2TG_{mul} + 1TG_{add} + 6T_H$	$2TG_{\epsilon} + 3TG_{mul} + 2TG_H + TG_{add} + 4T_h$
Server	$3TG_{\epsilon} + 2TG_{mul} + 6T_h$	$3TG_{mul} + 4T_h$	$1TG_{\epsilon} + 2TG_{mul} + 1TG_{add} + 6T_H$	$2TG_{\epsilon} + 2TG_{mul} + 2TG_H + TG_{add} + 4T_h$




As shown in Table 8, the computational cost of an IoT node in our protocol is less than Chatterjee et. al.'s [13] protocol, and there is no verifier in our protocol. According to Chatterjee et. al.'s implementation, the latency of running the end-to-end authentication protocol of our protocol in Chatterjee et. al.'s implementation would be less than 480.11 ms.

9.1.4 Our Protocol 4 (CAKE/ACAKE)

In the online update phase, the time cost of cryptographic operations of a member and RC is $2TG_{mul} + 3TG_H + 4TG_{add} + 2T_h$ and $3TG_{mul} + 4TG_H + 2TG_{add} + 2T_h$, respectively. The time cost of cryptographic operations of each member is $TG_{\hat{e}} + 2TG_{mul} + 2TG_H + TG_{add} + 4T_h$ in CAKE phase. In ACAKA phase, the time cost of cryptographic operations of NP and his/her partner member is $2TG_{\hat{e}} + 3TG_{mul} + 2TG_H + TG_{add} + 4T_h$ and $2TG_{\hat{e}} + 2TG_{mul} + 2TG_H + TG_{add} + 4T_h$, respectively. Note that TG_{add} and T_h are negligible and ignored here. Table 9 summarizes the estimated execution times (in milliseconds) of executing cryptographic operations and the estimated execution times of RC and each kind of members during online update phase, CAKE, and ACAKE phases of our protocol 4. As shown in Table 9, the executing times are appropriate.

Assume that RC has 100 computers, with frequency 3GHz, simultaneously




perform member's registration and online updating, then RC can perform registration for 10615 entities in one second, where $1000\text{ms}/((6.38\text{ms}+3.04\text{ms})/\text{entity})*100=10615$ entities, or perform online updating for 3194 members in one second, where $1000\text{ms}/(31.3\text{ms}/\text{member})*100=3194$ members. Thus, RC has ability performing online update for all members in each time period to deal with the member revocation issue. The proper length of a time period, which is decided by the supervisor, may be hours, days, or weeks.

9.2 Comparisons

In this section, we compare the four proposed protocols first, and then compare them with relevant secure protocols in different applications. We compare our Protocol 1 and Protocol 2 with relevant secure AAKA protocols for general multi-server environments, compare our Protocol 2 with relevant secure AAKA protocols for TMIS, compare our Protocol 3 with relevant secure IoT AAKE protocols, and compare our Protocol 4 with relevant secure AKE protocols for smart cities.

9.2.1 Four Proposed Protocols

We have elaborated on the ten properties (P1~P10) of the proposed Protocol 1, 2, 3, and 4 in Chapter 5, 6, 7, and 8, respectively, and we are going to make a description of



their difference. The computation costs and the properties of the four proposed protocols are summarized in Table 10 and Table 11, respectively. As shown in Table 10, the computation costs of Protocol 1, 2, and 4 are close. Protocol 2 has lowest computation cost; meanwhile, it lost some properties (P6, P7, P8).

Protocol 1, 2, and 3 do not deal with the member revocation problem efficiently, and Protocol 4 adopts online update in each time period to revoke the member. Protocol 4 achieves all the properties and keeps the efficiency because of the key update method and identity based authentication. Only valid members can get their new secret keys through the online update. As mentioned in the design guideline, the user encrypts his/her identity by using the server's public key in Protocol 4, and the public keys are identities of members.

No perfect protocol, only the most appropriate protocol. Efficiency and properties are trade-off. Choose the most appropriate protocol for different applications. Our Protocol 1 is applicable to common multi-server environment; Protocol 2 is applicable to the TMIS, autonomous vehicle, etc.; Protocol 3 is applicable to the Wireless Body Area Network (WBAN), factories, etc.; Protocol 4 is applicable to the smart city, the Vehicular Ad hoc Network (VANET), etc.



9.2.2 Our Protocol 1 and Relevant Secure AKA Protocols

The properties of relevant AKA protocols [45][51][53][54][58][60][61], which are designed for general multi-server environments, are compared in Table 12. It shows that no present AKA protocol achieves the entire properties (P1~P10). Therefore, we propose Protocol 1, which achieves more properties.

We compare the computation costs of members during the login and AKA phase in our Protocol 1 and relevant secure protocols [45][51][53][54][58][60][61] in Table 13.

All the protocols, the computation costs of the user in the login and AKA phase are low and close, all of them are applicable to low power devices. Only our Protocol 1 simultaneously achieves all the properties except member revocation (P6). The registration center need not be involved in the user login and AKA phases and need not maintain any table; meanwhile, no public key needs to be announced. Moreover, we provide formal security proofs of Protocol 1, and the computation costs in Protocol 1 are acceptable and applicable for low power devices, which have been discussed in Section 9.1.1.

Table 11. Properties of the proposed protocols

Our protocols	Protocol 1 (Chapter 5)	Protocol 2 (Chapter 6)	Protocol 3 (Chapter 7)	Protocol 4 (Chapter 8)
Application	General Case	TMIS	IoT	ALL ^a
Property				
Compatible authentication	No	No	No	Yes
(P1) Three-factor authentication	Yes ^b	Yes ^b	Yes ^c	Yes ^{bc}
(P2) Multi-server environment	Yes	Yes	N/A	Yes
(P3) User anonymity	Yes	Yes	Yes	Yes
(P4) User untraceability	Yes	Yes	Yes	Yes
(P5) Perfect forward secrecy	Yes	Yes	Yes	Yes
(P6) Member revocation	No ^d	No ^d	No ^e	Yes
(P7) Independent authentication	Yes	Optional ^f	No	Yes
(P8) Table free	Yes		No	Yes
(P9) Public key announcement free	Yes	No ^g	Yes	Yes
(P10) Formal security proof	Yes	Yes	Yes	Yes

^a:it is not only applicable to smart cities but is also applicable to other applications, ^b: biometric, ^c:PUF, ^d: CRL, ^e:delete the helper data, ^f: achieves either P7 or P8, ^g: the server's public keys are issued to ther users in the online update phase.

Table 12. Properties of our Protocol 1 and relevant secure protocols

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Lin et al. [45]	Y ^a	Y	Y	Y	Y	N	Y	N ^{de}	Y	N
Jiang et al. [51]	Y ^a	Y	Y	N	Y	N	N	Y	Y	Y
Odelu et al. [53]	Y ^a	Y	Y	Y	Y	Y ^b	N	N ^d	Y	N ⁱ
Park-Park [54]	Y ^a	Y	Y	Y	Y	N	N	N ^d	Y	N ⁱ
Irshad et al. [58]	Y ^a	Y	Y	Y	Y	N ^c	Y	Y	N ^{de}	Y
Xu et al. [60]	Y ^a	Y	Y	Y	Y	Y ^b	Y	N ^{de}	N ^f	Y
Qi et al. [61]	Y ^a	Y	Y	Y	Y	N	N	Y	N ^f	N ⁱ
Our Protocol 1 (General)	Y ^a	Y	Y	Y	Y	N ^c	Y	Y	Y	Y

Y: Yes, N: No, ^a: biometric based, ^b: data deletion, ^c: CRL, ^d: RC, ^e: users, ^f: servers, ^g: achieves either P7 or P8, ^h: the server's public keys are issued to ther users in the online update phase, ⁱ: BAN logic.

Table 13. Computational costs of members executing the login and AKA phase

	RC	User	Server
Lin et al. [45]	0	$3T_{sym}+2TG_{mul}+8T_h$	$2T_{sym}+2TG_{mul}+3T_h$
Jiang et al. [51]	$10T_h$	$2TG_{mul}+7T_h$	$2TG_{mul}+5T_h$
Odelu et al. [53]	$3T_{sym}+1TG_{mul}+10T_h$	$1T_{sym}+3TG_{mul}+7T_h$	$2T_{sym}+2TG_{mul}+6T_h$
Park-Park [54]	$11T_h$	$2TG_{mul}+10T_h$	$3TG_{mul}+4T_h$
Irshad et al. [58]	0	$4T_C+7T$	$4T_C+4T_h$
Xu et al. [60]	0	$3TG_{mul}+10T_h$	$3TG_{mul}+6T_h$
Qi et al. [61]	$1T_{sym}+1TG_{mul}+T_{kdf}+5T_h$	$3TG_{mul}+6T_h$	$1T_{sym}+4TG_{mul}+T_{kdf}+4T_h$
Our Protocol 1 (General)	0	$3TG_{\varepsilon}+3TG_{mul}+9T_h$	$3TG_{\varepsilon}+2TG_{mul}+6T_h$
Our Protocol 2 (TMIS)	0	$3TG_{mul}+T_{mul}+6T_h$	$3TG_{mul}+4T_h$

9.2.3 Our Protocol 2 and Relevant Secure AKA Protocols for TMIS

We compare our Protocol 2 with relevant AKA protocols for TMIS [66][69][72][76][77][80] in Table 14. It shows that only our Protocol 2 provides independent authentication (P7) and public key announcement free (P9); only our Protocol 2 and Wei et al. [80] are applicable to multi-server environments (P2); only our Protocol 2 and Jiang et al. [77] deal with member revocation issue (P6). Obviously, our Protocol 2 is preferable to other relevant protocols for TMIS.



9.2.4 Our Protocol 3 and Relevant PUF Based AKE Protocols for IoT

The comparison of Braeken's [89], Chatterjee et. al.'s [13], and our Protocol 3 are shown in Table 15. Only in our protocol, IoT nodes can independently authenticate each other without the help of trusted third party. No explicit CRP is in Chatterjee et. al.'s [13] or our protocol. It shows that our Protocol 3 is preferable to other relevant protocols for IoT.

9.2.5 Our Protocol 4 and Relevant AKE Protocols for a Smart City

The comparison of our Protocol 4 and the relevant AKE protocol for a smart city are shown in Table 16. Table 16 shows that only our ACAKE protocol achieves all the properties (P1~P10); meanwhile, it keeps the efficiency that the computation costs in Protocol 1, which have been discussed in Section 9.1.4, are acceptable and applicable to smart cities.

Table 14. Comparisons of our Protocol 2 and relevant AKA protocols for TMIS

	Xu-Wu [66]	Das [69]	Jiang et al. [72]	Wazid et al. [76]	Jiang et al. [77]	Wei et al. [80]	Our Protocol 2 (TMIS)
(P1) Three-factor authentication	Y ^a	Y ^a	Y ^a	Y ^a	Y ^a	Y ^a	Y ^a
(P2) Multi-server environment	N	N	N	N	N	Y	Y
(P3) User anonymity	Y	Y	Y	Y	Y	Y	Y
(P4) User untraceability	N	Y	Y	N	Y	N	Y
(P5) Perfect forward secrecy	Y	Y	Y	Y	Y	Y	Y
(P6) Member revocation	N	N	N	N	Y	N	Y
(P7) Independent authentication	-	-	-	-	-	-	Y/N ^b
(P8) Table free	Y	Y	Y	Y	N ^c	Y	Y/N ^b
(P9) Public key announcement free	-	-	-	-	-	-	Y ^d
(P10) Formal security proof	N	N ^e	N	Y	N ^f	N	Yes

Y: Yes, N:No, ^abiometric based, ^beither independent authentication or table free, ^cServer, ^duse identity as public key, ^eAVISPA, ^fBAN logic.

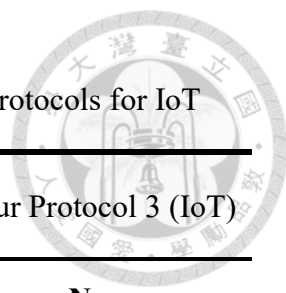


Table 15. Comparisons of our Protocol 3 and the relevant AKE protocols for IoT

	Braeken's protocol [89]	Chatterjee et. al's protocol [13]	Our Protocol 3 (IoT)
Explicit CRPs	Stored in Server	None	None
Implicit CRPs	None	Stored in Security Association Provider	Stored in Data Provider
Helper ^a	Server	Verifier	None
Outer database	None	Security Association Provider	Data Provider
(P1) Multi-factor authentication	N ^b	N ^b	N ^b
(P2) Multi-server environment	-	-	-
(P3) User anonymity	N	N	N
(P4) User untraceability	N	N	N
(P5) Perfect forward secrecy	Y	Y	Y
(P6) Member revocation	N	N	N
(P7) Independent authentication	N	N	Y
(P8) Table free	N	N	N
(P9) Public key announcement free	Y	Y	Y
(P10) Formal security proof	N	Y	Y

Y: Yes, N:No, ^a trusted third party involved in authentication phase, ^b only using PUF as the fingerprint of the IoT device.

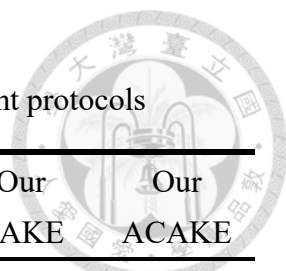


Table 16. Comparisons of our Protocol 4 and the relevant protocols

	[95]	[96]	[97]	[98]	[99]	Our CAKE	Our ACAKE
Compatible authentication	N	N	N	N	N	Y^{ab}	Y^{ab}
(P1) Multi-factor authentication	N	N	N	N	N	Y	Y
(P2) Multi-server environment	N	N	N	Y	Y	Y	Y
(P3) User anonymity	Y	Y	Y	Y	Y	N	Y
(P4) User untraceability	Y	Y	Y	Y	Y	N	Y
(P5) Perfect forward secrecy	Y	Y	Y	Y	Y	Y	Y
(P6) Member revocation	N	N	N	N	N	Y	Y
(P7) Independent authentication	N	Y	Y	N	Y	Y	Y
(P8) Table free	Y	N ^c	Y	Y	Y	Y	Y
(P9) Public key announcement free	Y	Y	Y	Y	N	Y	Y
(P10) Formal security proof	N	Y	Y	N	N	Y	Y

^abiometric based, ^bPUF based, ^cServer, ^duse identity as public key, ^eAVISPA, ^fBAN logic.



Chapter 10

Conclusions and Future Work

10.1 Conclusions

In this dissertation, we surveyed recently proposed three-factor AKE protocols with user privacy preserving, and then discuss and analyze these protocols to propose the guideline for designing a secure AKE protocol with user privacy protection for multi-server environments. We then obeyed the guideline to design three AKE protocols that are applicable to general situations, the TMIS, and the IoT, respectively.

We brought up the concept of a compatible authentication and key exchange (CAKE) protocol, which provides cross-species authentication that any two valid entities can authenticate with each other and create a secure session key. We proposed a CAKE protocol for a smart city, and extended the CAKE protocol to an anonymous CAKE (ACAKE) protocol, which additionally provides an anonymity option for natural persons to protect their privacy.

We constructed a formal security model for an AKE protocol in multi-server environments, and gave the formal proofs of each proposed protocol. We demonstrated that our protocols are efficient enough for portable mobile devices by estimating the

executing times of performing our protocols on low-power devices. We also compared our protocol to relevant AKE protocols to show that our protocols are better than existing protocols.



There is no perfect protocol, only the most appropriate protocol. Efficiency and properties are trade-off. We proposed four AKE protocols to provide the chance to choose the most appropriate protocol for each common application.

10.2 Future Work

There has been steady progress towards building quantum computers in recent years. The large-scale quantum computers would threaten the security of many present public-key cryptosystems if they are realized. Public-key encryption (PKE), key-establishment mechanisms (KEM), and digital signatures based on factoring, discrete logarithms, and elliptic curve cryptography will be the most severely affected; meanwhile, symmetric cryptographic primitives, such as block ciphers and hash functions, will only be mildly affected. [116] Post-quantum cryptography (PQC) is a study of cryptosystems that would be secure against adversaries who have both quantum and classical computers and that can be deployed without drastic changes to existing communication networks and protocols.



Table 17. PQC algorithms selected by the NIST

NIST selection	Algorithm	Hardness	
PKE & KEM	Classic McEliece	Code-based	
	Third-Round finalists	CRYSTALS-KYBER	Module Learning With Errors (MLWE)
		NTRU	Lattice-based KEM
		SABER	Module Learning With Rounding (MLWR)
	Alternate candidate	BIKE	Code-based
		FrodoKEM	Lattice-based LWE
		HQC	Code-based
		NTRU Prime	Lattice-based
		SIKE	Isogenies of elliptic curves
	Digital signatures	Third-Round finalists	CRYSTALS-DILITHIUM
		FALCON	Lattice-based
		Rainbow	Multivariate
	Alternate candidate	GeMSS	Multivariate
		Picnic	Hash-based
		SPHINCS+	Hash-based

Generally speaking, there are five categories of post-quantum cryptography algorithms: lattice-based, code-based, isogenies of elliptic curves, multivariate, and hash-based. The National Institute of Standards and Technology (NIST) is in the process of selecting public-key cryptographic algorithms through a public, competition like process. The new public-key cryptography standards will specify one or more

additional algorithms for digital signatures, public-key encryption, and key-establishment. The NIST Post-Quantum Cryptography Standardization Process began in 2017 with 69 candidate algorithms that met both the minimum acceptance criteria and submission requirements. NIST selected 26 algorithms to advance to the second round for more analysis in 2019, and selected 7 third-round finalists and 8 alternate candidates to advance to the third round in 2020 [116]. We summarize these 15 algorithms, which are selected by the NIST, in Table 17.

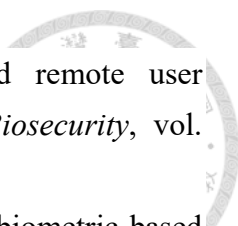
Our future work is to design quantum resistant AAKA/ACAKE protocols and encryption/signature/signcryption schemes for various circumstances, such as the Internet of things (IoT), the cloud service, the Telecare Medicine Information System (TMIS), the smart city, the Vehicular Ad hoc Network (VANET), the epidemic control system, the financial system, the electronic voting, etc.

Bibliography



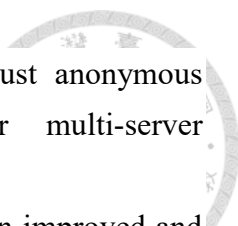
- [1] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770-772, 1981.
- [2] T. Hwang, Y. Chen, and C. J. Lai, "Non-interactive password authentications without password tables," in: *Proceedings of IEEE Region 10 Conference on Computer and Communication Systems (TENCON'90)*, IEEE, Hong Kong, 1990, vol. 1, pp. 429-431.
- [3] Y.H. Chuang and Y.M. Tseng, "Towards generalized ID-based user authentication for mobile multi-server environment," *International Journal of Communication Systems*, vol. 24, no. 4, pp. 447-460, 2012.
- [4] Y.H. Chuang, Y.M. Tseng, and C.L. Lei, "Efficient mutual authentication and key agreement with user anonymity for roaming services in global mobility networks," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 9, pp. 6415-6428, 2012.
- [5] Y.M. Tseng, S.S. Huang, T.T. Tsai, and J.H. Ke, "List-free ID-based mutual authentication and key agreement protocol for multi-server architectures," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 1, pp. 102-122, 2016.
- [6] W. Wang, Y. Chen, and Q. Zhang, "Privacy-preserving location authentication in Wi-Fi networks using fine-grained physical layer signature," *IEEE Transactions on Wireless Communications*, vol. 15, no. 2, pp. 1218-1225, 2016.
- [7] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in: *Proceedings of 1st ACM conference on Computer and Communications Security (CCS1993)*, Fairfax, Virginia, USA, 1993, pp. 62-73.
- [8] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: how to generate strong keys from biometrics and other noisy data," in: *Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004)*, Interlaken, Switzerland, May 2–6 2004, vol. 3027, pp. 523-540.
- [9] D. Mukhopadhyay, "PUFs as promising tools for security in internet of things," *IEEE Design and Test*, vol. 33, no. 3, pp. 103-115, June 2016.
- [10] P. Tuyls and B. Skoric, "Secret key generation from classical physics: physical uncloneable functions," In: S. Mukherjee, R.M. Aarts, R. Roovers, F.

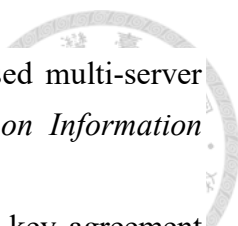
- Widdershoven, and M. Ouwerkerk (eds), *AmIware Hardware Technology Drivers of Ambient Intelligence*, Philips Research, vol. 5, pp. 421-447, 2006.
- [11] D. Lim, J. W. Lee, B. Gassend, G. Edward Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on VLSI Systems*, vol. 13, no. 10, pp. 1200-1205, 2005.
- [12] C. Herder, M.D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: a tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp.1126-1141, 2014.
- [13] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R.S. Chakraborty, D. Mahata, and M.M. Prabhu, "Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp.424-437, MAY/JUNE 2019.
- [14] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura, "Cryptographic key generation from PUF data using efficient fuzzy extractors," in: *Proceedings of 16th International Conference on Advanced Communication Technology (ICACT)*, IEEE, Feb. 2014, pp. 23-26.
- [15] J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M.M. Yu, "Efficient fuzzy extraction of PUF-induced secrets: theory and applications," in: *Proceedings of 18th International Conference on Cryptographic Hardware and Embedded Systems, (CHES 2016)*, Santa Barbara, CA, USA, 2016, pp. 412-431.
- [16] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, New York, USA, 2004.
- [17] K.Y. Choi, J.Y. Hwang, and D.H. Lee, "Efficient ID-based group key agreement with bilinear maps," in: *Proceedings of 7th International Workshop on Theory and Practice in Public Key Cryptography(PKC 2004)*, Springer, Singapore, March 1-4, 2004, vol. 2947, pp: 130-144.
- [18] K.Y. Choi, J.Y. Hwang, D.H. Lee, and I.S. Seo, "ID-based authenticated key agreement for low-power mobile devices" in: *Proceedings of 10th Australasian Conference on Information Security and Privacy (ACISP2005)*, Springer, Brisbane, Australia, 2005, vol. 3574, pp. 494-505.
- [19] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in: *Proceedings of 21st Annual International Cryptology Conference (Crypto 2001)*, Springer, Santa Barbara, California, USA, August 19-23, 2001, vol. 2139, pp. 213-229.

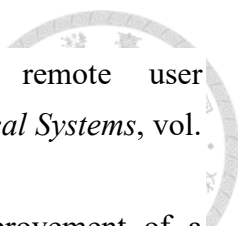
- 
- [20] M.K. Khan and S. Kumari, "An improved biometrics-based remote user authentication scheme with user anonymity," *Biometrics and Biosecurity*, vol. 2013, no. 491289, pp. 1-9, 2013.
- [21] A. Chaturvedi, D. Mishra, and S. Mukhopadhyay, "Improved biometric-based three-factor remote user authentication scheme with key agreement using smart card," in: *Proceedings of International Conference on Information Systems Security (ICISS)*, ACM, Kolkata, India, 2013, pp. 63-77.
- [22] Y. An, "Improved biometrics-based remote user authentication scheme with session key agreement," in: *Proceedings of Computer Applications for Graphics, Grid Computing, and Industrial Environment*, Springer, Gangneug, Korea, 2012, vol. CCIS, pp: 307-315.
- [23] X. Li, J. Niu, Z. Wang, and C. Chen, "Applying biometrics to design three-factor remote user authentication scheme with key agreement," *Security and Communication Networks*, vol. 7, pp. 1488-1497, 2014.
- [24] S.H. Islam, "Provably secure dynamic identity-based three-factor password authentication scheme using extended chaotic maps," *Nonlinear Dynamics*, vol. 78, no. 3, pp. 2261-2276, 2014.
- [25] L. Cao and W. Ge, "Analysis and improvement of a multi-factor biometric authentication scheme," *Security and Communication Networks*, vol. 8, no. 4, pp. 617-625, 2015.
- [26] Y. An, "Security analysis and enhancements of an effective biometric-based remote user authentication scheme using smart cards," *Journal of Biomedicine and Biotechnology*, no. 519723, 2012.
- [27] Y. Choi, Y. Lee, J. Moon, and D. Won, "Security enhanced multi-factor biometric authentication scheme using bio-hash function," *PLoS ONE*, vol. 12, no. 5, e0176250, 2017.
- [28] Y. Park, K. Park, and K. Lee, "Security analysis and enhancements of an improved multi-factor biometric authentication scheme," *International Journal of Distributed Sensor Networks*, vol. 13, no. 8, pp. 1-12, 2017.
- [29] Y. Zhao, S. Li, L. Jiang, and T. Liu, "Security-enhanced three-factor remote user authentication scheme based on Chebyshev chaotic maps," *International Journal of Distributed Sensor Networks*, vol. 15, no. 4, pp. 1-12, 2019.
- [30] A.G. Reddy, A.K. Das, V. Odelu, A. Ahmad, and J.S. Shin, "A privacy preserving three-factor authenticated key agreement protocol for client-server environment,"

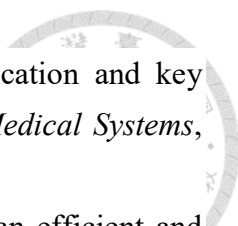
Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 2, pp. 661-680, 2019.

- [31] M. Qi and J. Chen, "An efficient two-party authentication key exchange protocol for mobile environment," *International Journal of Communication Systems*, vol. 30, no. 16, e3341, 2017.
- [32] Y. Lu, L. Li, H. Peng, and Y. Yang, "Robust anonymous two-factor authenticated key exchange scheme for mobile client-server environment," *Security and Communication Networks*, vol. 9, no. 11, pp. 1331-1339, 2016.
- [33] M.C. Chuang and M.C. Chen, "An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1411-1418, 2014.
- [34] T. Maitra and D. Giri, "An efficient biometric and password-based remote user authentication using smart card for telecare medical information systems in multi-server environment," *Journal of Medical Systems*, vol. 38, no. 142, 2014.
- [35] Y. Choi, J. Nam, D. Lee, J. Kim, J. Jung, and D. Won, "Security enhanced anonymous multiserver authenticated key agreement scheme using smart cards and biometrics," *The Scientific World Journal*, no. 281305, 2014.
- [36] D. Mishra, A.K. Das, and S. Mukhopadhyay, "A secure user anonymity-preserving biometric-based multi-server authenticated key agreement scheme using smart cards," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8129-8143, 2014.
- [37] C. Wang, X. Zhang, and Z. Zheng, "Cryptanalysis and improvement of a biometric-based multi-server authentication and key agreement scheme," *PLoS ONE*, vol. 11, no. 2, e0149173, 2016.
- [38] L. Yang and Z. Zheng, "Cryptanalysis and improvement of a biometrics-based authentication and key agreement scheme for multi-server environments," *PLoS ONE*, vol. 13, no. 3, e019409, 2018.
- [39] Y. Lu, L. Li, X. Yang, and Y. Yang, "Robust biometrics based authentication and key agreement scheme for multi-server environments using smart cards," *PLoS ONE*, vol. 10, no. 5, e0126323, May 2015.
- [40] Y. Lu, L. Li, H. Peng, and Y. Yang, "A biometrics and smart cards-based authentication scheme for multi-server environments," *Security and Communication Networks*, vol. 8, pp. 3219-3228, March 2015.
- [41] J. Moon, Y. Choi, J. Jung, and D. Won, "An improvement of robust biometrics-based authentication and key agreement scheme for multi-server environments using smart cards," *PLoS ONE*, vol. 10, no. 12, e0145263, 2015.

- 
- [42] H. Guo, P. Wang, X. Zhang, Y. Huang, and F. Ma, “A robust anonymous biometric-based authenticated key agreement scheme for multi-server environments,” *PLoS ONE*, vol. 12, no. 11, e0187403, 2017.
- [43] S.A. Chaudhry, H. Naqvi, M.S. Farash, T. Shon, and M. Sher, “An improved and robust biometrics-based three factor authentication scheme for multiserver environments,” *Journal of Supercomput*, vol. 74, pp. 3504-3520, 2018.
- [44] M. Wazid, A.K. Das, Saru Kumari, X. Li, and F. Wu, “Provably secure biometric-based user authentication and key agreement scheme in cloud computing,” *Security and Communication Networks*, vol. 9, pp. 4103-4119, 2016.
- [45] H. Lin, F. Wen, and C. Du, “An improved anonymous multi-server authenticated key agreement scheme using smart cards and biometrics,” *Wireless Personal Communications* , vol. 84, pp. 2351-2362, 2015.
- [46] R. Amin and G. Biswas, “Design and analysis of bilinear pairing based mutual authentication and key agreement protocol usable in multi-server environment,” *Wireless Personal Communications*, vol. 84, no. 1, pp. 439-462, 2015.
- [47] W.B. Hsieh and J.S. Leu, “An anonymous mobile user authentication protocol using self-certified public keys based on multi-server architectures,” *Journal of Supercomput*, vol. 70, no. 1, pp. 133-148, 2014.
- [48] P. Chandraka and H. Om, “Cryptanalysis and improvement of a biometric-based remote user authentication protocol usable in a multiserver environment,” *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 12, e3200, 2017.
- [49] Y.H. Chuang, C.L. Lei, and Hung-Jr Shiu, “Cryptanalysis of four biometric based authentication schemes with privacy-preserving for multi-server environment,” in: *Proceedings of the 15th Asia Joint Conference on Information Security (ASIAJCIS2020)*, IEEE, Taipei, Taiwan, 2020, pp. 66-73.
- [50] P. Chandrakar and H. Om, “A secure and robust anonymous three-factor remote user authentication scheme for multi-server environment using ECC,” *Computer Communications*, vol. 110, pp. 26-34, 2017.
- [51] P. Jiang, Q. Wen, W. Li, Z. Jin, and H. Zhang, “An anonymous and efficient remote biometrics user authentication scheme in a multi server environment,” *Frontiers of Computer Science*, vol. 9, no. 1, pp. 142-156, 2015.
- [52] D. He and D. Wang, “Robust biometrics-based authentication scheme for multiserver environment,” *IEEE Systems Journal*, vol. 9, no. 3, pp. 816-823, 2015.

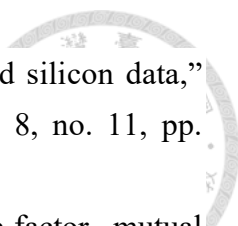
- 
- [53] V. Odelu, A.K. Das, and A. Goswami, "A secure biometrics-based multi-server authentication protocol using smart cards," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1953-1966, 2015.
- [54] Y.H. Park and Y.H. Park, "Three-Factor user authentication and key agreement using elliptic curve cryptosystem in wireless sensor networks," *Sensors*, vol. 16, no. 12, article 2123, pp. 1-17, 2016.
- [55] I.P. Chang, T.F. Lee, T.H. Lin, and C.M. Liu, "Enhanced two-factor authentication and key agreement using dynamic identities in wireless sensor networks," *Sensors*, vol. 15, no. 12, pp. 29841-29854, 2015.
- [56] Y. Choi, J. Nam, D. Lee, J. Kim, J. Jung, and D. Won, "Security improvement on biometric based authentication scheme for wireless sensor networks using fuzzy extraction," *International Journal of Distributed Sensor Networks*, no. 8572410, 2016.
- [57] E.J. Yoon and C. Kim, "Advanced biometric-based user authentication scheme for wireless sensor networks," *Sensor Letters*, vol. 11, no. 9, pp. 1836-1843, 2013.
- [58] A. Irshad, M. Sher, S.A. Chaudhary, H. Naqvi, and M.S. Farash, "An efficient and anonymous multi-server authenticated key agreement based on chaotic map without engaging registration Centre," *Journal of Supercomput*, vol. 72, pp. 1623-1644, 2016.
- [59] A.G. Reddy, E.J. Yoon, A.K. Das, V. Odelu, and K.Y. Yoo, "Design of mutually authenticated key agreement protocol resistant to impersonation attacks for multi-server environment," *IEEE Access*, vol. 5, no. 99, pp. 3622-3639, 2017.
- [60] D. Xu, J. Chen, and Q. Liu, "Provably secure anonymous three-factor authentication scheme for multi-server environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 611-627, 2019.
- [61] M. Qi, J. Chen, and Y. Chen, "A secure biometrics-based authentication key exchange protocol for multi-server TMIS using ECC," *Computer Methods and Programs in Biomedicine*, vol. 164, pp. 101-109, 2018.
- [62] R. Ali, and A. K. Pal, "An efficient three factor-based authentication scheme in multiserver environment using ECC," *Communication Systems*, vol. 31, no. 4, pp. e3484, 2018.
- [63] A.K. Das and A.Goswami, "A secure and efficient uniqueness and anonymity preserving remote user authentication scheme for connected health care," *Journal of Medical System.*, vol. 37, no. 9948, 2013.

- 
- [64] F. Wen, "A robust uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care," *Journal of Medical Systems*, vol. 37, no. 9980, 2013.
- [65] Q. Xie, W. Liu, S. Wang, L. Han, B. Hu, and T. Wu, "Improvement of a uniqueness-and-anonymity-preserving user authentication scheme for connected health care," *Journal of Medical Systems*, vol. 38, no. 91, 2014.
- [66] L. Xu and F. Wu, "Cryptanalysis and improvement of a user authentication scheme preserving uniqueness and anonymity for connected health care," *Journal of Medical Systems*, vol. 39, no. 10, 2015.
- [67] Z. Tan, "A user anonymity preserving three-factor authentication scheme for telecare medicine information systems," *Journal of Medical Systems*, vol. 38, no. 16, 2014.
- [68] H. Arshad and M. Nikooghadam, "Three-factor anonymous authentication and key agreement scheme for telecare medicine systems information," *Journal of Medical Systems*, vol. 38, no. 136, 2014.
- [69] A.K. Das, "A secure user anonymity-preserving three-factor remote user authentication scheme for the telecare medicine information systems," *Journal of Medical Systems*, vol. 39, no. 30, 2015.
- [70] Y. Lu, L. Li, H. Peng, and Y. Yang, "An enhanced biometric-based authentication scheme for telecare medicine information systems using elliptic curve cryptosystem," *Journal of Medical Systems*, vol. 39, no. 32, 2015.
- [71] R. Amin, S.H. Islam, G.P. Biswas, M.K. Khan, and M.S. Obaidat, "Design and analysis of an enhanced patient-server mutual authentication protocol for telecare medical information system," *Journal of Medical Systems*, vol. 39, no.11, 2015.
- [72] Q. Jiang, Z. Chen, B. Li, J. Shen, L. Yang, and J. Ma, "Security analysis and improvement of bio-hashing based three factor authentication scheme for telecare medical information systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, pp. 1061-1073, 2018.
- [73] D. Mishra, S. Mukhopadhyay, A. Chaturvedi, S. Kumari, and M. Khan, "Cryptanalysis and improvement of Yan et al.'s biometric-based authentication scheme for telecare medicine information systems," *Journal of Medical Systems*, vol. 38, no. 24, pp. 1-12, Jun. 2014.
- [74] X. Yan, W. Li, P. Li, J. Wang, X. Hao, and P. Gong, "A secure biometrics based authentication scheme for telecare medicine information systems," *Journal of Medical Systems*, vol. 37, no. 9972, pp. 1-6, Oct. 2013.

- 
- [75] R. Amin and G.P. Biswas, "A secure three-factor user authentication and key agreement protocol for TMIS with user anonymity," *Journal of Medical Systems*, vol. 39, no. 78, 2015.
- [76] M. Wazid, A.K. Das, S. Kumari, X. Li, and F. Wu, "Design of an efficient and provably secure anonymity preserving three-factor user authentication and key agreement scheme for TMIS," *Security and Communication Networks*, vol. 9, no. 13, pp. 1983-2001, 2016.
- [77] Q. Jiang, M.K. Khan, X. Lu, J. Ma, and D. He, "A privacy preserving three-factor authentication protocol for e-Health clouds," *Journal of Supercomput*, vol. 72, pp. 3826-3849, 2016.
- [78] A. Irshad and S.A. Chaudhry, "Comments on "A privacy preserving three-factor authentication protocol for e-health clouds","" *Journal of Supercomput*, vol. 73, pp. 1504-1508, 2017.
- [79] L. Zhang, S. Zhu, and S. Tang, "Privacy protection for telecare medicine information systems using a chaotic map-based three-factor authenticated key agreement scheme," *Journal of Biomedical and Health Informatics*, vol. 21, no. 2, pp. 465-475, 2017.
- [80] J. Wei, W. Liu, X. Hu, "On the security and improvement of privacy preserving 3-factor authentication scheme for TMIS," *International Journal of Communication Systems*, vol. 31, no. e3767, 2018.
- [81] T. Alam, "A reliable communication framework and its use in Internet of Things (IoT)," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 2, no. 5, pp. 450-456, 2018.
- [82] M. Rostami, M. Majzoobi, F. Koushanfar, D.S. Wallach, and S. Devadas, "Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computational*, vol. 2, no. 1, pp. 37-49, 2014.
- [83] A. Aysu, E. Gulcan, D. Moriyama, P. Schaumont, M. Yung, "End-to-end design of a PUF-based privacy preserving authentication protocol," in: *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, Saint-Malo, France, 13-16 September 2015; pp. 556-576.
- [84] M.D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Transactions on Multi-Scale Computing Systems.*, vol. 2, issue 3, pp. 146-159, 2016.

- [85] W. Che, M. Martin, G. Pocklassery, V. K. Kajuluri, F. Saqib , and J. Plusquellic, "A privacy-preserving, mutual PUF-based authentication protocol," *Cryptography*, vol. 1, issue 1, article 3, pp. 1-17, 2017.
- [86] P. Gope and B. Sikdar, "Privacy-aware authenticated key agreement scheme for secure smart grid communication," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3953-3962, JULY 2019.
- [87] J.W. Byun, "An efficient multi-factor authenticated key exchange with physically unclonable function," in: *Proceedings of International Conference on Electronics, Information, and Communication (ICEIC)*, Auckland, New Zealand, May 2019, pp. 1-4.
- [88] U. Chatterjee, R.S. Chakraborty, D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Transactions on Embedded Computing Systems*, no. 3, article 67, pp. 1-25, 2017.
- [89] A. Braeken, "PUF based authentication protocol for IoT," *Symmetry*, vol. 10, no. 8, article 352, pp.1-15, 2018.
- [90] M. Agiwal, A. Roy, and N. Saxen, "Next generation 5G wireless networks: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp.1617-1655, 2016.
- [91] S. Borkar and H. Pande, "Application of 5G next generation network to Internet of Things," in: *Proceedings of IoT and Applications (IOTA) Maharashtra Institute of Technology*, Pune, India, Jan 2016, pp.443-447.
- [92] S.K. Rao and R. Prasad, "Impact of 5G technologies on smart city implementation," *Wireless Personal Communications*, vol. 100, pp. 161-176, 2018.
- [93] M.D. Cia, F. Mason, D. Peron, and F. Chiariotti, "Using smart city data in 5G self-organizing networks," *IEEE IoT*, vol. 5, no. 2, pp. 645-654, April 2018.
- [94] R.T. Tiburski, L.A. Amaral, and F. Hessel, "Security challenges in 5G-Based IoT middleware systems," in: Mavromoustakis C., Mastorakis G., Batalla J. (eds) *Internet of Things (IoT) in 5G Mobile Technologies, Modeling and Optimization in Science and Technologies*, vol 8, Springer, Cham, pp. 399-418, April 2016.
- [95] X. Li, A.K. Sangaiah, S. Kumari, F. Wu, J. Shen, and M.K. Khan, "An efficient authentication and key agreement scheme with user anonymity for roaming service in smart city," *Personal and Ubiquitous Computing* , vol. 21, pp. 791-805, 2017.
- [96] J.L. Li, W.G. Zhang, V. Dabra, K.K. R. Choo, S. Kumari, D. Hogrefe, "AEP-PPA: an anonymous, efficient and provably-secure privacy- preserving authentication

- protocol for mobile services in smart cities,” *Journal of Network and Computer Applications*, vol. 134, pp. 52-61, May 2019.
- [97] A.G. Reddy, D. Suresh, K. Phaneendra, J.S. Shin, and V. Odelude, “Provably secure pseudo-identity based device authentication for smart cities environment,” *Sustainable Cities and Society*, vol. 41, pp. 878-885, Aug. 2018.
- [98] Q. Xie and L. Hwang, “Security enhancement of an anonymous roaming authentication scheme with two-factor security in smart city,” *Neurocomputing*, vol. 347, pp. 131-138, 2019.
- [99] S. Jegadeesan, M. Azees, P.M. Kumar, G. Manogaran, N. Chilamkurti, R.Varatharajan, and C.H. Hsu, “An efficient anonymous mutual authentication technique for providing secure communication in mobile cloud computing for smart city applications,” *Sustainable Cities and Society*, vol. 49, no. 101522, 2019.
- [100] Y.M. Tseng, S.S. Huang, and M.L. You, “Strongly secure ID-based authenticated key agreement protocol for mobile multi-server environments,” *International Journal of Communication Systems*, vol. 30, no.11, e3251, 2017.
- [101] C.L. Lei and Y.H. Chuang, “Privacy protection for telecare medicine information systems with multiple servers using a biometric-based authenticated key agreement scheme,” *IEEE Access*, vol.7, pp. 186480-186490, December 2019.
- [102] M. Burrows, M. Abadi, and R. Needham, “A logic of authentication,” *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18-36, Feb. 1990.
- [103] A. Armando et al., “Automated security protocol analysis with the AVISPA tool,” in: *Proceedings of International Conference on Computer Aided Verification (CAV2005)*, Edinburgh, United Kingdom, Jul. 2005, vol. 3576, pp. 281-285.
- [104] D. Wang and P. Wang, “Two birds with one stone: Two-factor authentication with security beyond conventional bound,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp.708-722, 2018.
- [105] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in: *Annual International Cryptology Conference*, Springer, 1999, pp. 388-397.
- [106] T.S. Messerges, E.A. Dabbish, and R.H. Sloan, “Examining smart-card security under the threat of power analysis attacks,” *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541-552, 2002.
- [107] S. Rane, Y. Wang, S.C. Draper, and P. Ishwar, “Secure biometrics: Concepts, authentication architectures, and challenges,” *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 51-64, 2013.

- 
- [108] U. Rührmair et al., “PUF modeling attacks on simulated and silicon data,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876-1891, 2013.
- [109] Y. H. Chuang and C. L. Lei, “An independent three-factor mutual authentication and key agreement scheme with privacy preserving for multi-server environment and a survey,” *International Journal of Communication Systems*, article in press, DOI: 10.1002/dac.4660, Oct. 2020.
- [110] R. C. Bose and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes”, *Information and Control*, vol. 3, no. 1, pp. 68-79, March 1960.
- [111] J. Vliegen, N. Mentens, J. Genoe, A. Braeken, S. Kubera, A. Touhafi, and I. Verbauwhede, “A compact FPGA-based architecture for elliptic curve cryptography over prime fields,” in: *Proceedings of 21st IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2010)*, Rennes, France, 2010, pp. 313-316.
- [112] S. Cavalieri and G. Cutuli, “Implementing encryption and authentication in KNX using Diffie-Hellman and AES algorithms,” in: *Proceedings of 35th Annual Conference of IEEE Industrial Electronics*, IEEE, Porto, Portugal, November 2009, pp. 2459-2464.
- [113] M. Scott, N. Costigan, and W. Abdulwahab, “Implementing cryptographic pairings on smartcards,” in: *Proceedings of Cryptographic Hardware and Embedded Systems*, Springer, Yokohama, Japan, 2006, vol. 4249, pp. 134-147.
- [114] X. Cao, X. Zeng, W. Kou, and L. Hu, “Identity-based anonymous remote authentication for value-added services in mobile networks,” *IEEE Transactions on Vehicular Technology*, vol. 58, no. 7, pp. 3508-3517, September 2009.
- [115] H. Xiong and Z. Qin, “Revocable and scalable certificateless remote authentication protocol with anonymity for wireless body area networks,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1442-1455, 2015.
- [116] G. Alagic et al., “Status report on the second round of the NIST post-quantum cryptography standardization process,” *National Institute of Standards and Technology (NIST)*, <https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf>, July 2020.