

University of Wollongong - Research Online

Thesis Collection

Title: A study on undeniable signatures and their variants

Author: Xinyi Huang

Year: 2009

Repository DOI:

Copyright Warning

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

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material.

Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.

Research Online is the open access repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

University of Wollongong Thesis Collections

University of Wollongong Thesis Collection

University of Wollongong

Year 2009

A study on undeniable signatures and
their variants

Xinyi Huang
University of Wollongong

Huang, Xinyi, A study on undeniable signatures and their variants, PhD thesis, School of Computer Science and Software Engineering, University of Wollongong, 2009.
<http://ro.uow.edu.au/theses/788>

This paper is posted at Research Online.
<http://ro.uow.edu.au/theses/788>

NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

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

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



A Study on Undeniable Signatures and Their Variants

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

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Xinyi Huang

School of Computer Science and Software Engineering
June 2009

© Copyright 2009

by

Xinyi Huang

All Rights Reserved

Dedicated to
My Family

Declaration

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

Xinyi Huang
June 3, 2009

Abstract

In an ordinary digital signature scheme, the verification of a signature requires the associated message, the signer’s public key and other public information (e.g. public parameter). Anyone in the system can verify the validity of the digital signature. This property is useful, as it has many applications. However, it is undesirable for some situations where signer’s privacy is a concern, especially in personally and commercially sensitive applications. In this thesis, we investigate several special signature schemes that accommodate the signer privacy.

In undeniable signatures, the most distinctive feature is that the signer is able to choose who can be convinced about his/her undeniable signature, as the validity of an undeniable signature can only be verified in collaboration with the signer. The property *selective convertibility* enables the signer to convert one or more undeniable signatures into ordinary digital signatures at some time later, while one can make all his/her undeniable signatures publicly verifiable in an undeniable signature scheme with *universal convertibility*. Undeniable signatures with selective and universal convertibility have found many applications in practice such as keeping digital records of confidential political decisions. However, the most known constructions bear a long signature length and some schemes can only be proven secure under strong complexity assumptions. In this thesis, we describe a new undeniable signature scheme with selective and universal convertibility, of which the signature length is the shortest among all comparable ones and the security can be reduced to weaker complexity assumptions. This scheme is considered in the traditional public key infrastructure, where the authenticity of a user’s public key is ensured by certificates. We also provide the first selectively and universally convertible undeniable signature scheme where a user’s public key is his/her identity.

Designated verifier signatures bridge the gap between ordinary digital signatures and undeniable signatures, in the sense that they will limit who can be convinced by the signer’s signature *without* any collaboration with the signer. The designated

verifier can be chosen by the signer in the generation of designated verifier signatures. Although the verification of a designated verifier signature usually needs only public information, only the designated verifier can believe that the designated verifier signature has been generated by the signer. This is due to the fact that the designated verifier is able to generate designated verifier signatures which are indistinguishable from those produced by the signer. Strong designated verifier signatures provide a higher level of privacy, as anyone cannot even verify the validity of strong designated verifier signatures with public information. All known constructions of strong designated verifier signatures have a relatively long signature length and require costly operations, which affect the overall performance of the system. In this thesis, we present two new constructions of strong designated verifier signatures, in traditional public key infrastructure and in identity-based cryptography, respectively. Both schemes have high computational efficiency, short signature length and provable security in the random oracle model.

We finally consider universal designated verifier signatures, which can be viewed as an application of the general idea of designated verifier signatures. This notion was introduced to address the user privacy issue in certification systems, where a certificate holder (or more generally, a signature holder) wishes to generate a proof which can prove to a designated verifier his/her possession of the certificate, but does not want anyone else to be convinced. Universal designated verifier signatures achieve this by giving the designated verifier the full ability to generate that proof. The conviction thus is no longer transferable. In this thesis, we revise the notion of non-transferability in universal designated verifier signatures and give a new definition, which is meaningful both in theory and in practice. Our analysis, however, shows that not all existing schemes have that property. We describe a new universal designated verifier signature scheme, which can be proven secure without random oracles and has the property of non-transferability defined in this thesis. This thesis also investigates another property “delegatability”, which was previously believed as an inherent flaw in universal designated verifier signatures. We show that this problem can be overcome by proposing the first universal designated verifier signature scheme without delegatability.

Acknowledgement

I am most grateful to my supervisor Associate Professor Yi Mu, for his support and guidance of this thesis. He has been providing invaluable suggestions and encouragement from the beginning of my research career. This thesis would have been impossible without his support.

I would like to thank my co-supervisor, Associate Professor Willy Susilo, for his continuous guidance in the process of conducting this research. My thanks also go to Professor Futai Zhang for his advice and support since 2003. I have had helpful discussions and suggestions from many people, a non-exhaustive list of whom includes: Man Ho Au, Xiaofeng Chen, Hua Guo, Fuchun Guo, Shekh Faisal ABDUL LATIP, Jiguo Li, Ching Yu Ng, Angela Piper, Shams Ud Din Qazi, Mohammad Reza Reyhanitabar, Siamak Fayyaz Shahandashti, Pairat Thorncharoensri, Raylin Tso, Rungrat Wiangsripanawan, Duncan S. Wong, Qianhong Wu, Shidi Xu, Yong Yu, Tsz Hon Yuen and Fangguo Zhang, as well as the anonymous referees who reviewed the papers included in this thesis. I would also like to thank all staff of Centre for Computer and Information Security Research and the School of Computer Science and Software Engineering.

I am also grateful to the International Postgraduate Research Scholarships and the University Postgraduate Awards, which were essential in helping me achieve my goals.

I would like to thank my wife Wei Wu, for her patience and love. Without her, this work would never be possible.

Publications

During my PhD studies, I wrote and published the following papers which are related to this thesis.

1. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Secure Universal Designated Verifier Signature without Random Oracles*. International Journal of Information Security 7(3), pages 171-183. Springer, 2008.
2. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. *Short Designated Verifier Signature Scheme and Its Identity-based Variant*. International Journal of Network Security 6(1), pages 82-93. 2008.
3. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Provably Secure Identity-Based Undeniable Signatures with Selective and Universal Convertibility*. In Dingyi Pei, Moti Yung, Dongdai Lin and Chuankun Wu, editors, Information Security and Cryptology, Third SKLOIS Conference, Inscrypt 2007, Lecture Notes in Computer Science 4990, pages 25-39. Springer, 2008.
4. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Delegating Authentication Power in Mobile Networks*. Book Chapter in S. Gritzalis, T. Karygiannis and C. Skianis, editors, Security and Privacy in Wireless and Mobile Computing. Troubador Publishing, 2009.
5. Xinyi Huang, Yi Mu, Willy Susilo and Wei Wu. *Provably Secure Pairing-Based Convertible Undeniable Signature with Short Signature Length*. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto and Takeshi Okamoto, editors, Pairing-Based Cryptography-Pairing 2007, First International Conference, Lecture Notes in Computer Science 4575, pages 367-391. Springer, 2007.
6. Xinyi Huang, Yi Mu, Willy Susilo and Wei Wu. *A Generic Construction for*

- Universally-Convertible Undeniable Signatures*. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang and Chaoping Xing, editors, Cryptology and Network Security, 6th International Conference, CANS 2007, Lecture Notes in Computer Science 4856, pages 15-33. Springer, 2007.
7. Dennis Y. W. Liu, Duncan S. Wong, Xinyi Huang, Guilin Wang, Qiong Huang, Yi Mu and Willy Susilo. *Formal Definition and Construction of Nominative Signature*. In Sihan Qing, Hideki Imai and Guilin Wang, editors, Information and Communications Security, 9th International Conference, ICICS 2007, Lecture Notes in Computer Science 4861, pages 57-68. Springer, 2007.
 8. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Convertible Undeniable Proxy Signatures: Security Models and Efficient Construction*. In Sehun Kim, Moti Yung and Hyung-Woo Lee, editors, Information Security Applications, 8th International Workshop, WISA 2007, Lecture Notes in Computer Science 4867, pages 16-29. Springer, 2008.
 9. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Universal Designated Verifier Signature Without Delegatability*. In Peng Ning, Sihan Qing and Ninghui Li, editors, Information and Communications Security, 8th International Conference, ICICS 2006, Lecture Notes in Computer Science 4307, pages 479-498. Springer, 2006.
 10. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. *Short (Identity-Based) Strong Designated Verifier Signature Schemes*. In Kefei Chen, Robert H. Deng, Xuejia Lai, Jianying Zhou, editors, Information Security Practice and Experience, Second International Conference, ISPEC 2006, Lecture Notes in Computer Science 3903, pages 214-225. Springer, 2006.
 11. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. *Certificateless Designated Verifier Signature Schemes*. In 20th International Conference on Advanced Information Networking and Applications, AINA 2006, pages 15-19. IEEE Computer Society, 2006.
 12. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. *Restricted Universal Designated Verifier Signature*. In Jianhua Ma, Hai Jin, Laurence Tianruo Yang and Jeffrey J. P. Tsai, editors, Ubiquitous Intelligence and Computing,

Third International Conference, UIC 2006, Lecture Notes in Computer Science 4159, pages 874-882. Springer, 2006.

Other Publications.

1. Raylin Tso, Xun Yi and Xinyi Huang. *Efficient and Short Certificateless Signature*. In Matthew K. Franklin, Lucas Chi Kwong Hui and Duncan S. Wong, editors, Cryptology and Network Security, 7th International Conference CANS 2008, Lecture Notes in Computer Science 5339, pages 64-79. Springer, 2008.
2. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Server-Aided Verification Signatures: Definitions and New Constructions*. In Joonsang Baek, Feng Bao, Kefei Chen and Xuejia Lai, editors, Provable Security, Second International Conference, ProvSec 2008, Lecture Notes in Computer Science 5324, pages 141-155. Springer, 2008.
3. Wei Wu, Yi Mu, Willy Susilo and Xinyi Huang. *Certificate-Based Signatures: New Definitions and A Generic Construction from Certificateless Signatures*. In Information Security Applications, 9th International Workshop, WISA 2008, Lecture Notes in Computer Science 5379, pages 99-114, Springer, 2009.
4. Jiguo Li, Xinyi Huang, Yi Mu, Willy Susilo and Qianhong Wu. *Constructions of Certificate-Based Signature Secure against Key Replacement Attacks*. Journal of Computer Security. (Accepted, 2008)
5. Lei Zhang, Futai Zhang and Xinyi Huang. *A Secure and Efficient Certificateless Signature Scheme Using Bilinear Pairing*. Chinese Journal of Electronics, 18(1), pages 145-148. 2009.
6. Jiguo Li, Xinyi Huang, Yi Mu and Wei Wu. *Cryptanalysis and Improvement of An Efficient Certificateless Signature Scheme*. Journal of Communications and Networks 10(1), pages 10-17. 2008.
7. Yong Yu, Chunxiang Xu, Xinyi Huang and Yi Mu. *An Efficient Anonymous Proxy Signature Scheme with Provable Security*. Computer Standards & Interfaces, 31(2), pages 348-353. 2009.

8. Xu'an Wang, Xinyi Huang, and Xiaoyuan Yang. *Further Observations on Certificateless Public Key Encryption*. In Information Security and Cryptology, 4th SKLOIS Conference, Inscrypt 2008. Lecture Notes in Computer Science 5487, pages 217-239, Springer, 2009.
9. Dongdong Sun, Xinyi Huang, Yi Mu and Willy Susilo. *Identity-Based On-line/Off-line Signcryption*. In 2008 IFIP International Conference on Network and Parallel Computing, pages 34-41. IEEE Computer Society, 2008.
10. Xinyi Huang, Yi Mu, Willy Susilo, Duncan S. Wong and Wei Wu. *Certificateless Signature Revisited*. In Josef Pieprzyk, Hossein Ghodosi and Ed Dawson, editors, Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Lecture Notes in Computer Science 4586, pages 308-322. Springer, 2007.
11. Yong Yu, Bo Yang, Xinyi Huang and Mingwu Zhang. *Efficient Identity-Based Signcryption Scheme for Multiple Receivers*. In Bin Xiao, Laurence Tianruo Yang, Jianhua Ma, Christian Müller-Schloer, and Yu Hua, editors, Autonomic and Trusted Computing, 4th International Conference, ATC 2007, Lecture Notes in Computer Science 4610, pages 13-21. Springer, 2007.
12. Wei Wu, Yi Mu, Willy Susilo, Jennifer Seberry and Xinyi Huang. *Identity-Based Proxy Signature from Pairings*. In Bin Xiao, Laurence Tianruo Yang, Jianhua Ma, Christian Müller-Schloer, and Yu Hua, editors, Autonomic and Trusted Computing, 4th International Conference, ATC 2007, Lecture Notes in Computer Science 4610, pages 22-31. Springer, 2007.
13. Jiguo Li, Xinyi Huang, Yi Mu, Willy Susilo and Qianhong Wu. *Certificate-Based Signature: Security Model and Efficient Construction*. In Javier Lopez, Pierangela Samarati and Josep L. Ferrer, editors, Public Key Infrastructure, 4th European PKI Workshop: Theory and Practice, EuroPKI 2007, Lecture Notes in Computer Science 4582, pages 110-125. Springer, 2007.
14. Shidi Xu, Yi Mu, Willy Susilo, Xinyi Huang, Xiaofeng Chen, and Fangguo Zhang. *Efficient Authentication Schemes for AODV and DSR*. Book Chapter in Security in Distributed and Networking Systems, pages 367-390. World Scientific Publishing, 2007.

15. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. *Breaking and Repairing Trapdoor-Free Group Signature Schemes from Asiacrypt'2004*. Journal of Computer Science and Technology 22(1), pages 71-74. Springer, 2007.
16. Xinyi Huang, Willy Susilo, Yi Mu and Wei Wu. *Proxy Signature Without Random Oracles*. In Jiannong Cao, Ivan Stojmenovic, Xiaohua Jia and Sajal K. Das, editors, Mobile Ad-hoc and Sensor Networks, Second International Conference, MSN 2006, Lecture Notes in Computer Science 4325, pages 473-484. Springer, 2006.

Contents

Abstract	v
Acknowledgement	vii
Publications	viii
1 Introduction	1
1.1 Background	2
1.1.1 Undeniable Signatures with Selective and Universal Convert- ibility	2
1.1.2 Strong Designated Verifier Signatures	3
1.1.3 Universal Designated Verifier Signatures	5
1.2 Aims and Objectives	6
1.3 Structure of This Thesis and Contributions	7
1.4 Complexity Problems on \mathbb{Z}_p	8
1.5 Bilinear Maps and Related Complexity Problems	9
2 Convertible Undeniable Signatures with Short Signature Length	12
2.1 Introduction	12
2.2 Definitions of Convertible Undeniable Signatures	16
2.2.1 Difference from Previous Definitions	18
2.2.2 (Strong) Unforgeability of Convertible Undeniable Signatures .	19
2.2.3 (Strong) Invisibility of Convertible Undeniable Signatures . . .	20
2.2.4 (Strong) Anonymity of Convertible Undeniable Signatures . . .	22
2.2.5 Relationship Between Anonymity and Invisibility in Convert- ible Undeniable Signatures	24
2.2.6 Security of S-Convert	25

2.3	The Proposed Scheme	27
2.3.1	The Description of Our Scheme	27
2.3.2	Security Analysis: Confirmation and Disavowal	30
2.3.3	Security Analysis: Unforgeability	32
2.3.4	Security Analysis: Invisibility	34
2.3.5	Security Analysis: S-Convert	39
2.3.6	Comparison with Other Schemes	42
2.4	Conclusion	42
3	Selectively and Universally Convertible Identity-based Undeniable Signatures	43
3.1	Introduction	43
3.2	Definitions of Identity-based Convertible Undeniable Signatures . . .	45
3.2.1	Unforgeability of Identity-based Convertible Undeniable Signatures	47
3.2.2	Invisibility of Identity-based Convertible Undeniable Signatures	48
3.2.3	Security of S-Convert	50
3.3	The Proposed Scheme	51
3.3.1	The Description of Our Scheme	52
3.3.2	Security Analysis: Confirmation and Disavowal	55
3.3.3	Security Analysis: Unforgeability	57
3.3.4	Security Analysis: Invisibility	61
3.3.5	Security Analysis: S-Convert	66
3.4	Conclusion	71
4	Short (Identity-based) Strong Designated Verifier Signatures	72
4.1	Introduction	72
4.2	Definitions of Strong Designated Verifier Signatures	74
4.2.1	Unforgeability of Strong Designated Verifier Signatures	75
4.2.2	Privacy of Signer's Identity	76
4.3	A Short Strong Designated Verifier Signature Scheme	77
4.3.1	The Description of Our Scheme	77
4.3.2	Security Analysis: Unforgeability	78
4.3.3	Security Analysis: Privacy of Signer's Identity	81
4.4	Short Identity-based Strong Designated Verifier Signatures	85

4.4.1	Unforgeability of Identity-based Strong Designated Verifier Signatures	86
4.4.2	Privacy of Signer's Identity in Identity-based Strong Designated Verifier Signatures	87
4.4.3	The Proposed Short Identity-based Strong Designated Verifier Signature Scheme	88
4.4.4	Security Analysis: Unforgeability	89
4.4.5	Security Analysis: Privacy of Signer's Identity	93
4.5	Efficiency Comparison	99
4.6	Conclusion	100
5	New Constructions of Universal Designated Verifier Signatures	101
5.1	Introduction	101
5.2	Definitions of Universal Designated Verifier Signatures	103
5.2.1	Unforgeability of Universal Designated Verifier Signatures	105
5.3	Analysis of A UDVS Scheme without Random Oracles in [ZFI05]	107
5.3.1	Review of Zhang <i>et al.</i> 's [ZFI05] UDVS Scheme	107
5.3.2	Analysis of Non-Transferability	108
5.4	A New Construction of UDVS without Random Oracles	109
5.4.1	The Proposed Scheme	110
5.4.2	Security Analysis: Unforgeability	111
5.4.3	Security Analysis: Non-Transferability	117
5.4.4	Comparison with Other Schemes	118
5.5	Delegatability of (Universal) Designated Verifier Signatures	119
5.5.1	Vergnaud's UDVS-BB [Ver06]	120
5.5.2	Vergnaud's UDVS-BLS [Ver06]	121
5.5.3	Definition of Non-Delegatability	123
5.6	Universal Designated Verifier Signatures without Delegatability	124
5.6.1	Security Analysis: Non-Transferability	126
5.6.2	Security Analysis: Non-Delegatability	126
5.6.3	Security Analysis: Unforgeability	127
5.7	Conclusion	131
6	Conclusions	132
6.1	Undeniable Signatures with Selective and Universal Convertibility	132

6.2	Strong Designated Verifier Signatures	133
6.3	Universal Designated Verifier Signatures	134
	Bibliography	136

List of Tables

2.1	Existing Undeniable Signature Schemes with Convertibility	15
2.2	Comparison with A Pairing-based Scheme [LV05b]	42
4.1	Comparison with Two Efficient DVS Schemes [SKM03, LV04a]	100
4.2	Comparison with An ID-based DVS Scheme [SZM04]	100
5.1	Comparison with A UDVS Scheme Without Random Oracles [Ver06]	119

Chapter 1

Introduction

In a public key cryptography [DH76], a user is equipped with a pair of cryptographic keys: a public key and a private key. The private key is kept secret, while the public key may be widely distributed and published. It is computationally infeasible to derive the private key from the public key. One of the most important applications of public key cryptography is digital signatures. Digital signatures are equivalent to traditional handwritten signatures in many aspects. In particular, a digital signature generated by a user on a message guarantees that it was that user who signed the message and the message cannot be altered. In a public key cryptography, a digital signature scheme typically consists of three algorithms: a *key generation* algorithm that outputs a private key and the corresponding public key; a *signing* algorithm that, given a message and a private key, produces a signature on that message; and a *verifying* algorithm that verifies the validity of a message-signature pair under a given public key.

In ordinary digital signatures, the *verifying* algorithm only uses the signer's public key to check the validity of a message-signature pair. Thus, once given a valid message-signature pair under a user's public key, anyone will believe that user's commitment on the message. This property, together with the ease of copying and transmitting digital signatures, has great convenience in the implementation of verifying one's public announcement. However, this would be undesirable in other personally or commercially sensitive applications where signatures must not be publicly verifiable and signer's privacy is a concern. In this thesis, we will describe several special signature schemes which accommodate that property.

1.1 Background

This section describes the notion of undeniable signatures, and their variants—designated verifiable signatures and universal designated verifier signatures.

1.1.1 Undeniable Signatures with Selective and Universal Convertibility

The concept of undeniable signatures was introduced by Chaum and van Antwerpen [CA89] at Crypto'89. Like ordinary digital signatures, undeniable signatures can be created by the signer with the private key. The most distinctive feature in undeniable signatures is that the validity of undeniable signatures can only be tested in collaboration with the signer. In other words, the knowledge of the public key alone does not provide sufficient information to verify undeniable signatures. This property is useful in situations where the validity of a signature must not be publicly verifiable. For example, a software vendor must embed a signature into his/her product to assure customers that the software is provided by the vendor. Suppose, after the purchase, the customer (say, Alice) obtains the software and its ordinary digital signature from the vendor. Alice can then make several copies and sell them at a lower price, as Alice can prove that the software is from the vendor by presenting the vendor's signature on the product. This is because the vendor signs products with an ordinary digital signature scheme, and anyone who has the vendor's public key can verify vendor's signature. This problem can be solved by undeniable signatures where anyone cannot verify the validity of vendor's signatures without the cooperation from the vendor. In this case, the vendor can prove only to the payer about the validity of his/her undeniable signatures. Apart from algorithms *key generation*, *signing* and *verifying* as in an ordinary digital signature scheme, there are two new protocols, namely *confirmation* and *disavowal*, in an undeniable signature scheme. The *confirmation* protocol can be used by the signer to convince the signature recipient that an undeniable signature on a message was produced by himself/herself, while *disavowal* protocol is used to prove that an undeniable signature is not produced by himself/herself. If an undeniable signature is produced by the signer, anyone should not be able to deny it by using *disavowal* protocol.

In addition to properties of undeniable signatures described previously, it could

be useful if there was some secret information the signer can issue at some time later and would convert undeniable signatures into ordinary digital signatures. Thus, these signatures could be verified without the help of the signer, but they must be secure against forgery. Undeniable signatures with such an additional property are called *convertible undeniable signatures*, which was introduced by Boyar, Chaum, Damgård and Pedersen [BCDP90]. In a software company using convertible undeniable signatures, the employees of the company can be given the ability to verify undeniable signatures, without being able to produce valid signatures. Furthermore, if the company later goes bankrupt, the manager can release the secret information needed to verify signatures, and hence the software can still be used safely. At this point, the company would no longer be protected against the piracy of its softwares, but it would no longer care much about it either [BCDP90]. Convertible undeniable signatures have also been found useful in applications such as the problem of keeping digital records of confidential political decisions [DP96]. Convertible undeniable signatures have two types: *selectively convertible* and *universally convertible*. An undeniable signature scheme is said to be selectively convertible if the signer can issue *selective proofs* (also known as *individual proofs*), which can convert one or more undeniable signatures into ordinary signatures. An undeniable signature scheme is said to be universally convertible if the signer can issue a *universal proof* which will convert all his/her undeniable signatures into ordinary ones. Thus, one is able to verify the validity of all undeniable signatures without the cooperation of the signer.

1.1.2 Strong Designated Verifier Signatures

As an application of undeniable signatures, the software vendor embeds signatures on his/her products to assure the customer that the products are correct, free of viruses, etc. The software vendor only wants the payer to be able to verify the validity of his/her signatures, and must know to whom he/she is proving the validity of an undeniable signature because of blackmailing [DY91, Jak94] and mafia attacks¹ [DGB87]. In such attacks, several cooperating verifiers can believe the validity of the signature by jointly setting the challenge in a way such that none of them can determine the outcome. The conviction thus can be transferred to one or more hidden co-verifiers while the prover is not able to be aware of it. E-voting

¹also known as the man-in-the middle attack.

is another application where the prover must designate who can be convinced by the validity of a vote. More precisely, the voting centre wants only the voter (say, Daniel) to be assured that the vote he cast was counted, but prevents an armed coercer from forcing Daniel to prove how he voted.

Jakobsson, Sako and Impagliazzo suggested a solution *designation of verifiers* [JSI96] to resolve the conflict between authenticity and privacy by limiting who can be convinced by a proof. The intuition behind the solution can be described in one sentence: *Instead of proving Θ , the prover will prove the statement “Either Θ is true, or I am Cindy”* [JSI96]. Cindy will certainly trust this proof upon seeing such a proof. However, if Cindy transfers this proof to Bob, Bob will have no reason at all to believe that Θ is true, as Cindy is fully capable of generating such proof by proving herself to be Cindy even Θ is not true. In other words, there are two parties who can generate the proof, and given a proof, one cannot distinguish who is the proof generator. The solution to generate a proof with a designated verifier in [JSI96] is trapdoor commitment schemes (also known as chameleon commitment schemes) introduced by Brassard, Chaum and Crépeau [BCC88], where the trapdoor is some secret information of Cindy. If the prover commits to a value, Cindy will trust the commitment as the prover cannot find collisions. However, if Cindy forwards the commitment to some other party Bob, Bob will not believe that as Cindy has the trapdoor and can decommit it. The designation of a verifier still holds even the verifier shares his/her secret information with other parties, as the designated verifier is still able to generate the proof after revealing secret information.

In [JSI96], the notion of designation of verifiers was originally introduced in the scenario of undeniable signatures to serve as the confirmation protocol, which is interactive. Jakobsson, Sako and Impagliazzo also introduced a non-interactive designated verifier proof, which is called designated verifier signatures (or, DVS for short). The designated verifier of a DVS can trust the signer’s commitment on a message, but cannot make any other party believe that, as he/she can also generate a DVS designated to himself/herself. Therefore, DVS does not provide the main property of ordinary digital signatures, namely “non-repudiation”. However, DVS bridges the gap between ordinary digital signatures and undeniable signatures, in the sense that it limits who can be convinced about it *without* the cooperation from the signer [JSI96]. A DVS scheme can be constructed from a ring signature scheme introduced by Rivest, Shamir and Tauman in [RST01], where two participants in a ring signature collaborate and generate a signature.

In a DVS, it could be possible to check that there are only two potential signature generators. One example is designated verifier signature schemes where the verification only requires the public information of the signer and the verifier. If signatures are captured on the line before reaching the verifier, an eavesdropper will believe that the designated verifier did not produce the signature. Therefore, Jakobsson *et al.* [JSI96] suggested a stronger notion *strong designated verifier signature*, which requires the “privacy of signer’s identity” (also known as “strongness” of designated verifier signatures). This property requires that given a designated verifier signature and two potential signing public keys, it is *computationally infeasible* for an eavesdropper to determine under which of the two corresponding secret keys the signature was performed. A generic construction of strong designated verifier signatures was introduced in [LV04a], namely, a strong designated verifier signature scheme can be obtained by applying an additional IND-CCA2 public-key encryption to a designated verifier signature scheme.

1.1.3 Universal Designated Verifier Signatures

The notion of universal designated verifier signatures (or, UDVS for short) was introduced by Steinfeld, Bull, Wang and Pieprzyk [SBWP03] to address the user privacy issue in user certification systems.

In these systems, a (trusted) third party sends a signed certificate (e.g., birth certificates, driving licences and academic transcripts) to a user Anna. Anna can present her certificate to any interested verifier Paul, who can verify the third party’s signature and become convinced of the truth of the statements contained in the certificate. In an electronic world, the ease of copying and transmitting electronic signatures may make the verifier Paul easily disseminate Anna’s certificate and make an unlimited number of verifiers believe the statement on that certificate. This possibility poses a serious threat to Anna, especially when the certificate contains the private information of Anna. Once Anna sends out the signed certificate to Paul, she no longer has any control over the number of entities besides Paul who can not only learn all the statements about Anna contained in the certificate, but also become convinced about the truth of these statements by verifying the third party’s signature on the certificate [SBWP03].

In a universal designated verifier signature scheme, a user Anna has an ordinary signature of a message produced by the signer. Anna is also called as the “signature

holder” in universal designated verifier signatures. To prove to a verifier Paul the possession of that signature, Anna does not need to send the signature to Paul. Instead, Anna will use Paul’s public key to transform that signature into a UDVS, and sends the message along with the UDVS to Paul. Upon receiving the UDVS and the message, Paul will believe that the signer has signed the message, but is unable to use this UDVS to convince any other party about that, even if Paul is willing to reveal his private key to that party. This is achieved because Paul’s private key allows him to create valid UDVSs by himself, and therefore, no one is able to tell who produced the UDVS (whereas Paul can, because he knows that he did not produce it). Thus, by incorporating a UDVS scheme, the user Anna’s privacy is preserved as Paul is unable to disseminate convincing statements about Anna. In this sense, UDVS is quite similar to DVS as described previously. The difference is that in a UDVS scheme any signature holder is able to designate the signature to a verifier, while in a DVS scheme the designation can only be performed by the signature signer. More precisely, if the signature holder and signer are the same user, a universal designated verifier signature will be a designated verifier signature. Therefore, universal designated verifier signatures can be viewed as an application of the general idea of designated verifier signatures.

1.2 Aims and Objectives

This thesis focuses on undeniable signatures, and their variants, designated verifier signatures and universal designated verifier signatures. The aim of this thesis addresses three aspects defined as follows.

1. In the literature, several constructions of selectively and universally convertible undeniable signatures have been proposed. However, most of them have a long signature length and the security of others can only be reduced to strong complexity assumptions. The first aim of this thesis is to construct an undeniable signature scheme with selective convertibility, universal convertibility, short signature length and provable security based on weak complexity assumptions. In addition to that, we will also investigate the construction of selectively and universally convertible undeniable signatures in identity-based public key cryptography, where a user’s public key is his/her identity.
2. The second aim of this thesis is to find alternative methods of constructing

strong designated verifier signatures with short signature length. All previous constructions have a relatively long signature length which limits their application in practice, especially in low bandwidth environments. Meanwhile, existing constructions of strong designated verifier signatures require costly operations. This affects the overall performance of systems they are applied to. In this thesis, we want to design new constructions of strong designated verifier signatures with a shorter signature length and less computational cost.

3. Universal designated verifier signatures with new properties is another aim of this thesis. More precisely, we will first investigate the notion of non-transferability. This notion requires that universal designated verifier signatures can convince only designated verifiers. We will investigate its meaning both in theory and in practice, and give a new reasonable definition. After that, we consider the property delegatability, which was believed as an inherent problem in universal designated verifier signatures. The aim is to find new techniques to construct universal designated verifier signatures without delegatability.

1.3 Structure of This Thesis and Contributions

Chapter 1 briefly describes the background of this thesis and gives a short introduction of each chapter and its contribution. The mathematical tools and complexity assumptions required in this thesis are also reviewed in the same chapter.

Chapter 2 focuses on undeniable signatures with selective and universal convertibility. We first define the syntax and the security of undeniable signatures with selective and universal convertibility, and compare them to prior definitions. The new definition in this chapter can reflect the essence of undeniable signatures with selective and universal convertibility. After that, we describe a new construction of undeniable signatures with selective and universal convertibility. The signature length of the proposed scheme is as short as Boneh-Lynn-Shacham signature [BLS01], the shortest among all comparable ones. The security analysis of the proposed scheme is provided in the random oracle model. Its unforgeability can be reduced to the hardness of Computational Diffie-Hellman problem, and its invisibility can be reduced to the hardness of 3-Decisional Diffie-Hellman problem, which is a natural extension of the classic Decisional Diffie-Hellman problem on bilinear groups. The

original scheme described in Chapter 2 was presented at *Pairing 2007* [HMSW07].

In Chapter 3, we describe a new construction of identity-based undeniable signatures. Our construction is the first selectively and universally convertible identity-based undeniable signature scheme where there is no certificate and a user's public key is his/her identity. Our construction is based on the bilinear mapping over elliptic curves. Its unforgeability can be reduced to the hardness of Computational Diffie-Hellman problem in the random oracle model, and its invisibility can be reduced to the hardness of Decisional Bilinear Diffie-Hellman problem. The original scheme described in this chapter was presented at *Inscript 2007* [WMSH07].

In Chapter 4, we investigate the notion of strong designated verifier signatures and present two new constructions. Our first construction is proposed in the traditional public key infrastructure where the authenticity of a user's public key is ensured by the certificate. Our second construction belongs to the identity-based cryptography, where a user's public key is his/her identity. Both constructions have a shorter signature length and less computational cost than comparable ones. The original schemes described in this chapter were presented at *ISPEC 2006* [HSMZ06b] and the full version of [HSMZ06b] was published in the *International Journal of Network Security* [HSMZ08].

Chapter 5 describes two new constructions of universal designated verifier signatures. Our first construction in Section 5.4 was published in the *International Journal of Information Security* [HSMW08]. It can be proven secure without random oracles and has the property of unconditional non-transferability defined in the same Chapter. Our second construction was presented at *ICICS 2006* [HSMW06]. It is the first universal designated verifier signature scheme with non-delegatability property.

Chapter 6 concludes this thesis.

1.4 Complexity Problems on \mathbb{Z}_p

Let p be a large prime and q a prime divisor of $p - 1$. Let G be a subgroup in \mathbb{Z}_p^* with prime order q . The generator of G is denoted as g . We say that (g, g^a, g^b, g^c) is a DH-tuple on G if $\mathbf{c} = \mathbf{a}\mathbf{b} \pmod{q}$.

Decisional Diffie-Hellman Oracle \mathcal{O}_{DDH} on G . Given (g, g^a, g^b, g^c) , this oracle

outputs “1” if it is a DH-tuple or “0” otherwise².

Gap Diffie-Hellman Problem on G . Given (g, g^a, g^b) , compute g^{ab} with the help of Decisional Diffie-Hellman Oracle \mathcal{O}_{DDH} .

An algorithm \mathcal{B} solves the Gap Diffie-Hellman problem on G with advantage ε if $\text{Adv GDH}_{\mathcal{B}} = \Pr[\mathcal{B}(g, g^a, g^b, \mathcal{O}_{\text{GDH}}) = g^{ab} : \mathbf{a}, \mathbf{b} \in_R \mathbb{Z}_q^*] \geq \varepsilon$. The probability is over the uniform random choice of \mathbf{a}, \mathbf{b} from \mathbb{Z}_q^* , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -breaks Gap Diffie-Hellman problem if \mathcal{B} runs in time at most t , and $\text{Adv GDH}_{\mathcal{B}}$ is at least ε .

Definition 1.1 *The Gap Diffie-Hellman (GDH) on G is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -breaks it.*

1.5 Bilinear Maps and Related Complexity Problems

We briefly review the necessary facts about bilinear maps and groups, in the notation of [BLS04]:

- (\mathbb{G}_1, \cdot) , (\mathbb{G}_2, \cdot) , (\mathbb{G}_T, \cdot) are three cyclic groups of prime order p ;
- g_1 is a generator of \mathbb{G}_1 and g_2 is a generator of \mathbb{G}_2 ;
- ψ is an isomorphism from \mathbb{G}_2 to \mathbb{G}_1 , with $\psi(g_2) = \psi(g_1)$; and
- e is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, i.e., a map satisfying the following properties:

1. Bilinearity: $\forall v_1 \in \mathbb{G}_1$ and $\forall v_2 \in \mathbb{G}_2$, $\forall a, b \in \mathbb{Z}$, $e(v_1^a, v_2^b) = e(v_1, v_2)^{ab}$;
2. Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ and is thus a generator of \mathbb{G}_T ; and
3. Efficiency: All group operations and the bilinear map must be efficiently computable.

Definition 1.2 *We say that $(\mathbb{G}_1, \mathbb{G}_2)$ are a bilinear group pair if there exists a group \mathbb{G}_T and a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, such that the group order $p = |\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$ is prime, and the pairing e and the group operations in $\mathbb{G}_1, \mathbb{G}_2$, and \mathbb{G}_T are all efficiently computable.*

²Throughout this thesis, fonts in mathfrak style (e.g. $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$) will indicate that they are associated with complexity problems.

Discrete Logarithm (DL) on \mathbb{G}_2 . Given $g_2, g_2^a \in \mathbb{G}_2$, compute $a \in \mathbb{Z}_p$.

An algorithm \mathcal{B} solves the DL problem on \mathbb{G}_2 with advantage ε if $\text{Adv DL}_{\mathcal{B}} = \Pr[\mathcal{B}(g_2, g_2^a) = a : a \in_R \mathbb{Z}_p] \geq \varepsilon$. The probability is over the uniform random choice of a from \mathbb{Z}_p , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -breaks DL on \mathbb{G}_2 if \mathcal{B} runs in time at most t , and $\text{Adv DL}_{\mathcal{B}}$ is at least ε .

Definition 1.3 *The DL problem on \mathbb{G}_2 is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -breaks it.*

Computational co-Diffie-Hellman (co-CDH) on $(\mathbb{G}_1, \mathbb{G}_2)$. Given $g_2, g_2^a \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$ as input, compute $h^a \in \mathbb{G}_1$.

An algorithm \mathcal{B} solves the co-CDH problem on the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage ε if $\text{Adv co-CDH}_{\mathcal{B}} = \Pr[\mathcal{B}(g_2, g_2^a, h) = h^a : a \in_R \mathbb{Z}_p, h \in_R \mathbb{G}_1] \geq \varepsilon$. The probability is over the uniform random choice of a from \mathbb{Z}_p and h from \mathbb{G}_1 , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -breaks co-CDH on $(\mathbb{G}_1, \mathbb{G}_2)$ if \mathcal{B} runs in time at most t , and $\text{Adv co-CDH}_{\mathcal{B}}$ is at least ε . When $\mathbb{G}_1 = \mathbb{G}_2$, co-CDH reduces to standard CDH problem.

Definition 1.4 *The co-CDH problem on $(\mathbb{G}_1, \mathbb{G}_2)$ is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -breaks it.*

Decisional co-3-Diffie-Hellman (co-3-DDH) on $(\mathbb{G}_1, \mathbb{G}_2)$. Given $g_2, g_2^a, g_2^b \in \mathbb{G}_2$ and $h, h^c \in \mathbb{G}_1$ as input, output “1” if $c = ab$ and “0”, otherwise. When the answer is “1” we say that $(g_2, g_2^a, g_2^b, h, h^c)$ is a co-3-Diffie-Hellman tuple.

An algorithm \mathcal{B} solves the co-3-DDH problem on the bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with advantage ε if

$$\begin{aligned} \text{Adv co-3-DDH}_{\mathcal{B}} = & \left| \Pr[\mathcal{B}(g_2, g_2^a, g_2^b, h, h^c) = 1 : a, b, c \in_R \mathbb{Z}_p, c = ab] \right. \\ & \left. - \Pr[\mathcal{B}(g_2, g_2^a, g_2^b, h, h^c) = 1 : a, b \in_R \mathbb{Z}_p, c \in_R \mathbb{Z}_p \setminus \{ab\}] \right| \geq \varepsilon. \end{aligned}$$

The probability is over the uniform random choice of a, b, c from \mathbb{Z}_p and h from \mathbb{G}_1 , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -breaks co-3-DDH on $(\mathbb{G}_1, \mathbb{G}_2)$ if \mathcal{B} runs in time at most t , and $\text{Adv co-3-DDH}_{\mathcal{B}}$ is at least ε . When $\mathbb{G}_1 = \mathbb{G}_2$, co-3-DDH reduces to the xyz-DDH problem defined in [LV05b].

Definition 1.5 *The co-3-DDH problem on $(\mathbb{G}_1, \mathbb{G}_2)$ is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -breaks it.*

In some parts of this thesis, we consider a special case of bilinear map where $\mathbb{G}_1 = \mathbb{G}_2$ and $g_1 = g_2$ as in [BF01]. Note that $e(,)$ defined in this way is symmetric since $e(g_1^a, g_1^b) = e(g_1, g_1)^{ab} = e(g_1^b, g_1^a)$. We say that $(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h})$ is a BDH-tuple if $\mathfrak{h} = e(g_1, g_1)^{abc}$.

Decisional Bilinear Diffie-Hellman (DBDH) on $(\mathbb{G}_1, \mathbb{G}_T)$. Given $(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h})$, this oracle outputs “1” if it is a BDH-tuple or “0” otherwise. An algorithm \mathcal{B} solves the DBDH problem on $(\mathbb{G}_1, \mathbb{G}_T)$ with advantage ε if

$$\begin{aligned} \text{Adv DBDH}_{\mathcal{B}} = & \left| \Pr[\mathcal{B}(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h}) = 1 : \mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in_R \mathbb{Z}_p, \mathfrak{h} = e(g_1, g_1)^{abc}] \right. \\ & \left. - \Pr[\mathcal{B}(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h}) = 1 : \mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in_R \mathbb{Z}_p, \mathfrak{h} \in_R \mathbb{G}_T \setminus \{e(g_1, g_1)^{abc}\}] \right| \geq \varepsilon. \end{aligned}$$

The probability is over the uniform random choice of $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ from \mathbb{Z}_p and \mathfrak{h} from \mathbb{G}_T , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -breaks DBDH on $(\mathbb{G}_1, \mathbb{G}_T)$ if \mathcal{B} runs in time at most t , and $\text{Adv DBDH}_{\mathcal{B}}$ is at least ε .

Definition 1.6 *The DBDH problem on $(\mathbb{G}_1, \mathbb{G}_T)$ is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -breaks it.*

Decisional Bilinear Diffie-Hellman Oracle $\mathcal{O}_{\text{DBDH}}$ on $(\mathbb{G}_1, \mathbb{G}_T)$. Given $(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h})$, this oracle outputs “1” if it is a BDH-tuple or “0” otherwise.

Gap Bilinear Diffie-Hellman Problem on $(\mathbb{G}_1, \mathbb{G}_T)$. Given $(g_1, g_1^a, g_1^b, g_1^c)$, compute $e(g_1, g_1)^{abc}$ with the help of Decisional Bilinear Diffie-Hellman Oracle $\mathcal{O}_{\text{DBDH}}$.

An algorithm \mathcal{B} solves the Gap Bilinear Diffie-Hellman problem on $(\mathbb{G}_1, \mathbb{G}_T)$ with advantage ε if $\text{Adv GBDH}_{\mathcal{B}} = \Pr[\mathcal{B}(g_1, g_1^a, g_1^b, g_1^c, \mathcal{O}_{\text{DBDH}}) = e(g_1, g_1)^{abc} : \mathfrak{a}, \mathfrak{b}, \mathfrak{c} \in_R \mathbb{Z}_p] \geq \varepsilon$. The probability is over the uniform random choice of $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}$ from \mathbb{Z}_p , and over the coin tosses of \mathcal{B} . We say that an algorithm \mathcal{B} (t, ε) -breaks Gap Bilinear Diffie-Hellman problem if \mathcal{B} runs in time at most t , and $\text{Adv GBDH}_{\mathcal{B}}$ is at least ε .

Definition 1.7 *The Gap Bilinear Diffie-Hellman (GBDH) problem on $(\mathbb{G}_1, \mathbb{G}_T)$ is said to be (t, ε) -hard if there exists no algorithm that (t, ε) -breaks it.*

Chapter 2

Convertible Undeniable Signatures with Short Signature Length

This chapter describes a new construction of undeniable signatures with selective convertibility, universal convertibility and short signature length. The original scheme was presented at *Pairing 2007* [HMSW07].

2.1 Introduction

In Crypto'89, Chaum and van Antwerpen [CA89] proposed the notion of undeniable signatures, whose most distinctive feature is that a user's undeniable signatures can only be verified in collaboration with the user. In other words, the knowledge of the public key alone does not provide sufficient information to verify undeniable signatures. This property is useful in situations where the validity of a signature must not be publicly verifiable. For example, a software vendor might want to embed undeniable signatures into his/her products and allow only payers to check the authenticity of products. If the vendor has actually signed the software, he/she must be able to prove it to the customer by the *confirmation* protocol. Otherwise, the vendor must also be able to prove that he/she is not the signer via the *disavowal* protocol. Such proofs must be *non-transferable*; that is, once a verifier is convinced about the validity of undeniable signatures, he/she should be unable to transmit the conviction to another third party.

In the past 20 years, a number of undeniable signature schemes were published. The first undeniable signature scheme was proposed by Chaum and van Antwerpen [CA89] and it was further improved by Chaum [Cha90] in Euocrypt'90. Yet, the unforgeability of the FDH (full domain hash) variant of Chaum's scheme remained as an open problem until the introduction of a new class of computational problems: Gap problems. Okamoto and Pointcheval [OP01] proved that the unforgeability of

the FDH variant of Chaum's scheme is equivalent to the hardness of Gap Diffie-Hellman problem. Unfortunately, Ogata, Kurosawa and Heng [OKH05, OKH06] pointed out the flaw in the security proof in [OP01], and proved that the unforgeability is actually equivalent to the hardness of Computational Diffie-Hellman problem. In Eurocrypt 2005, Kurosawa and Heng [KH05] proposed the first 3-move confirmation and disavowal protocols for Chaum's undeniable signature scheme which is claimed to be secure against active and concurrent attacks. This was later found incorrect as a cheating verifier is able to show the third party some evidence which can prove an undeniable signature is valid [OKH06]. Due to the Σ -compiler proposed by Furukawa, Kurosawa and Imai [FKI06], one can obtain a very efficient Chaum's undeniable signature with a 2-move confirmation protocol and a disavowal protocol which are concurrent deniable zero-knowledge as well. Other researches about undeniable signatures can be found in [BCDP90, BPT04, DP96, FOO91, GMP02, GM03, GKR97, GRK00, JSJK01, KW04, LQ04, LW02, LV05b, Miy00, MPH96, MV04b, Wan03, WQWZ01, WZD04, ZSNS03].

Boyar, Chaum, Damgård and Pedersen [BCDP90] introduced the concept of convertible undeniable signatures, allowing the signer to convert one or more his/her undeniable signatures into publicly verifiable ones. "Convert" in undeniable signatures has two types: *selectively convert* and *universally convert*. The signer can use the *selectively convert* algorithm to generate a *selective proof* (also known as *individual proof*) for an undeniable signature, whose validity can be verified with the signer's public key and that proof. The validity of other undeniable signatures still remains unknown and can only be verified via the *confirmation/disavowal* protocol with the aid of that signer. On the other hand, the signer can generate a universal proof which will make all his/her undeniable signatures publicly verifiable. One, thus, is able to verify the validity of any undeniable signatures of this signer without the aid from the latter. Convertible undeniable signatures have been found useful in applications such as the problem of keeping digital records of confidential political decisions [DP96]. The first convertible undeniable signature scheme proposed in [BCDP90] has been broken by Michels, Petersen and Horster [MPH96], who proposed a repaired version with heuristic security¹. In Eurocrypt'96, Damgård and Pedersen [DP96] proposed two convertible undeniable signature schemes, in which

¹Very recently, Aimani and Vergnaud [AV07] provided the proof of Michels-Petersen-Horster's scheme [MPH96] in the generic group model.

forging signatures is provably equivalent to forging the El Gamal signature. An efficient convertible undeniable signature based on Schnorr signature was proposed by Michels and Stadler in [MS97]. The new scheme can be used as a basis of an efficient extension to threshold signature. Other constructions in RSA systems were also introduced. The first RSA based (convertible) undeniable signature was proposed by Gennaro, Rabin and Krawczyk in CRYPTO'97 [GKR97], which was later improved by Miyazaki [Miy00]. Very recently, Kurosawa and Takagi [KT06] proposed a new approach for constructing selectively convertible undeniable signatures, and presented two schemes based on RSA related assumptions. Kurosawa and Takagi's second scheme is the first selectively convertible undeniable signature scheme which can be proven secure without random oracles. Based on the computation of characters, Monnerat and Vaudenay proposed a novel construction of undeniable signatures which offers the advantage of having an arbitrarily short signature (depending on the required security level) [MV04b]. Monnerat and Vaudenay generalized and optimized their scheme in [MV04a] and [MOV05], respectively, and mentioned that their scheme proposed in [MV04a] can provide the selective convertibility. Laguillaumie and Vergnaud proposed a new convertible undeniable signature scheme from pairings [LV05b]. The signature of their scheme only consists of an element of the group in elliptic curve and some additional random salt such that the total signature length can be as short as 272 bits. Very recently, a convertible undeniable signature scheme without random oracles was proposed in [YALS07], which uses more standard assumption than another undeniable signature scheme without random oracles in [LV05a]. The Table 2.1 summarizes the known undeniable signature schemes with selective convertibility (**SC**) and/or universal convertibility (**UC**).

The scheme proposed in this chapter is motivated by the following observations.

1. In undeniable signatures, the additional property “Convertible” enables the signer to generate selective or universal proofs which can convert his/her undeniable signatures into publicly verifiable ones. The signer thus does not need to prove to each verifier of the validity/invalidity of his/her undeniable signatures. Such proofs, however, could also provide extra information which might help the adversary to break the undeniable signature scheme. One example of this is Boyar-Chaum-Damgård-Pedersen's undeniable signature scheme [BCDP90]. Their scheme will become insecure after the signer publishes the universal proof, which will enable a key only adversary to forge

Table 2.1: Existing Undeniable Signature Schemes with Convertibility

Scheme	SC	UC	Security
[BCDP90]	✓	✓	Broken [MPH96]
[DP96]	✓	✓	DL related assumptions
[MPH96]	✓	✓	Proofs in generic group model
[MS97]	✓	✓	Sketchy proof
[GRK00]	✓	✓	RSA related assumptions
[Miy00]	✓	✓	Sketchy proof
[MV04a]	✓		GHI related assumptions [MV04a]
[LV05b]	✓	✓	Pairing related assumptions
[KT06]	✓		RSA related assumptions
[YALS07]	✓	✓	Pairing related assumptions
[Aim08]		✓	Generic Construction

their scheme universally [MPH96]. It is therefore crucial to define the security of convertible undeniable signatures. Although there exist some security models of convertible undeniable signatures [DP96, KT06, LV05b, MS97], some properties still remain unclear. The relationship between Anonymity and Invisibility, for example, has been proven closely related in the notion of undeniable signatures [GM03]. Yet, there is no such analysis about their relationship in convertible undeniable signatures. In this chapter, we will provide security models to define the security properties and their relationships in convertible undeniable signatures.

2. It is also interesting to note that the existing convertible undeniable signature schemes are not all both selectively and universally convertible. The schemes in [Miy00, MPH96, MS97] are both selectively and universally convertible, but their proofs are not satisfactory. There are only four schemes [DP96, GRK00, LV05b, YALS07] which have formal proofs, and meanwhile can provide both selective and universal convertibility. Among such schemes, Laguillaumie-Vergnaud's scheme [LV05b] is based on bilinear mappings and has the shortest signature length. Their scheme is unforgeable under the Computational Diffie-Hellman assumption. However, its invisibility is based on a non-standard decisional assumption: $(\ell, 1)$ -xyz-DCAA assumption introduced in [LV05b]. In this chapter, we will construct a new undeniable signature scheme with shorter signature length and provable security (in the random oracle model)

under relatively weaker assumptions. Our scheme can be viewed as the analogue of Chaum's undeniable signature scheme, but with the improvement of both selective and universal convertibility. The new construction has the shortest signature length among the comparable ones but requires the highest computational cost due to the use of bilinear maps.

Organization of This Chapter. The rest of this chapter is organized as follows. In the next section, we will review the definitions of convertible undeniable signatures. The new construction of convertible undeniable signatures is described in Section 2.3, where the security analysis of the proposed scheme is also provided. Section 2.4 concludes this chapter.

2.2 Definitions of Convertible Undeniable Signatures

This section will define the syntax and the security of convertible undeniable signatures, and relate them to prior definitions.

Definition 2.1 *A selectively and universally convertible undeniable signature scheme Σ is made up of seven algorithms, KeyGen, Sign, Verify, Confirmation, Disavowal, S-Convert, and S-Verify. For a fixed security parameter $param$, these algorithms work as follows:*

$\text{KeyGen}(param) \rightarrow (ssk, svk, pk)$.

This algorithm takes $param$ as input and returns the secret key and the public key. The secret key consists of a *secret signing key* ssk and a *secret verification key* svk . The public key is denoted as pk , which includes the description of the message space \mathcal{M} and the signature space \mathcal{S} . The user can publish the secret verification key svk to make all his/her undeniable signatures publicly verifiable with the algorithm **Verify** defined below.

$\text{Sign}(param, ssk, pk, m) \rightarrow \sigma$.

This algorithm takes $param$, a secret signing key ssk , a public key pk and a message $m \in \mathcal{M}$ as inputs, and returns an element $\sigma \in \mathcal{S}$.

$\text{Verify}(param, svk, pk, m, \sigma) \rightarrow \{\text{valid}, \text{invalid}, \perp\}$.

This algorithm takes $param$, a secret verification key svk , a public key pk

and a message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ as inputs, and returns **valid**, **invalid** or a symbol “ \perp ”. Here, “ \perp ” indicates that svk does not contain enough information to prove the validity (or, invalidity) of σ . This could happen, for example, when svk is not the secret verification key of the public key pk . A message-signature pair (m, σ) is said to be valid under pk if $\text{Verify}(param, svk, pk, m, \sigma) = \text{valid}$. Similarly, (m, σ) is said to be invalid under pk if $\text{Verify}(param, svk, pk, m, \sigma) = \text{invalid}$.

Confirmation $(param, pk, m, \sigma, svk) \rightarrow \{\text{valid}, \perp\}$.

This is a zero-knowledge proof system between a prover and a verifier on valid message-signature pairs (m, σ) under pk . The common inputs of both parties are $(param, pk, m, \sigma)$, and the prover has a secret verification key svk as an additional input. The verifier returns **valid** if the prover can prove (m, σ) is valid under pk . Otherwise, the verifier outputs “ \perp ”.

Disavowal $(param, pk, m, \sigma, svk) \rightarrow \{\text{invalid}, \perp\}$.

This is a zero-knowledge proof system between a prover and a verifier on invalid message-signature pairs (m, σ) under pk . The common inputs of both parties are $(param, pk, m, \sigma)$, while the prover has a secret verification key svk as an additional input. The verifier returns **invalid** if the prover can prove that (m, σ) is invalid under pk . Otherwise, the verifier outputs the symbol “ \perp ”.

The zero-knowlgedgeness of **Confirmation** and **Disavowal** ensures that the verifier is able to generate the communication transcript of **Confirmation** and **Disavowal** by itself. Hence, one cannot convince the third party the validity of (m, σ) by showing the transcript of **Confirmation** or **Disavowal**. This is the central requirement for undeniable signatures [KH06].

S-Convert $(param, svk, pk, m, \sigma) \rightarrow \pi\text{-}(m, \sigma, \text{bit})$.

The selectively-convert algorithm takes $param$, a secret verification key svk , its corresponding public key pk and a message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ as inputs, and returns a selective-proof $\pi\text{-}(m, \sigma, \text{bit})$, where $\text{bit} \in \{0, 1\}$. If $\text{bit} = 0$, π is a proof that (m, σ) is invalid under pk . Otherwise, $\text{bit} = 1$ and π is a proof that (m, σ) is valid under pk .

S-Verify $(param, pk, m, \sigma, \pi\text{-}(m, \sigma, \text{bit})) \rightarrow \{\text{valid}, \text{invalid}, \perp\}$.

The selectively-verify algorithm takes $param$, a public key pk , a message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ and a selective proof $\pi(m, \sigma, \text{bit})$ as inputs, and returns **valid**, **invalid** or a symbol “ \perp ”. Here, “ \perp ” indicates that π does not contain enough information to prove the validity (or, invalidity) of σ .

Correctness. Let (ssk, svk, pk) be the output of **KeyGen**, the correctness of a selectively and universally convertible undeniable signature scheme Σ includes:

1. Any undeniable signature produced by **Sign** is valid. That is,

$$\text{Verify}(param, svk, pk, m, \text{Sign}(param, ssk, pk, m)) = \text{valid}.$$

2. With the knowledge of svk corresponding to a public key pk , one is able to generate a selective-proof for any pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ such that

$$\begin{aligned} & \text{Verify}(param, svk, pk, m, \sigma) \\ = & \text{S-Verify}(param, pk, m, \sigma, \text{S-Convert}(param, svk, pk, m, \sigma)). \end{aligned}$$

2.2.1 Difference from Previous Definitions

We now make a further observation of our new definition above and compare it to prior definitions.

Firstly, in almost all definitions of undeniable signatures, a user’s secret key is denoted by only one symbol sk , which serves as both signing key and verification key. This reflects the essence of undeniable signatures without universal convertibility, as the publish of the verification key could enable anyone to produce valid undeniable signatures. However, in universally convertible undeniable signatures, it is certainly not necessary to use the signing key to verify undeniable signatures or execute **Confirmation/Disavowal** protocols. In our definition, the secret key is split into two different parts ssk and svk , which are used by **Sign** algorithm and **Verify** algorithm respectively. This, we believe, can show the most obvious difference between undeniable signatures with universal convertibility and those without such property. At the beginning, both ssk and svk are the secret information of the user, which makes the validity of this user’s undeniable signatures not publicly verifiable. However, at some later time, svk can be published to make the validity of undeniable signatures publicly verifiable by using the algorithm **Verify**. As we will see in next section, the publish of svk should not provide any advantage to the adversary in the game of unforgeability.

Secondly, splitting the secret key into ssk and svk also removes two algorithms **Universally-Convert** and **Universally-Verify** in previous definitions of universally convertible undeniable signatures [BCDP90, DP96, LV05b]. Algorithm **Universally-Convert** outputs a universal-proof (also known as universal-receipt), which enables anyone to check the validity of undeniable signatures by running algorithm **Universally-Verify**. As the universal-proof is determinate by user's public key rather than message-signature pairs, the production of such proof can be performed in algorithm **KeyGen** and thus we use svk instead. It follows that algorithm **Universally-Verify** is identical to algorithm **Verify** in our definition and does not need to be defined again.

Thirdly, if the verification key svk is not published, the validity of an undeniable signature is not publicly verifiable and the signature signer needs to use **Confirmation** (or, **Disavowal**) protocol to prove the validity (or, invalidity) to the verifier. Before doing that, it is fair to say that the prover needs to check the validity of undeniable signatures. In some definitions [KH06, KT06], σ is said to be a valid undeniable signature of m if there exists a random tape such that algorithm **Sign** will output σ . In this chapter, we use a different but traditional method to perform the verification. That is, as in most cases of digital signatures, we use algorithm **Verify** to decide if a given message-signature pair is valid or not. In this sense, algorithm **Verify** is similar to that in ordinary digital signatures. The difference is that **Verify** in undeniable signatures requires more information than the public key to perform the verification. We believe both methods are essentially the same but ours is more natural and can highlight the distinctive features of undeniable signatures.

2.2.2 (Strong) Unforgeability of Convertible Undeniable Signatures

The standard notion of security for a signature scheme is called existentially unforgeable under an adaptive chosen message attack [GMR88]. Recent works require undeniable signatures to satisfy a slightly stronger notion of security, namely strong unforgeability, which is defined using the following game between a challenger and an adversary:

Setup: The challenger runs algorithm **KeyGen**($param$) to obtain a secret signing key ssk , a secret verification key svk and a public key pk . The adversary is given $param$, svk and pk .

Signature Queries: Proceeding adaptively, the adversary requests signatures on at most q_S messages of its choice. For such a message m_i , the challenger responds with a signature $\sigma_i = \text{Sign}(param, ssk, pk, m_i)$. The adversary, with svk , does not need to make other queries.

Output: Eventually, the adversary outputs a pair (m^*, σ^*) and wins the game if

1. $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), (m_2, \sigma_2), \dots, (m_{q_S}, \sigma_{q_S})\}$; and
2. $\text{Verify}(param, svk, pk, m^*, \sigma^*) = \text{valid}$.

Let $\text{F-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 2.2 *A forger \mathcal{A} is said to (t, q_S, ε) -break the strong unforgeability of a selectively and universally convertible undeniable signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_S signature queries, and $\text{F-Adv}_{\mathcal{A}}$ is at least ε . Σ is (t, q_S, ε) -strongly existentially unforgeable under an adaptive chosen message attack if there exists no forger that (t, q_S, ε) -breaks it.*

In the random oracle model, we add the fourth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

To define the existential unforgeability of Σ under an adaptive chosen message attack [GMR88], we only need to make the following modification in Definition 4.2, namely, replacing $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), (m_2, \sigma_2), \dots, (m_{q_S}, \sigma_{q_S})\}$ with $m^* \notin \{m_1, m_2, \dots, m_{q_S}\}$ in the requirements of **Output**. Existential unforgeability and strongly existential unforgeability will be equivalent in Σ , if **Sign** is a deterministic algorithm. Recent research of undeniable signatures [LV05b] views strong unforgeability as necessary.

2.2.3 (Strong) Invisibility of Convertible Undeniable Signatures

The notion of invisibility in undeniable signatures was first introduced by Chaum *et al.* in [CvHP91]. Given a message-signature pair (m, σ) and a public key pk , the notion of invisibility is essentially the inability to determine whether σ is a valid undeniable signature of m under pk . This is defined by the following game between the challenger and the adversary:

Setup: The challenger runs algorithm $\text{KeyGen}(param)$ to obtain a secret signing key ssk , a secret verification key svk and a public key pk . The adversary is given $param$ and pk .

Queries I: Proceeding adaptively, the adversary is allowed to make queries as follows:

Signature Queries: Same as in Definition 4.2.

Verification Queries: For each verification query $(m_i, \sigma_i) \in \mathcal{M} \times \mathcal{S}$, the challenger responds by executing **Confirmation** protocol with the adversary if $\text{Verify}(param, svk, pk, m_i, \sigma_i) = \text{valid}$. Otherwise, **Disavowal** will be executed between the challenger and the adversary.

Selectively-Convert Queries: For each selectively-convert query $(m_i, \sigma_i) \in \mathcal{M} \times \mathcal{S}$, the challenger responds with the proof $\pi(m_i, \sigma_i, \text{bit}) = \text{S-Convert}(param, svk, pk, m_i, \sigma_i)$.

Output I: At the end of **Queries I**, the adversary outputs a message m^* , with the restriction that m^* is not one of **Signature Queries** during **Queries I** if **Sign** is a deterministic algorithm. The challenger responds by picking a random $b \in \{1, 0\}$ and computing $\sigma^* \in \mathcal{S}$ accordingly. If $b = 1$, $\sigma^* = \text{Sign}(param, ssk, pk, m^*)$. Otherwise, $b = 0$ and σ^* is an element chosen randomly from $\mathcal{S} \setminus \mathcal{S}_{m^*}$. Here, \mathcal{S}_{m^*} is the set of m^* 's valid signatures. In either case, σ^* is returned to the adversary as the response.

Queries II: After obtaining σ^* , the adversary can continue to make queries as in **Queries I** but with the restrictions that (m^*, σ^*) is not one of **Verification Queries** or **Selectively-Convert Queries**, and m^* is not one of **Signature Queries** if **Sign** is a deterministic algorithm.

Output II: Eventually, the adversary outputs b' and wins the game if $b' = b$.

Let $\text{D-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 2.3 A distinguisher \mathcal{A} is said to $(t, q_S, q_V, q_{SC}, \varepsilon)$ -break the strong invisibility of a selectively and universally convertible undeniable signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_S signature queries, q_V verification queries, q_{SC} selectively-convert queries, and $\text{D-Adv}_{\mathcal{A}}$ is at least $1/2 + \varepsilon$. Σ is

$(t, q_S, q_V, q_{SC}, \varepsilon)$ -strongly indistinguishable under an adaptive chosen message attack if there exists no distinguisher that $(t, q_S, q_V, q_{SC}, \varepsilon)$ -breaks it.

In the random oracle model, we add the sixth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

To define the invisibility of Σ under an adaptive chosen message attack, we only need to slightly modify Definition 2.3, by requiring that the challenge message m^* cannot be chosen as one of **Signature Queries** in **Output I** and **Queries II** no matter whether the **Sign** algorithm is deterministic or not. Invisibility and strong invisibility will be equivalent in Σ , if **Sign** is a deterministic algorithm.

2.2.4 (Strong) Anonymity of Convertible Undeniable Signatures

The property anonymity essentially requires that given a message-signature pair (m, σ) and two possible signers' public keys pk_0, pk_1 , it is computational impossible to decide whose secret key was used in the generation of σ . This property was introduced to the undeniable signature by Galbraith and Mao [GM03]. The authors suggested that anonymity is the most relevant security property for the undeniable signature. In convertible undeniable signatures, the property anonymity can be defined by the game as follows.

Setup: The challenger runs algorithm $\text{KeyGen}(param)$ to obtain (ssk_0, svk_0, pk_0) and (ssk_1, svk_1, pk_1) for two possible signers. The adversary is given $param$ as well as pk_0 and pk_1 .

Queries I: Proceeding adaptively, the adversary is allowed to make queries as follows.

Signature Queries: For each signature query (m, pk_i) , $i \in \{0, 1\}$, the challenger responds with a signature $\sigma = \text{Sign}(param, ssk_i, pk_i, m)$.

Verification Queries: For each verification query (m, σ, pk_i) , $i \in \{0, 1\}$, the challenger responds by executing **Confirmation** protocol with the adversary if $\text{Verify}(param, svk_i, pk_i, m, \sigma) = \text{valid}$. Otherwise, **Disavowal** will be executed between the challenger and the adversary.

Selectively-Convert Queries: For each selectively-convert query (m, σ, pk_i) , $i \in \{0, 1\}$, the challenger responds with the proof $\pi(m, \sigma, \text{bit}) = \text{S-Convert}(param, sk_i, pk_i, m, \sigma)$.

Output I: At the end of **Queries I**, the adversary outputs a message m^* , with the restriction that (m^*, pk_i) , $i \in \{0, 1\}$ is not one of **Signature Queries** during **Queries I** if **Sign** is a deterministic algorithm. The challenger responds by picking a random $b \in \{1, 0\}$ and calculating $\sigma^* = \text{Sign}(param, sk_b, pk_b, m^*)$. In either case, σ^* is returned to the adversary as the response.

Queries II: After obtaining σ^* , the adversary can continue to make queries as in **Queries I** with restrictions that (m^*, σ^*, pk_i) , $i \in \{0, 1\}$ is not one of **Verification Queries** or **Selectively-Convert Queries**, and (m^*, pk_i) , $i \in \{0, 1\}$ is not one of **Signature Queries** if **Sign** is a deterministic algorithm.

Output II: Eventually, the adversary outputs b' and wins the game if $b' = b$.

Let $\text{Anonymity-D-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 2.4 *A distinguisher \mathcal{A} $(t, q_S, q_V, q_{SC}, \varepsilon)$ -breaks the strong anonymity of a selectively and universally convertible undeniable signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_S signature queries, q_V verification queries, q_{SC} selectively-convert queries, and $\text{Anonymity-D-Adv}_{\mathcal{A}}$ is at least $1/2 + \varepsilon$. Σ is $(t, q_S, q_V, q_{SC}, \varepsilon)$ -strongly anonymous under an adaptive chosen message attack if there exists no distinguisher that $(t, q_S, q_V, q_{SC}, \varepsilon)$ -breaks it.*

In the random oracle model, we add the sixth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

To define the anonymity of Σ under an adaptive chosen-message attack, we only need to slightly modify Definition 2.4, requiring that the challenge message (m^*, pk_i) , $i \in \{0, 1\}$ cannot be chosen as one of **Signature Queries** in **Output I** and **Queries II** no matter whether the **Sign** algorithm is deterministic or not. Invisibility and strong invisibility will be equivalent in Σ , if **Sign** is a deterministic algorithm.

2.2.5 Relationship Between Anonymity and Invisibility in Convertible Undeniable Signatures

In [GM03], Galbraith and Mao showed that the property anonymity and invisibility are closely related in the undeniable signature. We now prove that anonymity and invisibility (or, strong anonymity and strong invisibility) are also closely related in convertible undeniable signatures.

Theorem 2.1 *If a selectively and universally convertible undeniable signature scheme Σ is (strongly) invisible in the sense of Definition 2.3, then Σ is also (strongly) anonymous in the sense of Definition 2.4.*

Proof. Suppose there exists an adversary \mathcal{D}_A who can $(t, q_S, q_V, q_{SC}, \varepsilon)$ -break the (strong) anonymity of Σ , then we show that there exists an adversary \mathcal{D}_I who can $(t, q_S, q_V, q_{SC}, \frac{\varepsilon}{2})$ -break the (strong) invisibility of Σ . Given $(param, pk)$, \mathcal{D}_I will act as \mathcal{D}_A 's challenger as follows:

Setup: \mathcal{D}_I sets $pk_1 = pk$ and runs algorithm $\text{KeyGen}(param)$ to obtain another user's key (ssk_0, svk_0, pk_0) . \mathcal{D}_I then flips a coin to obtain a bit b' . If $b' = 1$, then \mathcal{D}_A is given $param$ and (pk_0, pk_1) . Otherwise $b' = 0$, \mathcal{D}_A is given $param$ and (pk_1, pk_0) .

Queries I: \mathcal{D}_I is able to answer queries related to pk_0 by itself, as it has (ssk_0, svk_0) . For all queries related to pk_1 from \mathcal{D}_A , \mathcal{D}_I forwards them to its own challenger, and can thus answer such queries correctly.

Output I: For the challenge message m^* from \mathcal{D}_A , \mathcal{D}_I will send m^* to its own challenger. Let the response be σ^* , which will be returned to \mathcal{D}_A as the challenge signature.

Queries II: These queries can be correctly answered by \mathcal{D}_I as same as described in **Queries I**.

Output II: Eventually, \mathcal{D}_A output its guess b'' .

\mathcal{D}_I will output 1 if $b' = b''$ and 0 otherwise. The probability that \mathcal{D}_I outputs a correct guess is $\Pr[b' = b'' | b = 1] \Pr[b = 1] + \Pr[b' \neq b'' | b = 0] \Pr[b = 0] \geq (\frac{1}{2} + \varepsilon) \frac{1}{2} + \frac{1}{4} = \frac{1}{2} + \frac{\varepsilon}{2}$. This completes the proof of Theorem 2.1.

Theorem 2.2 *If a selectively and universally convertible undeniable signature scheme Σ is (strongly) anonymous in the sense of Definition 2.4, then Σ is also (strongly) invisible in the sense of Definition 2.3.*

Proof. Suppose there exists an adversary \mathcal{D}_I who can $(t, q_S, q_V, q_{SC}, \varepsilon)$ -break the (strong) invisibility of Σ , then there exists an adversary \mathcal{D}_A who can $(t, q_S, q_V, q_{SC}, \varepsilon - \epsilon)$ -break the (strong) anonymity of Σ . Here, ϵ is some value negligible. Given $(param, pk_0, pk_1)$, \mathcal{D}_A will act as \mathcal{D}_I 's challenger as follows:

Setup: \mathcal{D}_A sets $pk_1 = pk$ and \mathcal{D}_I is given $param$ and pk .

Queries I: For all queries related to pk from \mathcal{D}_I , \mathcal{D}_A forwards them to its own challenger, and can thus answer such queries correctly.

Output: Given \mathcal{D}_I 's challenge message m^* , \mathcal{D}_A will send m^* to its own challenger. Let the response be σ^* , which will be returned to \mathcal{D}_I as the response.

Queries II: These queries are answered by \mathcal{D}_A as same as described in **Queries I**.

Output II: Eventually, \mathcal{D}_I output its guess b' .

\mathcal{D}_A will output b' as its own guess. The probability that \mathcal{D}_A outputs a correct guess is $\Pr[b' = 1|b = 1] \Pr[b = 1] + \Pr[b' = 0|b = 0] \Pr[b = 0] = (\frac{1}{2} + \varepsilon)\frac{1}{2} + (\frac{1}{2} + \varepsilon - \epsilon)\frac{1}{2} \geq \frac{1}{2} + \varepsilon - \epsilon$. Here, ϵ is the probability that a signature produced by **Sign** using the secret key ssk_0 (corresponding to pk_0) is also valid under pk_1 . As in [GM03], ϵ is assumed to be negligible. This completes the proof Theorem 2.2.

2.2.6 Security of S-Convert

The security of algorithm S-Convert consists of two notions, namely, **Soundness of Selective-Proof** and **Unforgeability of Selective-Proof**. These two notions are defined as follows.

Soundness of Selective-Proof. This property requires the inability to generate a selective-proof which can prove a valid (invalid) message-signature pair as invalid (valid). It is formally defined in the game as below.

Given the system parameter $param$, the adversary outputs $svk, pk, (m, \sigma)$ and a selective-proof π . The adversary wins the game if

1. $\text{Verify}(param, sk, pk, m, \sigma) = \text{valid}$ and $\text{S-Verify}(param, pk, m, \sigma, \pi) = \text{invalid}$;
Or
2. $\text{Verify}(param, sk, pk, m, \sigma) = \text{invalid}$ and $\text{S-Verify}(param, pk, m, \sigma, \pi) = \text{valid}$.

Let $\text{S-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} .

Definition 2.5 *An adversary \mathcal{A} is said to (t, ε) -break the soundness of algorithm S-Convert in a selectively and universally convertible undeniable signature scheme Σ if \mathcal{A} runs in time at most t and $\text{S-Adv}_{\mathcal{A}}$ is at least ε . Algorithm S-Convert in Σ is (t, ε) -sound if there exists no adversary that (t, ε) -breaks it.*

Unforgeability of Selective-Proof. This property requires that selective-proofs can only be produced by the user with the knowledge of secret key. This is defined by the following game between the challenger and the adversary:

Setup: The challenger runs algorithm $\text{KeyGen}(param)$ to obtain a secret signing key ssk , a secret verification key sk and a public key pk . The adversary is given $param$ and pk .

Queries: Proceeding adaptively, the adversary is allowed to make all types of queries as described in Section 2.2.3 without any restrictions.

Output: Eventually, the adversary outputs a pair (m^*, σ^*) and a selective-proof π . The adversary wins the game if

1. $\text{S-Verify}(param, pk, m^*, \sigma^*, \pi) \neq \perp$; and
2. (m^*, σ^*) is not one of **Selectively-Convert** queries.

Let $\text{F-S-Convert-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 2.6 *An adversary \mathcal{A} is said to $(t, q_S, q_V, q_{SC}, \varepsilon)$ -break the unforgeability of algorithm S-Convert in a selectively and universally convertible undeniable signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_S signature queries, q_V verification queries, q_{SC} selectively-convert queries, and $\text{F-S-Convert-Adv}_{\mathcal{A}}$ is at least ε . Algorithm S-Convert is $(t, q_S, q_V, q_{SC}, \varepsilon)$ -existentially unforgeable if there exists no adversary that $(t, q_S, q_V, q_{SC}, \varepsilon)$ -breaks it.*

In the random oracle model, we add the sixth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

2.3 The Proposed Scheme

In this section, we will describe the proposed scheme and its security analysis.

2.3.1 The Description of Our Scheme

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair where $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime p . Let g_1 be the generator of \mathbb{G}_1 , g_2 be the generator of \mathbb{G}_2 and the bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We need three hash functions: $h_0, h_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and $h_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. The system-wide parameters are $param = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2, h_0, h_1, h_2)$.

KeyGen. Select two random integers $x, y \in \mathbb{Z}_p^*$, and compute $X = g_1^x, Y = g_2^y$. Also set the message space $\mathcal{M} = \{0, 1\}^*$ and the signature space $\mathcal{S} = \mathbb{G}_1$. The secret signing key $sk = (x, y)$, the secret verification key $vk = y$ and the public key $pk = (X, Y, \mathcal{M}, \mathcal{S})$.

Sign. Given the signer's secret signing key (x, y) and a message $m \in \mathcal{M}$, calculate the undeniable signature $\sigma = h_0(m)^{xy} \cdot h_1(m)^y$.

Verify. Given a secret verification key y , a public key $(X, Y, \mathcal{M}, \mathcal{S})$ and a pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$,

1. If $Y = g_2^y$ and $e(h_0(m), X^y) = e(\sigma \cdot h_1(m)^{-y}, g_2)$, output **valid**;
2. If $Y = g_2^y$ and $e(h_0(m), X^y) \neq e(\sigma \cdot h_1(m)^{-y}, g_2)$, output **invalid**;
3. Otherwise, output \perp .

Confirmation. Given a secret verification key y , a public key $(X, Y, \mathcal{M}, \mathcal{S})$ and a pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$, compute $W = e(\sigma, g_2) \cdot e(h_1(m), Y)^{-1}$. In order to prove that (m, σ) is valid, the prover must prove that the discrete logarithm $\text{DLog}_{g_2} Y$ is equal to the discrete logarithm $\text{DLog}_{e(h_0(m), X)} W$. In the proof, we use the designated verifier proofs [JSI96]. This can make the Confirmation protocol convince only one verifier, and avoid blackmailing [DY91, Jak94] and mafia attacks [DGB87] against undeniable signatures.

For a designated verifier with the public key $X_v = g_1^{x_v}$ and $Y_v = g_2^{y_v}$, the prover proves that “ $\text{DLog}_{g_2} Y = \text{DLog}_{e(h_0(m), X)} W$ ” or “It knows x_v ”. The prover sends the verifier (c_s, c_v, d_s, d_v) , which are produced as follows:

- **Transcript Generation:** Choose three random integers $c_v, d_v, r \in \mathbb{Z}_p$, and calculate:
 1. $A = g_2^r, B = e(h_0(m), X)^r, C = g_2^{d_v} X_v^{c_v}$;
 2. $h = h_2(m \parallel \sigma \parallel X_v \parallel Y_v \parallel A \parallel B \parallel C)$; and
 3. $c_s = h - c_v \pmod{p}$ and $d_s = r - c_s y \pmod{p}$.
- **Transcript Verification:** On receiving (c_s, c_v, d_s, d_v) , the verifier computes $A' = g_2^{d_s} Y^{c_s}$, $B' = e(h_0(m), X)^{d_s} W^{c_s}$ and $C' = g_2^{d_v} X_v^{c_v}$.
 1. If $c_s + c_v = h_2(m \parallel \sigma \parallel X_v \parallel Y_v \parallel A' \parallel B' \parallel C') \pmod{p}$, output **valid**;
 2. Otherwise, output \perp .

We use the technique in [CP92] to prove the equality of discrete logarithms, which is then converted into the non-interactive protocol by using Fiat-Shamir transformation. We will show later that **Confirmation** protocol is a zero knowledge proof with designated verifier in the random oracle model.

Disavowal. Given a secret verification key y , a public key $(X, Y, \mathcal{M}, \mathcal{S})$ and a pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$, calculate $W = e(\sigma, g_2) \cdot e(h_1(m), Y)^{-1}$. In order to prove that (m, σ) is invalid, the prover needs to prove that $\text{DLog}_{g_2} Y \neq \text{DLog}_{e(h_0(m), X)} W$. As in the **Confirmation** protocol, we use the designated verifier technique [JSI96] in the proof.

- **Transcript Generation:** For a designated verifier with the public key $X_v = g_2^{x_v}$ and $Y_v = g_2^{y_v}$, the prover sends the verifier $(A, c_s, c_v, d_s, \hat{d}_s, d_v)$, which are produced as follows:
 1. Choose five random integers $c_v, d_v, \alpha, \beta, r \in_R \mathbb{Z}_p$;
 2. Calculate $A = [e(h_0(m), X)^y / W]^r$, $B = e(h_0(m), X)^\alpha / W^\beta$, $C = g_2^\alpha / Y^\beta$ and $D = g_2^{d_v} X_v^{c_v}$;
 3. Set $h = h_2(m \parallel \sigma \parallel X_v \parallel Y_v \parallel A \parallel B \parallel C \parallel D)$ and calculate $c_s = h - c_v \pmod{p}$;
 4. Compute $d_s = \alpha + c_s y r \pmod{p}$ and $\hat{d}_s = \beta + c_s r \pmod{p}$.
- **Transcript Verification:** On receiving $(A, c_s, c_v, d_s, \hat{d}_s, d_v)$, the verifier calculates $B' = e(h_0(m), X)^{d_s} / (W^{\hat{d}_s} \cdot A^{c_s})$, $C' = g_2^{d_s} / Y^{\hat{d}_s}$ and $D' = g_2^{d_v} X_v^{c_v}$.

1. If $A \neq 1_{\mathbb{G}_T}$ and $c_s + c_v = h_2(m\|\sigma\|X_v\|Y_v\|A\|B'\|C'\|D') \pmod{p}$,
output **invalid**;
2. Otherwise, output \perp .

In **Disavowal**, the prover proves that “ $\text{DLog}_{g_2} Y \neq \text{DLog}_{e(h_0(m), X)} W$ ” or “it knows x_v ”. We use the technique in [CS03] to prove the inequality of discrete logarithms, which is then converted into the non-interactive protocol by using Fiat-Shamir transformation.

S-Convert. Given a public key $(X, Y, \mathcal{M}, \mathcal{S})$, its secret verification key y and a pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$, compute $W = e(\sigma, g_2) \cdot e(h_1(m), Y)^{-1}$.

- If (m, σ) is valid (one can check this by running **Verify** with the secret verification key y), one needs to prove that “ $\text{DLog}_{g_2} Y = \text{DLog}_{e(h_0(m), X)} W$ ”. The selective-proof $\pi(m, \sigma, 1) = (c, d) \in \mathbb{Z}_p^2$ is produced as following:
 1. Choose a random integer $r \in \mathbb{Z}_p$, and compute $A = g_2^r, B = e(h_0(m), X)^r$; and
 2. Set $c = h_2(m\|\sigma\|A\|B)$ and compute $d = r - cy \pmod{p}$.
- If (m, σ) is invalid, one needs to prove that “ $\text{DLog}_{g_2} Y \neq \text{DLog}_{e(h_0(m), X)} W$ ”. The proof $\pi(m, \sigma, 0) = (A, c, d, \hat{d}) \in \mathbb{G}_1 \times \mathbb{Z}_p^3$ is produced as the following:
 1. Choose three integers $\alpha, \beta, r \in_R \mathbb{Z}_p$;
 2. Compute $A = [e(h_0(m), X)^y / W]^r, B = e(h_0(m), X)^\alpha / W^\beta$ and $C = g_2^\alpha / Y^\beta$;
 3. Set $c = h_2(m\|\sigma\|A\|B\|C)$, and compute $d = \alpha + cyr \pmod{p}, \hat{d} = \beta + cr \pmod{p}$.

S-Verify. Given a public key $(X, Y, \mathcal{M}, \mathcal{S})$, a pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ and a proof π , calculate $W = e(\sigma, g_2) \cdot e(h_1(m), Y)^{-1}$.

- If $\pi = (c, d)$ and $c = h_2(m\|\sigma\|g_2^d Y^c \| e(h_0(m), X)^d W^c)$, output **valid**;
- If $\pi = (A, c, d, \hat{d}), A \neq 1_{\mathbb{G}_T}$ and $c = h_2(m\|\sigma\|A\|e(h_0(m), X)^d / (W^{\hat{d}} \cdot A^c) \| g_2^{\hat{d}} / Y^{\hat{d}})$, output **invalid**.
- Otherwise, output \perp .

The correctness of the our scheme is self-evident. We now prove that our scheme satisfies all the security requirements defined earlier.

2.3.2 Security Analysis: Confirmation and Disavowal

We will first prove that **Confirmation** satisfies the completeness, soundness and zero-knowledge in the random oracle model.

Completeness of Confirmation.

If (m, σ) is valid under the public key (X, Y) , then the verifier will output **valid** after receiving the transcript (c_s, c_v, d_s, d_v) . To see this, it suffices to show that $A' = A$, $B' = B$ and $C' = C$.

1. As $d_s = r - c_s y \pmod{p}$ and $A' = g_2^{d_s} Y^{c_s}$, we have $A' = g_2^{r - c_s y} g_2^{y c_s} = g_2^r = A$.
2. If (m, σ) is valid under the public key (X, Y) , then $W = e(\sigma, g_2) \cdot e(h_1(m), Y)^{-1} = e(h_0(m), X)^y$. As $B' = e(h_0(m), X)^{d_s} W^{c_s}$, we have $B' = e(h_0(m), X)^{r - c_s y} \cdot e(h_0(m), X)^{y c_s} = e(h_0(m), X)^r = B$.
3. C' is set as $g_2^{d_v} X_v^{c_v}$, which is exactly the same as C .

This completes the analysis of the completeness of **Confirmation**.

Soundness of Confirmation.

We now prove the soundness of **Confirmation** in the random oracle model. Suppose there is a prover can produce a transcript (c_s, d_s, c_v, d_v) which can make the verifier output **valid**. This means the following equality holds:

$$c_s + c_v = h_2(\xi) \pmod{p}, \xi = m \parallel \sigma \parallel X_v \parallel Y_v \parallel g_2^{d_s} Y^{c_s} \parallel e(h_0(m), X)^{d_s} W^{c_s} \parallel g_2^{d_v} X_v^{c_v}$$

In the random oracle model, if we assign another value (different from $c_s + c_v$) to $h_2(\xi)$, then this prover will generate another transcript (c'_s, d'_s, c'_v, d'_v) in polynomial time such that:

$$c'_s + c'_v = h_2(\xi) \pmod{p}, \xi = m \parallel \sigma \parallel X_v \parallel Y_v \parallel g_2^{d'_s} Y^{c'_s} \parallel e(h_0(m), X)^{d'_s} W^{c'_s} \parallel g_2^{d'_v} X_v^{c'_v}$$

If $c_s + c_v \neq c'_s + c'_v \pmod{p}$, then either $c_s \neq c'_s \pmod{p}$ or $c_v \neq c'_v \pmod{p}$.

1. If $c_s \neq c'_s \pmod{p}$, then σ must be a valid undeniable signature of m .

As the input to h_2 is identical, we have $g_2^{d_s} Y^{c_s} = g_2^{d'_s} Y^{c'_s}$ and $e(h_0(m), X)^{d_s} W^{c_s} = e(h_0(m), X)^{d'_s} W^{c'_s}$. The former equation leads us to extract $y = \text{Dlog}_{g_2} Y = (d_s - d'_s) \cdot (c'_s - c_s)^{-1} \pmod{p}$. In the latter equation, replacing $(d_s - d'_s) \cdot (c'_s - c_s)^{-1}$ with y , we have $W = e(h_0(m), X)^y$. As $W = e(\sigma, g_2) \cdot e(h_1(m), Y)^{-1}$, the signature σ must be equal to $h_0(m)^{xy} \cdot h_1(m)^y$. This completes our analysis in the case of $c_s \neq c'_s \pmod{p}$.

2. Otherwise, $c_v \neq c'_v \pmod{p}$ and we can extract $\text{DLog}_{g_2} X_v$.

Similar to the previous case, we have $g_2^{d_v} X_v^{c_v} = g_2^{d'_v} X_v^{c'_v}$. From this equation, we can extract $x_v = \text{Dlog}_{g_2} X_v = (d_v - d'_s) \cdot (c'_v - c_v)^{-1} \pmod{p}$. This completes our analysis in the case of $c_v \neq c'_v \pmod{p}$.

The above analysis shows that **Confirmation** satisfies the property of soundness in the random oracle model.

Zero-Knowledgeness of Confirmation. We show that the verifier with the knowledge x_v can generate the transcript of **Confirmation**, which has the same probability distribution (in the random oracle model) as that generated by the prover.

1. We first show that given a valid pair (m, σ) , the transcript (c_s, d_s, c_v, d_v) produced by the prover is a random element in \mathbb{Z}_p^4 (in the random oracle model).

In the **Confirmation** protocol, c_v and d_v are two random elements in \mathbb{Z}_p^2 . If h_2 is viewed as the random oracle, then its output h is independent of c_v, d_v, r , and will be a random element in \mathbb{Z}_p . This leads c_s and d_s to be two random elements in \mathbb{Z}_p^2 and independent of c_v, d_v . Therefore, (c_s, d_s, c_v, d_v) is a random element in \mathbb{Z}_p^4 .

2. With the knowledge x_v , one can also generate a transcript of **Confirmation**, such that it is a random element in \mathbb{Z}_p^4 (in the random oracle model).

Given a pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ (not necessarily valid under the public key $(X, Y, \mathcal{M}, \mathcal{S})$), choose three random integers $c_s, d_s, r \in \mathbb{Z}_p$ and compute:

- (a) $A = g_2^{d_s} Y^{c_s}$, $B = e(h_0(m), X)^{d_s} W^{c_s}$, and $C = g_2^r$; and
- (b) $h = h_2(m \parallel \sigma \parallel X_v \parallel Y_v \parallel A \parallel B \parallel C)$; and
- (c) $c_v = h - c_s \pmod{p}$ and $d_v = r - c_v x_v \pmod{p}$.

It is clear that the verifier will output **valid** given such (c_s, d_s, c_v, d_v) as the input. If the hash function h_2 is modeled as random oracle, c_v and d_v will be two random elements in \mathbb{Z}_p . Therefore, the transcript (c_s, d_s, c_v, d_v) is a random element in \mathbb{Z}_p^4 .

This completes the analysis that **Confirmation** is a zero-knowledge proof with designated verifier in the random oracle model. The analysis of **Disavowal** is quite similar to that of **Confirmation**, and is thus omitted.

2.3.3 Security Analysis: Unforgeability

The unforgeability of the proposed scheme can be reduced to the hardness of co-CDH problem defined in Section 1.5.

Theorem 2.3 *The proposed selectively and universally convertible undeniable signature scheme is $(t, q_S, q_H, \varepsilon)$ -(strongly) existentially unforgeable under an adaptive chosen message attack, if co-CDH problem is $(t + c_{\mathbb{G}_1}(q_H + 2q_S + 1), \varepsilon/\mathfrak{e}(q_S + 1))$ -hard on $(\mathbb{G}_1, \mathbb{G}_2)$. Here $c_{\mathbb{G}_1}$ is a constant that depends on \mathbb{G}_1 , and \mathfrak{e} is the base of the natural logarithm.*

Proof. Suppose there is a forger \mathcal{A} can $(t, q_S, q_H, \varepsilon)$ -break the proposed undeniable signature scheme. We show how to construct a $t + c_{\mathbb{G}_1}(q_H + 2q_S + 1)$ -time algorithm \mathcal{B} that solves co-CDH on $(\mathbb{G}_1, \mathbb{G}_2)$ with probability at least $\varepsilon/\mathfrak{e}(q_S + 1)$. This will contradict the fact that the co-CDH problem is $(t + c_{\mathbb{G}_1}(q_H + 2q_S + 1), \varepsilon/\mathfrak{e}(q_S + 1))$ -hard on $(\mathbb{G}_1, \mathbb{G}_2)$. Our proof employs the same technique as in [BLS01].

Let g_2 be a generator of \mathbb{G}_2 . Algorithm \mathcal{B} is given $g_2, g_2^a \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$. Let $u = g_2^a$, then \mathcal{B} 's goal is to obtain $h^a \in \mathbb{G}_1$. Algorithm \mathcal{B} simulates the challenger and interacts with forger \mathcal{A} as follows:

Setup: \mathcal{B} selects two random integers $r, y \in \mathbb{Z}_p^*$ and computes $X = u \cdot g_2^r, Y = g_2^y$.

The adversary is given (X, Y) and the secret verification key y .

h_0 **Queries:** At any time the forger \mathcal{A} can query the random oracle h_0 . To respond to these queries algorithm \mathcal{B} maintains a list of tuples (m_j, w_j, b_j, c_j) as explained below. We refer to this list as the h_0 -list. The list is initially empty. When \mathcal{A} makes a request m_i to the oracle h_0 , algorithm \mathcal{B} responds as follows

1. If m_i already appears on the h_0 -list in a tuple (m_i, w_i, b_i, c_i) , the algorithm \mathcal{B} responds with $h_0(m_i) = w_i \in \mathbb{G}_1$.
2. Otherwise, m_i is a fresh query and \mathcal{B} picks a random coin $c_i \in \{0, 1\}$ so that $\Pr[c_i = 0] = \frac{1}{q_S + 1}$.
3. Algorithm \mathcal{B} then chooses a random $b_i \in \mathbb{Z}_p$ and computes $w_i \leftarrow h^{1-c_i} \cdot \psi(g_2)^{b_i} \in \mathbb{G}_1$.
4. At last, \mathcal{B} adds (m_i, w_i, b_i, c_i) on the h_0 -list and responds to \mathcal{A} by setting $h_0(m_i) = w_i \in \mathbb{G}_1$.

h_1 Queries: At any time the forger \mathcal{A} can query the random oracle h_1 . To respond to these queries algorithm \mathcal{B} maintains a list of pairs (m_j, k_j) as explained below. We refer to this list as the h_1 -list. The list is initially empty. For a request m_i to the oracle h_1 , if m_i already appears on the h_1 -list in a pair (m_i, k_i) , the algorithm \mathcal{B} responds with $h_1(m_i) = k_i \in \mathbb{G}_1$. Otherwise, m_i is a fresh query and \mathcal{B} picks a random element $k_i \in \mathbb{G}_1$. The algorithm \mathcal{B} then adds (m_i, k_i) on the h_1 -list and responds \mathcal{A} by setting $h_1(m_i) = k_i \in \mathbb{G}_1$.

h_2 Queries: At any time the forger \mathcal{A} can query the random oracle h_2 . To respond to these queries algorithm \mathcal{B} maintains a list of pairs (ξ_j, η_j) as explained below. We refer to this list as the h_2 -list. The list is initially empty. For a request ξ_i to the oracle h_2 , if ξ_i already appears on the h_2 -list in a pair (ξ_i, η_i) , the algorithm \mathcal{B} responds with $h_2(\xi_i) = \eta_i \in \mathbb{Z}_p$. Otherwise, ξ_i is a fresh query and \mathcal{B} picks a random element $\eta_i \in \mathbb{Z}_p$. The algorithm \mathcal{B} then adds (ξ_i, η_i) on the h_2 -list and responds \mathcal{A} by setting $h_2(\xi_i) = \eta_i \in \mathbb{Z}_p$.

Signature Queries: For a signature query m_i chosen by the algorithm \mathcal{A} , Algorithm \mathcal{B} responds as follows:

1. Algorithm \mathcal{B} runs the algorithm for responding to h_0 -queries to obtain a $w_i \in \mathbb{G}_1$ such that $h_0(m_i) = w_i$. Algorithm \mathcal{B} then runs the algorithm for responding to h_1 -queries to obtain a $k_i \in \mathbb{G}_1$ such that $h_1(m_i) = k_i$. Let (m_i, w_i, b_i, c_i) be the corresponding tuple on the h_0 -list. If $c_i = 0$ then \mathcal{B} reports failure and terminates.
2. Otherwise, $c_i = 1$ and $w_i = \psi(g_2)^{b_i} \in \mathbb{G}_1$. In this case, algorithm \mathcal{B} is able to compute a valid undeniable signature $\sigma = \psi(u)^{yb_i} \cdot w_i^{yr} \cdot k_i^y$. Observe that $\sigma_i = w_i^{(a+r)y} \cdot k_i^y \in \mathbb{G}_1$ and therefore is a valid undeniable signature of m_i under the public key (X, Y) .

Output: Eventually algorithm \mathcal{A} produces a pair (m^*, σ^*) which is not the response as one of the signature queries. If there is no tuple on the h_0 -list (or, h_1 -list) containing m^* then \mathcal{B} issues a query itself for $h_0(m^*)$ (or $h_1(m^*)$) to ensure that such a tuple exists. We assume σ^* is a valid signature on m^* under the given public key. If it is not, \mathcal{B} reports failure and terminates. Otherwise, m^* is not one of the signature queries as **Sign** is a deterministic algorithm. Next, algorithm \mathcal{B} finds the tuple (m^*, w, b, c) on the h_0 -list and (m^*, k) in h_1 -list. If $c = 1$, then \mathcal{B} reports failure and terminates. Otherwise, $c = 0$ and therefore

$h_0(m^*) = w = h \cdot \psi(g_2)^b$. Hence, $\sigma^* = w^{y(a+r)} \cdot k^y = (h^a)^y \cdot \psi(u)^{yb} \cdot w^{yr} \cdot k^y$. Then, \mathcal{B} outputs the required $h^a \leftarrow (\sigma^*)^{y^{-1}} \cdot \psi(u)^{-b} \cdot w^{-r} \cdot k^{-1}$.

This completes the description of algorithm \mathcal{B} . We now compute the probability that \mathcal{B} does not fail. To do so, we analyze the three events needed for \mathcal{B} to succeed:

\mathcal{E}_1 : \mathcal{B} does not abort as a result of any \mathcal{A} 's signature queries.

\mathcal{E}_2 : \mathcal{A} outputs a valid pair (m^*, σ^*) .

\mathcal{E}_3 : Event \mathcal{E}_2 occurs and $c = 0$ for the tuple containing m^* on the h_0 -list.

Similar to the analysis in [BLS01], we have

$$\Pr[\mathcal{E}_1] \geq (1 - 1/(q_S + 1))^{q_S} \geq 1/\epsilon, \Pr[\mathcal{E}_2|\mathcal{E}_1] \geq \varepsilon, \text{ and } \Pr[\mathcal{E}_3|\mathcal{E}_1 \wedge \mathcal{E}_2] \geq 1/(q_S + 1).$$

Here, ϵ is the base of natural logarithm. Thus, the probability that \mathcal{B} can output h^a is at least $\varepsilon/\epsilon(q_S + 1)$.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to $(q_H + q_S)$ random oracle queries q_S signature queries, and compute h^a from σ^* . Each requires at most three multiplications and three exponentiations on \mathbb{G}_1 which we assume takes time $c_{\mathbb{G}_1}$. Hence, the total running time is at most $t + c_{\mathbb{G}_1}(q_H + 2q_S + 1)$. This completes the proof of Theorem 2.3.

2.3.4 Security Analysis: Invisibility

The invisibility of the proposed scheme can be reduced to the hardness of co-3-DDH problem defined in Section 1.5.

Theorem 2.4 *The proposed selectively and universally convertible undeniable signature scheme is $(t, q_S, q_V, q_{SC}, q_H, \varepsilon)$ -indistinguishable under an adaptive chosen message attack if co-3-DDH problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_S + 2q_V + 2q_{SC} + 1), \frac{2\varepsilon}{\epsilon^3(q_S + 1)^2})$ -hard on $(\mathbb{G}_1, \mathbb{G}_2)$. Here $c_{(\mathbb{G}_1, \mathbb{G}_2)}$ are two constants that depend on \mathbb{G}_1 and \mathbb{G}_2 , and ϵ is the base of the natural logarithm.*

Proof. Suppose there is a distinguisher \mathcal{A} can $(t, q_S, q_V, q_{SC}, q_H, \varepsilon)$ -break the invisibility of the proposed undeniable signature scheme. We show how to construct a $t + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_S + 2q_V + 2q_{SC} + 1)$ -time algorithm \mathcal{B} that solves co-3-DDH on $(\mathbb{G}_1, \mathbb{G}_2)$ with the advantage at least $\frac{2\varepsilon}{\epsilon^3(q_S + 1)^2}$.

Let g_2 be a generator of \mathbb{G}_2 . Algorithm \mathcal{B} is given $g_2, g_2^a, g_2^b \in \mathbb{G}_2$ and $h, h^c \in \mathbb{G}_1$. Its goal is to decide if $c = ab \pmod{p}$. Algorithm \mathcal{B} simulates the challenger and interacts with the distinguisher \mathcal{A} as follows.

Setup: \mathcal{B} selects two random integers $r_x, r_y \in \mathbb{Z}_p^*$ and computes $X = (g_2^a)^{r_x}$, $Y = (g_2^b)^{r_y}$. The adversary is given the public key $pk = (X, Y)$.

h_0 **Queries:** As in the proof of Theorem 2.3, \mathcal{B} will maintain an h_0 -list of tuples (m_j, w_j, b_j, c_j) as explained below. When \mathcal{A} makes a request m_i to the oracle h_0 , algorithm \mathcal{B} responds as follows

1. If m_i already appears on the h_0 -list in a tuple (m_i, w_i, b_i, c_i) , the algorithm \mathcal{B} responds with $h_0(m_i) = w_i \in \mathbb{G}_1$.
2. Otherwise, m_i is a fresh query and \mathcal{B} picks a random coin $c_i \in \{0, 1\}$ so that $\Pr[c_i = 0] = \frac{1}{q_S + 1}$.
3. If $c_i = 0$, \mathcal{B} then chooses a random $b_i \in \mathbb{Z}_p$ and computes $w_i \leftarrow h^{b_i} \in \mathbb{G}_1$.
4. Otherwise, $c_i = 1$ and \mathcal{B} needs to check if m_i already appears as one of h_1 queries. As shown later, \mathcal{B} will also maintain an h_1 -list consisting of tuples $(m_j, k_j, \zeta_j, \mu_j, \nu_j)$.
 - (a) If m_i already appears on the h_1 -list in a tuple $(m_i, k_i, \zeta_i, \mu_i, \nu_i)$, \mathcal{B} will set $b_i \leftarrow \nu_i$ and compute $w_i \leftarrow \psi(g_2)^{b_i}$.
 - (b) Otherwise, m_i has not appeared on the h_1 -list and \mathcal{B} will choose a random $b_i \in \mathbb{Z}_p$ and compute $w_i \leftarrow \psi(g_2)^{b_i}$.
5. At last, \mathcal{B} adds (m_i, w_i, b_i, c_i) on the h_0 -list and responds to \mathcal{A} by setting $h_0(m_i) = w_i \in \mathbb{G}_1$. We will show later that $h_0(m_i)$ is uniform in \mathbb{G}_1 and independent of \mathcal{A} 's current view.

h_1 **Queries:** At any time the distinguisher \mathcal{A} can query the random oracle h_1 . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(m_j, k_j, \zeta_j, \mu_j, \nu_j)$ as explained below. We refer to this list as the h_1 -list. The list is initially empty. When \mathcal{A} makes a request m_i to the oracle h_1 , algorithm \mathcal{B} responds as follows

1. If m_i already appears on the h_1 -list in a tuple $(m_i, k_i, \zeta_i, \mu_i, \nu_i)$, the algorithm \mathcal{B} responds with $h_1(m_i) = k_i \in \mathbb{G}_1$.

2. Otherwise, m_i is a fresh query and \mathcal{B} picks a random coin $\zeta_i \in \{0, 1\}$ so that $\Pr[\zeta_i = 0] = \frac{1}{q_S+1}$.
3. If $\zeta_i = 0$, algorithm \mathcal{B} then chooses two random integers $\mu_i, \nu_i \in \mathbb{Z}_p$ and computes $k_i \leftarrow \psi(g_2)^{\mu_i \nu_i} \in \mathbb{G}_1$.
4. Otherwise, $\zeta_i = 1$ and \mathcal{B} needs to check the h_0 -list.
 - (a) If m_i already appears on the h_0 -list in a tuple (m_i, w_i, b_i, c_i) , \mathcal{B} will choose a random integer $\mu_i \in \mathbb{Z}_p$, set $\nu_i \leftarrow b_i$ and compute $k_i \leftarrow \psi(g_2)^{\mu_i} / \psi(X)^{\nu_i}$.
 - (b) Otherwise, m_i has not appeared on the h_0 -list and \mathcal{B} will choose two random integers $\mu_i, \nu_i \in \mathbb{Z}_p$ and compute $k_i \leftarrow \psi(g_2)^{\mu_i} / \psi(X)^{\nu_i}$.
5. At last, \mathcal{B} adds $(m_i, k_i, \zeta_i, \mu_i, \nu_i)$ on the h_1 -list and responds to \mathcal{A} by setting $h_1(m_i) = k_i \in \mathbb{G}_1$.

We now show that both $h_0(m_i)$ and $h_1(m_i)$ are uniform in \mathbb{G}_1 and independent of \mathcal{A} 's current view. We first consider $h_0(m_i)$. It is clear that $h_0(m_i)$ is a random element in \mathbb{G}_1 if m_i does not appear on the h_1 -list when \mathcal{A} issues the h_0 query m_i . Otherwise, $h_0(m_i)$ will be computed as $\psi(g_2)^{\nu_i}$ and the tuple $(m_i, k_i, \zeta_i, \mu_i, \nu_i)$ already appears on h_1 -list. As ν_i is a random number in \mathbb{Z}_p , $h_0(m_i)$ will be uniform in \mathbb{G}_1 . For $h_1(m_i)$, its uniformness is guaranteed by the random integer $\mu_i \in \mathbb{Z}_p$, no matter whether $\zeta_i = 0$ or 1.

h_2 Queries: Same as in the proof of Theorem 2.3.

Signature Queries: For a signature query m_i chosen by the distinguisher \mathcal{A} , algorithm \mathcal{B} responds as follows:

1. Algorithm \mathcal{B} first runs the algorithm for responding to h_0 -queries to obtain a $w_i \in \mathbb{G}_1$ such that $h_0(m_i) = w_i$. Algorithm \mathcal{B} then runs the algorithm for responding to h_1 -queries to obtain a $k_i \in \mathbb{G}_1$ such that $h_1(m_i) = k_i$. Let (m_i, w_i, b_i, c_i) be the corresponding tuple on the h_0 -list, and $(m_i, k_i, \zeta_i, \mu_i, \nu_i)$ on the h_1 -list. If $c_i = 0$ or $\zeta_i = 0$ then \mathcal{B} reports failure and terminates.
2. Otherwise, $c_i = 1$ and $\zeta_i = 1$. In this case, we have $h_0(m_i) = \psi(g_2)^{b_i}$ and

$h_1(m_i) = \psi(g_2)^{\mu_i} / \psi(X)^{\nu_i}$. Thus, the undeniable signature of m_i is

$$\begin{aligned} \sigma &= h_0(m_i)^{(\mathbf{ar}_x \mathbf{br}_y)} \cdot h_1(m_i)^{\mathbf{br}_y} \\ &= \psi(g_2)^{b_i(\mathbf{ar}_x \mathbf{br}_y)} \cdot \psi(g_2)^{\mu_i(\mathbf{br}_y)} \cdot \psi(X)^{-\nu_i(\mathbf{br}_y)} \\ &= \psi(g_2)^{b_i(\mathbf{ar}_x \mathbf{br}_y)} \cdot \psi(g_2^{\mathbf{b}})^{\mu_i r_y} \cdot \psi(g_2)^{-\nu_i(\mathbf{ar}_x \mathbf{br}_y)}. \end{aligned}$$

Recall that $b_i = \nu_i$ if $c_i = 1$ and $\zeta_i = 1$, thus \mathcal{B} can respond \mathcal{A} with $\sigma = \psi(g_2^{\mathbf{b}})^{\mu_i r_y}$.

Verification Queries: As our Confirmation and Disavowal protocols are designated verifier proofs, each verification query must have the form as (m_i, σ_i, pk_v) . Here, $pk_v = (X_v, Y_v)$ is the verifier's public key chosen by the adversary \mathcal{A} . \mathcal{A} is required to prove its knowledge of the corresponding secret key.

1. If (m_i, σ_i) is returned by \mathcal{B} as the response to one of signature queries, \mathcal{B} will return a transcript of Confirmation protocol as the following.

\mathcal{B} randomly chooses four numbers $c_s, c_v, d_s, d_v \in \mathbb{Z}_p$, and computes $W = e(\sigma_i, g_2) \cdot e(h_1(m_i), Y)^{-1}$, $A = g_2^{d_s} Y^{c_s}$, $B = e(h_0(m_i), X)^{d_s} \cdot W^{c_s}$ and $C = g_2^{d_v} X_v^{c_v}$. Then \mathcal{B} sets $\xi_i = m_i \parallel \sigma_i \parallel X_v \parallel Y_v \parallel A \parallel B \parallel C$. If ξ_i or $c_s + c_v \pmod{p}$ already appears on the h_2 -list, \mathcal{B} will choose another four random numbers and recompute A, B, C such that ξ_i and $c_s + c_v \pmod{p}$ are both fresh on the h_2 -list. Then \mathcal{B} adds $(\xi_i, c_s + c_v \pmod{p})$ on the h_2 -list and returns (c_s, c_v, d_s, d_v) to \mathcal{A} .

2. Otherwise, (m_i, σ_i) is not returned by \mathcal{B} and \mathcal{B} will return a transcript of Disavowal protocol. This is justified as \mathcal{A} can produce a new valid message-signature pair with the probability at most $\frac{1}{q_V + q_{SC} + 1}$ according to Theorem 2.3 under the assumption that co-CDH problem is $(t + c_{\mathbb{G}_1}(q_H + 2q_S + 1), 1/\epsilon(q_S + 1)(q_V + q_{SC} + 1))$ -hard on $(\mathbb{G}_1, \mathbb{G}_2)$. Here $c_{\mathbb{G}_1}$ is a constant that depends on \mathbb{G}_1 , and ϵ is the base of the natural logarithm. \mathcal{B} can generate the transcript as the following.

\mathcal{B} chooses a random non-identity element $A \in \mathbb{G}_T$ and 5 random integers $c_s, c_v, d_s, \hat{d}_s, d_v \in \mathbb{Z}_p$, then it computes $W = e(\sigma_i, g_2) / e(h_1(m_i), Y)$, $B = e(h_0(m_i), X)^{d_s} / (W^{\hat{d}_s} \cdot A^{c_s})$, $C = g_2^{d_s} / Y^{\hat{d}_s}$ and $D = g_2^{d_v} X_v^{c_v}$. \mathcal{B} then sets $\xi_i = m_i \parallel \sigma_i \parallel X_v \parallel Y_v \parallel A \parallel B \parallel C \parallel D$. If ξ_i or $c_s + c_v \pmod{p}$ already appears on the h_2 -list, \mathcal{B} will regenerate A, B, C, D such that ξ_i and $c_s + c_v \pmod{p}$ are both fresh on the h_2 -list. Finally, \mathcal{B} adds $(\xi_i, c_s + c_v \pmod{p})$ on the h_2 -list and returns $(A, c_s, c_v, d_s, \hat{d}_s, d_v)$ to \mathcal{A} .

Selectively-Convert Queries. For each selectively-convert query (m_i, σ_i) , we assume that m_i already appears on the h_0 -list in the tuple (m_i, w_i, b_i, c_i) . If not, \mathcal{B} can make an h_0 query by itself such that the above tuple exists. \mathcal{B} then generates the selective proof using the similar proof as it responds verification queries.

Output I: At the end of **Queries I**, \mathcal{A} will output a message m^* . As the Sign algorithm in our scheme is deterministic, the message m^* can not be chosen as one of the signature queries. We assume that m^* already appears on the h_0 -list in the tuple (m^*, w, b, c) , and h_2 -list in the tuple $(m^*, k, \zeta, \mu, \nu)$. If not, \mathcal{B} can make an h_0 query and h_1 query by itself such that the above tuples exist.

1. If $c \neq 0$ or $\zeta \neq 0$, \mathcal{B} reports failure and aborts.
2. Otherwise, $c = 0$ and $\zeta = 0$. In this case, $h_0(m) = h^b$ and $h_1(m) = \psi(g_2)^{\mu\nu}$. \mathcal{B} computes $\sigma^* \leftarrow (h^c)^{br_x r_y} \cdot \psi(g_2^b)^{\mu\nu r_y}$.

Queries II: After receiving the challenging signature σ^* , \mathcal{A} can continue to issue queries with restrictions defined in Section 2.2.3. \mathcal{B} responds as same as described in the phase of **Queries I**.

Output II: Eventually, \mathcal{A} outputs its guess b' . \mathcal{B} sets b' as its own output of the co-3-DDH problem. If $\mathbf{c} = \mathbf{ab}$ in the instance of the co-3-DDH problem, then σ^* will be a valid undeniable signature under the public key (X, Y) . If \mathcal{A} can $(t, q_S, q_V, q_{SC}, \varepsilon)$ -break the invisibility of the proposed scheme, then $b' = 1$ happens with the probability at least $1/2 + \varepsilon$. Therefore, $\Pr[\mathcal{B}(g_2, g_2^a, g_2^b, h, h^c) = 1 : \mathbf{a}, \mathbf{b} \in_R \mathbb{Z}_p, \mathbf{c} = \mathbf{ab}] \geq 1/2 + \varepsilon$. Similarly, if $\mathbf{c} \neq \mathbf{ab}$, \mathcal{A} will output 1 with the probability at most $1/2 - \varepsilon$. Thus, we have $\Pr[\mathcal{B}(g_2, g_2^a, g_2^b, h, h^c) = 1 : \mathbf{a}, \mathbf{b} \in_R \mathbb{Z}_p, \mathbf{c} \in_R \mathbb{Z}_p \setminus \{\mathbf{ab}\}] \leq 1/2 - \varepsilon$. It follows that $\text{Adv co-3-DDH}_{\mathcal{B}} \geq 2\varepsilon$ if \mathcal{B} does not abort during the simulation.

It remains to compute the probability that \mathcal{B} does not abort during the simulation.

\mathcal{E}_1 : \mathcal{B} does not abort as a result of any \mathcal{A} 's signature queries. This happens with probability $(1 - \frac{1}{q_S+1})^{q_S} (1 - \frac{1}{q_S+1})^{q_S} \geq 1/\mathbf{e}^2$. Here, \mathbf{e} is the base of natural logarithm.

\mathcal{E}_2 : \mathcal{A} cannot generate a new valid message-signature pair. This happens with probability at least $(1 - \frac{1}{q_V + q_{SC} + 1})^{q_V + q_{SC}} \geq 1/\epsilon$ if co-CDH problem is $(t + c_{\mathbb{G}_1}(q_H + 2q_S + 1), \frac{1}{\epsilon(q_S + 1)(q_V + q_{SC} + 1)})$ -hard on $(\mathbb{G}_1, \mathbb{G}_2)$.

\mathcal{E}_3 : In the phase **Output I**, $c = 0$ and $\zeta = 0$ for the message m^* chosen by \mathcal{A} . This happens with the probability $\frac{1}{(q_S + 1)^2}$.

Therefore, the advantage that \mathcal{B} has in solving the given instance of the co-3-DDH problem is at least $\frac{2\epsilon}{\epsilon^3(q_S + 1)^2}$.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to $(q_H + q_S + q_V + q_{SC})$ random oracle queries, q_S signature queries, q_V verification queries, q_{SC} selectively-convert queries and compute the challenging signature σ^* . Among these queries, the most costly one is the response to each verification query which we assume takes time $c_{(\mathbb{G}_1, \mathbb{G}_2)}$. Hence, the total running time is at most $t + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_S + 2q_V + 2q_{SC} + 1)$. This completes the proof of Theorem 2.4.

2.3.5 Security Analysis: S-Convert

The algorithm S-Convert is almost the same as Confirmation and Disavowal protocols, with the only difference that there is no designated verifier in S-Convert. Thus, the analysis of the Soundness of S-Convert is almost the same as that in Confirmation and Disavowal protocols. We now prove that the unforgeability of S-Convert can be reduced to the Discrete Logarithm problem on \mathbb{G}_2 .

Theorem 2.5 *The algorithm S-Convert in the proposed selectively and universally convertible undeniable signature scheme is $(t, q_S, q_V, q_{SC}, \epsilon)$ -unforgeable under an adaptive chosen message attack if the Discrete Logarithm Problem is $(23q_H q_{SV} t / \epsilon + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_S + 2q_V + 2q_{SC} + 1), 1/9)$ -hard on \mathbb{G}_2 . Here, $c_{(\mathbb{G}_1, \mathbb{G}_2)}$ is a constant that depends on \mathbb{G}_1 and \mathbb{G}_2 ,*

Proof. Suppose there is a forger \mathcal{A} can $(t, q_S, q_V, q_{SC}, \epsilon)$ -break the unforgeability of S-Convert in the proposed undeniable signature scheme. We show how to construct a $23q_H q_{SV} t / \epsilon + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_S + 2q_V + 2q_{SC} + 1)$ -time algorithm \mathcal{B} that solves the Discrete Logarithm on \mathbb{G}_2 with success probability at least $1/9$.

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair with the pairing e and prime order p . Let g_1 be a generator of \mathbb{G}_1 and g_2 be a generator of \mathbb{G}_2 . Algorithm \mathcal{B} is given $g_2, u = g_2^a$.

Its goal is to output \mathbf{a} . Algorithm \mathcal{B} simulates the challenger and interacts with forger \mathcal{A} as follows.

Setup: \mathcal{B} selects two random integers $x, r \in \mathbb{Z}_p^*$ and computes $X = g_2^x$, $Y = u^r$.

The adversary is given (X, Y) and $(\mathbb{G}_1, \mathbb{G}_2, p, g_1, g_2, e)$.

h_0 Queries: As in proofs of previous theorems, \mathcal{B} will maintain an h_0 -list of tuples (m_j, w_j, b_j) as explained below. When \mathcal{A} makes a request m_i to the oracle h_0 , algorithm \mathcal{B} responds as follows.

1. If m_i already appears on the h_0 -list in a tuple (m_i, w_i, b_i) , the algorithm \mathcal{B} responds with $h_0(m_i) = w_i \in \mathbb{G}_1$.
2. Otherwise, m_i is a fresh query and \mathcal{B} picks a random element $b_i \in \mathbb{Z}_p$ and computes $w_i \leftarrow \psi(g_2)^{b_i} \in \mathbb{G}_1$.
3. At last, \mathcal{B} adds (m_i, w_i, b_i) on the h_0 -list and responds to \mathcal{A} by setting $h_0(m_i) = w_i \in \mathbb{G}_1$.

h_1 Queries: As in proofs of previous theorems, \mathcal{B} will maintain an h_1 -list of tuples (m_j, k_j, μ_j) as explained below. When \mathcal{A} makes a request m_i to the oracle h_1 , algorithm \mathcal{B} responds as follows.

1. If m_i already appears on the h_1 -list in a tuple (m_i, k_i, μ_i) , the algorithm \mathcal{B} responds with $h_1(m_i) = k_i \in \mathbb{G}_1$.
2. Otherwise, m_i is a fresh query and \mathcal{B} picks a random element $\mu_i \in \mathbb{Z}_p$ and computes $k_i \leftarrow \psi(g_2)^{\mu_i} \in \mathbb{G}_1$.
3. At last, \mathcal{B} adds (m_i, k_i, μ_i) on the h_1 -list and responds to \mathcal{A} by setting $h_1(m_i) = k_i \in \mathbb{G}_1$.

h_2 Queries: Same as in the proof of Theorem 2.3.

Signature Queries: For a signature query m_i chosen by the adversary \mathcal{A} , algorithm \mathcal{B} first runs the algorithm for responding to h_0 -queries to obtain a $w_i \in \mathbb{G}_1$ such that $h_0(m_i) = w_i$. Algorithm \mathcal{B} then runs the algorithm for responding to h_1 -queries to obtain a $k_i \in \mathbb{G}_1$ such that $h_1(m_i) = k_i$. Let (m_i, w_i, b_i) be the corresponding tuple on the h_0 -list, and (m_i, k_i, μ_i) on the h_1 -list. We have $h_0(m_i) = w_i = \psi(g_2)^{b_i}$ and $h_1(m_i) = k_i = \psi(g_2)^{\mu_i}$. Thus, the undeniable signature of m_i is $\sigma = h_0(m_i)^{x\mathbf{ar}} \cdot h_1(m_i)^{\mathbf{ar}} = \psi(g_2)^{b_i x \mathbf{ar}} \cdot \psi(g_2)^{\mu_i \mathbf{ar}} = \psi(u)^{b_i x r} \cdot \psi(u)^{\mu_i r}$.

Verification Queries: For any message signature pair (m_i, σ_i) , algorithm \mathcal{B} can verify the validity by checking the equation $e(\sigma_i, g_2) \cdot e(h_1(m_i), Y)^{-1} = e(h_0(m_i), Y^x)$ (Remind that x is chosen by \mathcal{B}). If the equality holds, (m_i, σ_i) is valid. Otherwise, it is invalid. \mathcal{B} then generates the transcript of the Confirmation (or, Disavowal) protocols using the same technique in the proof of Theorem 2.4.

Selectively-Convert Queries: For any message signature pair (m_i, σ_i) , algorithm \mathcal{B} first checks its validity as it responds verification queries. \mathcal{B} then generates the selective proof using the same technique in the proof of Theorem 2.4.

Output: Eventually, the adversary outputs a pair (m^*, σ^*) and a selective-proof π^* .

If \mathcal{A} can win the game with probability at least ε , then $\text{S-Verify}(param, pk, m^*, \sigma^*, \pi^*) \neq \perp$ and (m^*, σ^*) has never been chosen as one of the selectively-convert queries. We distinguish the following two case: (1) $\text{S-Verify}(param, pk, m, \sigma, \pi) = \text{valid}$; and (2) $\text{S-Verify}(param, pk, m, \sigma, \pi) = \text{invalid}$. We will show that in either case, \mathcal{B} can solve the given instance of Discrete Logarithm problem.

Suppose $\text{S-Verify}(param, pk, m^*, \sigma^*, \pi^*) = \text{valid}$, then we have $\pi^* = (c^*, d^*)$ and $c^* = h_2(\xi)$, where $\xi = m^* \parallel \sigma^* \parallel g_2^{d^*} Y^{c^*} \parallel e(h_0(m^*), X)^{d^*} W^{c^*}$ and $W = e(\sigma^*, g_2) \cdot e(h_1(m^*), Y)^{-1}$. Due to the forking lemma [PS00], if \mathcal{B} assigns another value c' (different from c^*) to $h_2(\xi)$, then \mathcal{A} will generate another proof (c', d') with probability greater than $1/9$ in time $23q_H q_{SC} t / \varepsilon$ such that $c' = h_2(\xi)$ and $\xi = m^* \parallel \sigma^* \parallel g_2^{d'} Y^{c'} \parallel e(h_0(m^*), X)^{d'} W^{c'}$. As the input to h_2 is identical, we have $g_2^{d^*} Y^{c^*} = g_2^{d'} Y^{c'}$. It follows that $Y = g_2^{(d^* - d')(c' - c^*)^{-1}}$. Recall that Y is set as $u^r = g_2^{ar}$, \mathcal{B} thus can compute $\mathbf{a} = (d^* - d')(c' - c^*)^{-1} r^{-1} \pmod{p}$. For the case $\text{S-Verify}(param, pk, m^*, \sigma^*, \pi^*) = \text{invalid}$, \mathcal{B} can compute \mathbf{a} using the same technique.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to $(q_H + q_S + q_V + q_{SC})$ random oracle queries, q_S signature queries, q_V verification queries, q_{SC} selectively-convert queries and compute discrete logarithm from (m^*, σ^*, π^*) . Among these queries, the most costly one is the response to each verification query which we assume takes time $c_{(\mathbb{G}_1, \mathbb{G}_2)}$. Hence, the total running time is at most $23q_H q_{SC} t / \varepsilon + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_S + 2q_V + 2q_{SC} + 1)$. This completes the proof of Theorem 2.5.

Table 2.2: Comparison with A Pairing-based Scheme [LV05b]

Scheme	Signature	Unforgeability	Invisibility
<i>The Scheme in [LV05b]</i>	272 bits	CDH	$(\ell, 1)$ -xyz-DCAA
<i>Our Proposed Scheme</i>	170 bits	CDH	co-3-DDH

2.3.6 Comparison with Other Schemes

We now compare our scheme with the known selectively and universally convertible undeniable signature schemes in [DP96, GRK00, LV05b]. For other schemes, some [LQ04, MV04a, KT06] are only selectively convertible, and others [Miy00, MPH96, MS97] are lacking of formal proofs. By comparison with the schemes in [DP96, GRK00, LV05b], our scheme requires the highest computation cost due to the bilinear mappings. Our scheme requires two Map-to-Point operations and two exponentiations on \mathbb{G}_1 in the signature generation, and two Map-to-Point operations, one exponentiation on \mathbb{G}_1 , two exponentiations on \mathbb{G}_2 and two pairing operations in the verification, which is even more time consuming than another pairing-based scheme in [LV05b]. However, our scheme has the shortest signature length (approximately 170 bits with a level of security similar to that of [LV05b]). In addition, the invisibility of our scheme is based on co-3-DDH assumption, which is a weaker assumption than the non-standard one introduced in [LV05b]. The comparison is summarized in Table 2.2.

2.4 Conclusion

In this chapter, we described a new construction of undeniable signatures with selective convertibility, universal convertibility and short signature length. The signature length of our scheme is as short as Boneh-Lynn-Shacham signature [BLS01]. The security of the proposed scheme is proved in the random oracle model. Its unforgeability can be reduced to the hardness of Computational Diffie-Hellman problem, and its invisibility can be reduced to the hardness of 3-Decisional Diffie-Hellman problem, which is a natural extension of the classic Decisional Diffie-Hellman problem on bilinear groups.

Chapter 3

Selectively and Universally Convertible Identity-based Undeniable Signatures

This chapter describes a new construction of identity-based undeniable signatures. It is the first selectively and universally convertible identity-based undeniable signature scheme where there is no certificate and a user's public key is his/her identity. The original scheme was presented at *Inscrypt 2007* [WMSH07].

3.1 Introduction

The central problem in a public key system is to prove that a public key is genuine and authentic, and has not been tampered with or replaced by a malicious third party. The usual approach to ensure the authenticity of a public key is to use a certificate. A (digital) certificate is a signature of a trusted certificate authority (CA) that binds together the identity of a user, his/her public key and other information. This kind of systems is referred as traditional public key infrastructure (traditional PKI). Although traditional PKI has been widely used, the certificate management is rather complicated and costly.

Shamir [Sha84] introduced the concept of identity-based public key cryptography (or, ID-PKC for short), whose original motivation is to ease the certificate management in the e-mail system. The essential idea is that the public key of a user can be directly derived from his/her identity, and therefore digital certificates are avoidable. The user obtains his/her secret key by interacting with some trusted master entity. In his paper, Shamir proposed an identity-based signature scheme. In contrast, the problem of designing an efficient and secure identity-based encryption scheme remained open until concrete constructions from the pairing over elliptic curve proposed in [BF01, SOK01].

A large number of identity-based signature schemes have been proposed so far,

but in general, these schemes can be classified into two categories. The schemes in the first category are constructed in the structure introduced by Boneh-Franklin [BF01], and use pairings as the building block. However, pairing is not the only choice to build identity-based signature schemes, as a traditional PKI-based signature scheme already implies an identity-based signature scheme by using the signature scheme twice. This idea was first formalized by Bellare, Neven, and Namprempe [BNN04] who proposed a generic and secure construction of identity-based signature schemes from any secure PKI-based signature schemes, and was further extended by Galindo, Herranz, and Kiltz [GHK06] who proposed a generic construction of identity-based signatures with additional properties. In particular, as shown in [GHK06], one can obtain an identity-based undeniable signature scheme from an undeniable signature scheme in traditional PKI. More precisely, the user first generates a public/private key pair as in traditional-PKI and then obtains a certificate proving the validity of the public key. The certificate is established by the third party by signing (using the master signing key) that user's public key together with the user's identity. In the actual identity-based signing process the user uses his/her private key to sign the message. The identity-based signature itself consists of three components: a signature in traditional PKI, a public key and the associated certificate generated by the third party. Such constructions, however, still have the same problem in the traditional PKI due to the use of certificates. This chapter, thus, will focus on the construction of selectively and universally convertible undeniable signature where there is no certificate and a user's public key is his/her identity.

Related Work. In [HYW03], Han, Yeung and Wang proposed the first identity-based undeniable signature scheme¹. However, Zhang, Safavi-Naini and Susilo showed that their scheme is insecure against the denial attack and the forge attack [ZSNS03]. Chow [Cho04] introduced a new scheme to fix the flaw in Han-Yeung-Wang's scheme but unfortunately, there is no formal proof provided, and therefore the security of the scheme is questionable. A new construction of identity-based undeniable signature was given by Libert and Quisquater in CT-RSA 2004 [LQ04]. It is also mentioned in [LQ04] that their scheme has the property of selective convertibility.

In this chapter, we will describe the first construction of selectively and universally convertible identity-based undeniable signatures.

¹The authors claimed that their scheme is a confirmer signature scheme, but it is actually an undeniable signature scheme which has been pointed out by Zhang, Safavi-Naini and Susilo [ZSNS03].

Organization of This Chapter. The rest of this chapter is organized as follows. In the next section, we will review the definitions of identity-based convertible undeniable signatures. The new construction of identity-based convertible undeniable signatures is described in Section 3.3, where the security analysis of the proposed scheme is also provided. Section 3.4 concludes this chapter.

3.2 Definitions of Identity-based Convertible Undeniable Signatures

This section will define the syntax and the security of identity-based convertible undeniable signatures.

Definition 3.1 *An identity-based convertible undeniable signature scheme is made up of eight algorithms, Setup, Key-Extract, Sign, Verify, Confirmation, Disavowal, S-Convert and S-Verify. For a fixed security parameter $param$, these algorithms work as follows:*

Setup($param$) $\rightarrow (msk, mpk)$.

This algorithm takes $param$ as input and returns the master secret key msk and master public key mpk . The master public key also includes the description of the message space \mathcal{M} and the signature space \mathcal{S} .

Key-Extract($param, msk, ID$) $\rightarrow (ssk_{ID}, svk_{ID})$.

This algorithm takes $param$, the master secret key msk and an entity's identity ID as inputs, and returns the secret signing key ssk_{ID} and the secret verification key svk_{ID} . The user can publish the secret verification key svk_{ID} to make all his/her undeniable signatures publicly verifiable with the algorithm **Verify** defined below.

Sign($param, mpk, ID, ssk_{ID}, m$) $\rightarrow \sigma$.

This algorithm takes $param$, the master public key mpk , a user's identity ID , a secret signing key ssk_{ID} and a message $m \in \mathcal{M}$ as inputs, and returns an element $\sigma \in \mathcal{S}$.

Verify($param, mpk, ID, svk, m, \sigma$) $\rightarrow \{\text{valid}, \text{invalid}, \perp\}$.

This algorithm takes $param$, the master public key mpk , a user's identity ID , a secret verification key svk and a message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ as

inputs, and returns **valid**, **invalid** or a symbol “ \perp ”. Here, “ \perp ” indicates that svk does not contain enough information to prove the validity (or, invalidity) of σ . This could happen, for example, when svk is not the secret verification key of the user with identity ID . A message-signature pair (m, σ) is said to be valid under ID if $\text{Verify}(param, mpk, ID, svk, m, \sigma) = \text{valid}$. Similarly, (m, σ) is said to be invalid under ID if $\text{Verify}(param, mpk, ID, svk, m, \sigma) = \text{invalid}$.

$\text{Confirmation}(param, mpk, ID, m, \sigma, svk_{ID}) \rightarrow \{\text{valid}, \perp\}$.

This is a zero-knowledge proof system between a prover and a verifier on a valid message-signature pair (m, σ) of the user with identity ID . The common inputs of both parties are $(param, mpk, ID, m, \sigma)$, and the prover has ID 's secret verification key svk_{ID} as an additional input. The verifier returns **valid** if the prover can prove (m, σ) is valid under ID . Otherwise, the verifier outputs “ \perp ”.

$\text{Disavowal}(param, mpk, ID, m, \sigma, svk_{ID}) \rightarrow \{\text{invalid}, \perp\}$.

This is a zero-knowledge proof system between a prover and a verifier on an invalid message-signature pair (m, σ) under ID . The common inputs of both parties are $(param, mpk, ID, m, \sigma)$, while the prover has a secret verification key svk_{ID} as an additional input. The verifier returns **invalid** if the prover can prove that (m, σ) is invalid under ID . Otherwise, the verifier outputs the symbol “ \perp ”.

$\text{S-Convert}(param, mpk, ID, svk_{ID}, m, \sigma) \rightarrow \pi(m, \sigma, \text{bit})$.

The selectively-convert algorithm takes $param$, the master public key mpk , a user's identity ID , ID 's secret verification key svk_{ID} and a message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ as inputs, and returns a selective-proof $\pi(m, \sigma, \text{bit})$, where $\text{bit} \in \{0, 1\}$. If $\text{bit} = 0$, π is a proof that (m, σ) is invalid under ID . Otherwise, $\text{bit} = 1$ and π is a proof that (m, σ) is valid under ID .

$\text{S-Verify}(param, mpk, ID, m, \sigma, \pi(m, \sigma, \text{bit})) \rightarrow \{\text{valid}, \text{invalid}, \perp\}$.

The selectively-verify algorithm takes $param$, the master public key mpk , a user's identity ID , a message-signature pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ and a selective proof $\pi(m, \sigma, \text{bit})$ as inputs, and returns **valid**, **invalid** or a symbol “ \perp ”. Here, “ \perp ” indicates that π does not contain enough information to prove the validity (or, invalidity) of σ .

Correctness. Let (ssk_{ID}, svk_{ID}) be the output of **Key-Extract** algorithm for the input ID , the correctness of a selectively and universally convertible identity-based undeniable signature scheme Σ includes:

1. Any undeniable signature produced by **Sign** is valid. That is,

$$\text{Verify}(param, mpk, ID, svk_{ID}, m, \text{Sign}(param, mpk, ID, ssk_{ID}, m)) = \text{valid}.$$

2. With the knowledge of svk_{ID} , one is able to generate a selective-proof for any pair $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$ such that

$$\begin{aligned} & \text{Verify}(param, mpk, ID, svk_{ID}, m, \sigma) \\ &= \text{S-Verify}(param, mpk, ID, m, \sigma, \text{S-Convert}(param, mpk, ID, svk_{ID}, m, \sigma)). \end{aligned}$$

3.2.1 Unforgeability of Identity-based Convertible Undeniable Signatures

In an identity-based convertible undeniable signature scheme, the existential unforgeability under an adaptive chosen message and chosen identity attack is defined using the following game between a challenger and an adversary:

Setup: The challenger first generates the system parameter $param$ and the master secret/public key pair (msk, mpk) . The adversary is given $param$ and mpk .

User-Create Queries: Proceeding adaptively, the adversary can ask the challenger to create at most q_{UC} identities $\{ID_1, \dots, ID_{q_{UC}}\}$. For such a request ID_i , the challenger first runs algorithm **Key-Extract** $(param, msk, ID_i)$ and obtains the key pair (ssk_i, svk_i) . After that, the challenger adds (ID_i, ssk_i, svk_i) on the *user-list* which is initially empty. The adversary is given svk_i . We assume that the adversary must first create user ID before any queries related to ID .

Signing-Key-Extract Queries: Proceeding adaptively, the adversary requests secret signing keys of at most q_E identities $\{ID_1, \dots, ID_{q_E}\}$ which have been created. For such an identity ID_i , the challenger responds with ssk_i which appears in the tuple (ID_i, ssk_i, svk_i) on the *user-list*.

Signature Queries: Proceeding adaptively, the adversary can make at most q_S signatures requests with the form (m_i, ID_i) . For such a query, the challenger responds with a signature $\sigma_i = \text{Sign}(param, mpk, ID_i, ssk_i, m_i)$.

Output: Eventually, the adversary outputs (m^*, σ^*, ID^*) and wins the game if

1. ID^* is not one of **Signing-Key-Extract Queries**;
2. (m^*, ID^*) is not one of **Signature Queries**; and
3. $\text{Verify}(param, mpk, ID^*, sk^*, m^*, \sigma^*) = \text{valid}$.

Let $\text{F-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 3.2 *A forger \mathcal{A} is said to $(t, q_{UC}, q_E, q_S, \varepsilon)$ -break the unforgeability of a selectively and universally convertible identity-based undeniable signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_{UC} user-create queries, q_E signing-key-extract queries, q_S signature queries, and $\text{F-Adv}_{\mathcal{A}}$ is at least ε . Σ is $(t, q_{UC}, q_E, q_S, \varepsilon)$ -existentially unforgeable under an adaptive chosen message and chosen identity attack if there exists no forger that $(t, q_{UC}, q_E, q_S, \varepsilon)$ -breaks it.*

In the random oracle model, we add an additional parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

3.2.2 Invisibility of Identity-based Convertible Undeniable Signatures

The invisibility of identity-based convertible undeniable signatures is defined by the following game between the challenger and the adversary:

Setup: The challenger first generates the system parameter $param$ and the master secret/public key pair (msk, mpk) . The adversary is given $param$ and mpk .

Queries I: Proceeding adaptively, the adversary is allowed to make queries as follows:

User-Create Queries: Proceeding adaptively, the adversary can ask the challenger to create at most q_{UC} identities $\{ID_1, \dots, ID_{q_{UC}}\}$. For such a request ID_i , the challenger first runs algorithm $\text{Key-Extract}(param, msk, ID_i)$ and obtain the key pair (ssk_i, sk_i) . After that, the challenger adds (ID_i, ssk_i, sk_i) on the *user-list* which is initially empty. We assume that the adversary must first create user ID before any queries related to ID .

Key-Extract Queries: Proceeding adaptively, the adversary requests secret signing keys and secret verification keys of at most q_E identities $\{ID_1, \dots, ID_{q_E}\}$ which have been created. For such an identity ID_i , the challenger responds with (ssk_i, svk_i) which appears in the tuple (ID_i, ssk_i, svk_i) on the *user-list*.

Signature Queries: Proceeding adaptively, the adversary can make at most q_S signature requests with the form (m_i, ID_i) . For such a query, the challenger responds with a signature $\sigma_i = \text{Sign}(param, mpk, ID_i, ssk_i, m_i)$.

Verification Queries: Proceeding adaptively, the adversary can make at most q_V verification requests with the form (m_i, σ_i, ID_i) . The challenger responds by executing **Confirmation** protocol with the adversary if $\text{Verify}(param, mpk, ID_i, svk_i, m_i, \sigma_i) = \text{valid}$. Otherwise, **Disavowal** will be executed between the challenger and the adversary.

Selectively-Convert Queries: Proceeding adaptively, the adversary can make at most q_{SC} selectively-convert requests with the form (m_i, σ_i, ID_i) . The challenger responds with the proof $\pi(m_i, \sigma_i, \text{bit}) = \text{S-Convert}(param, mpk, ID_i, svk_i, m_i, \sigma_i)$.

Output I: At the end of **Queries I**, the adversary outputs a message m^* and a created user ID^* , with restrictions that ID^* is not one of **Key-Extract** queries and (m^*, ID^*) is not one of **Signature Queries** during **Queries I**. The challenger responds by picking a random $b \in \{1, 0\}$ and computing $\sigma^* \in \mathcal{S}$ accordingly. If $b = 1$, $\sigma^* = \text{Sign}(param, mpk, ID^*, ssk^*, m^*)$. Otherwise, $b = 0$ and σ^* is an element chosen randomly from $\mathcal{S} \setminus \mathcal{S}_{m^*}$. Here, \mathcal{S}_{m^*} is the set of m^* 's valid signatures. In either case, σ^* is returned to the adversary as the response.

Queries II: After obtaining σ^* , the adversary can continue to make queries as in **Queries I** but with restrictions that ID^* is not one of **Key-Extract** queries, (m^*, σ^*, ID^*) is not one of **Verification Queries** or **Selectively-Convert Queries** and (m^*, ID^*) is not one of **Signature Queries**.

Output II: Eventually, the adversary outputs b' and wins the game if $b' = b$.

Let $\text{D-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 3.3 A distinguisher \mathcal{A} is said to $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, \varepsilon)$ -break the invisibility of a selectively and universally convertible identity-based undeniable signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_{UC} user-create queries, q_E key-extract queries, q_S signature queries, q_V verification queries, q_{SC} selectively-convert queries, and $\text{D-Adv}_{\mathcal{A}}$ is at least $1/2 + \varepsilon$. Σ is $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, \varepsilon)$ -indistinguishable under an adaptive chosen message and chosen identity attack if there exists no distinguisher that $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, \varepsilon)$ -breaks it.

When discussing security in the random oracle model, we add an additional parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

3.2.3 Security of S-Convert

The security of algorithm S-Convert consists of two notions, namely, **Soundness of Selective-Proof** and **Unforgeability of Selective-Proof**. These two notions are defined as follows.

Soundness of Selective-Proof. This property requires the inability to generate a selective-proof which can prove a valid (invalid) message-signature pair as invalid (valid). It is formally defined in the game as below.

Given the system parameter $param$, the master public key mpk , the user ID 's secret signing key ssk_{ID} and a user's verification key svk_{ID} , the adversary outputs (m, σ) and a selective-proof π . The adversary wins the game if

1. $\text{Verify}(param, mpk, ID, svk_{ID}, m, \sigma) = \text{valid}$ and $\text{S-Verify}(param, mpk, ID, m, \sigma, \pi) = \text{invalid}$; Or
2. $\text{Verify}(param, mpk, ID, svk_{ID}, m, \sigma) = \text{invalid}$ and $\text{S-Verify}(param, mpk, ID, m, \sigma, \pi) = \text{valid}$.

Let $\text{S-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} .

Definition 3.4 An adversary \mathcal{A} is said to (t, ε) -break the soundness of algorithm S-Convert in a selectively and universally convertible identity-based undeniable signature scheme Σ if \mathcal{A} runs in time at most t and $\text{S-Adv}_{\mathcal{A}}$ is at least ε . Algorithm S-Convert in Σ is (t, ε) -sound if there exists no adversary that (t, ε) -breaks it.

Unforgeability of Selective-Proof. This property requires that selective-proofs can only be produced by the user with the knowledge of secret key. This is defined by the following game between the challenger and the adversary:

Setup: The challenger first generates the system parameter $param$ and the master secret/public key pair (msk, mpk) . The adversary is given $param$ and mpk .

Queries: Proceeding adaptively, the adversary is allowed to make all types of queries as described in Section 3.2.2.

Output: Eventually, the adversary outputs (m^*, σ^*, ID^*) and a selective-proof π . The adversary wins the game if

1. ID^* is not one of **Key-Extract Queries**;
2. $S\text{-Verify}(param, mpk, ID^*, m^*, \sigma^*, \pi) \neq \perp$; and
3. (m^*, σ^*, ID^*) is not one of **Selectively-Convert** queries.

Let $F\text{-S-Convert-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 3.5 *An adversary \mathcal{A} is said to $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, \varepsilon)$ -break the unforgeability of algorithm **S-Convert** in a selectively and universally convertible identity-based undeniable signature scheme Σ if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_{UC} user-create queries, q_E key-extract queries, q_S signature queries, q_V verification queries, q_{SC} selectively-convert queries, and $F\text{-S-Convert-Adv}_{\mathcal{A}}$ is at least ε . Algorithm **S-Convert** is $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, \varepsilon)$ -existentially unforgeable if there exists no adversary that $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, \varepsilon)$ -breaks it.*

When discussing security in the random oracle model, we add an additional parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

3.3 The Proposed Scheme

In this section, we will describe the proposed scheme and its security analysis.

3.3.1 The Description of Our Scheme

Let e be the pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ described in Section 1.5. The order of \mathbb{G}_1 is a prime p where $p \geq 2^k$, k is the system's security number. The generator of \mathbb{G}_1 is g_1 . There are four cryptographic hash functions in our scheme: $h_1, h_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$, $h_3 : \mathbb{G}_T \times \mathbb{G}_1 \rightarrow \mathbb{G}_1$ and $h_4 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. The system parameter $param = \{\mathbb{G}_1, \mathbb{G}_T, e, p, k, g_1, h_1, h_2, h_3, h_4\}$.

Setup: The PKG in the identity-based system picks a random element $msk \in \mathbb{Z}_p^*$ and calculates the master public key $mpk = g_1^{msk}$.

Key-Extract: For a user with the identity ID , the PKG calculates $sk_{ID} = h_1(ID)^{msk}$, $svk_{ID} = h_1(ID \parallel \text{Undeniable})^{msk}$. The string " $ID \parallel \text{Undeniable}$ " indicates that svk_{ID} is the secret verification key of ID in undeniable signatures. The secret signing key $ssk_{ID} = (sk_{ID}, svk_{ID})$.

Sign: The signer ID generates a convertible undeniable signature for the message m as follows:

- Calculate $U = e(svk_{ID}, h_2(m))$; and
- Choose a random number $v \in \mathbb{Z}_p$ and set $V = g_1^v$, $W = sk_{ID} \cdot h_3(U, V)^v$.

The undeniable signature of the message m is $\sigma = (U, V, W)$. The signature space of user ID is $\mathcal{S}_{ID} = \{(U, V, W) \in \mathbb{G}_T \times \mathbb{G}_1 \times \mathbb{G}_1 : e(W, g_1) = e(h_1(ID), mpk)e(h_3(U, V), V)\}$.

Verify: Given a message-signature pair $(m, \sigma = (U, V, W))$, an identity ID and a secret verification key svk ,

- If $e(svk, g_1) = e(h_1(ID \parallel \text{Undeniable}), mpk)$, output **valid** if $U = e(svk, h_2(m))$ and $e(W, g_1) = e(h_1(ID), mpk)e(h_3(U, V), V)$, and **invalid** otherwise.
- Otherwise, output " \perp " as svk is not the secret verification key of ID .

Confirmation. Given a user's ID and the associated secret verification key svk_{ID} , one can prove that a pair (m, σ) is valid under ID . In the proof, we use the designated verifier proofs [JSI96]. This can make the **Confirmation** protocol convince only one verifier, and avoid blackmailing [DY91, Jak94] and mafia

attacks [DGB87] against undeniable signatures. Given the verifier's identity ID_V , the transcript is generated as follows:

- **Transcript Generation:** Choose $r, c_v \in_R \mathbb{Z}_p$, $C_v \in_R \mathbb{G}_1$, then calculate
 1. $R_1 = e(g_1, g_1)^r$, $R_2 = e(h_2(m), g_1)^r$ and $R_3 = e(C_v, g_1)e(h_1(ID_v), mpk)^{c_v}$;
 2. $c = h_4(ID \| ID_v \| R_1 \| R_2 \| R_3 \| m \| \sigma)$, $c_s = c - c_v \pmod{p}$, $C_s = g_1^r \cdot (svk_{ID})^{-c_s}$. (c_s, c_v, C_s, C_v) is sent to the verifier V as the transcript of **Confirmation**.
- **Transcript Verification:** For the transcript (c_s, c_v, C_s, C_v) and the message-signature pair $(m, \sigma = (U, V, W))$ under ID :
 1. If $e(W, g_1) \neq e(h_1(ID), mpk)e(h_3(U, V), V)$, output **invalid** as σ is not in the valid signature space;
 2. Otherwise, $e(W, g_1) = e(h_1(ID), mpk)e(h_3(U, V), V)$,
 - Calculate $R'_1 = e(C_s, g_1)e(h_1(ID \| Undeniable), mpk)^{c_s}$, $R'_2 = e(C_s, h_2(m))U^{c_s}$, $R'_3 = e(C_v, g_1) \cdot e(h_1(ID_v), mpk)^{c_v}$, and
 - Output **valid** if $c_s + c_v = h_4(ID \| ID_v \| R'_1 \| R'_2 \| R'_3 \| m \| \sigma)$, and “ \perp ” otherwise.

Disavowal. Given a user's ID and the associated secret verification key svk_{ID} , one can prove that a pair (m, σ) is invalid under ID . In the proof, we use the designated verifier proofs [JSI96]. Given the verifier's identity ID_V , the transcript is generated as follows:

- **Transcript Generation:** Choose $z, d_v, \alpha, \beta \in_R \mathbb{Z}_p$, $D_v \in_R \mathbb{G}_1$ and calculate:
 1. $A = [\frac{e(svk_{ID}, h_2(m))}{U}]^z$, $B = \frac{[e(h_2(m), g_1)]^\alpha}{U^\beta}$, $C = \frac{e(g_1, g_1)^\alpha}{e(h_1(ID \| Undeniable), mpk)^\beta}$ and $D = e(D_v, g_1)e(h_1(ID_v), mpk)^{d_v}$;
 2. $d = h_4(ID \| ID_v \| A \| B \| C \| D \| m \| \sigma)$, $d_s = d - d_v \pmod{p}$; and
 3. $D_s = g_1^\alpha \cdot (svk_{ID})^{d_s z}$, $\hat{d}_s = \beta + d_s z \pmod{p}$. $(A, d_s, d_v, \hat{d}_s, D_s, D_v)$ is sent to the verifier ID_V as a transcript of **Disavowal**.
- **Transcript Verification:** For the transcript $(A, d_s, d_v, \hat{d}_s, D_s, D_v)$ and $(m, \sigma = (U, V, W), ID)$,

1. Output **invalid** if $e(W, g_1) \neq e(h_1(ID), mpk)e(h_3(U, V), V)$;
2. Otherwise, $e(W, g_1) = e(h_1(ID), mpk)e(h_3(U, V), V)$,
 - Calculate $B' = \frac{e(D_s, h_2(m))}{U^{d_s} \cdot A^{d_s}}$, $C' = \frac{e(D_s, g_1)}{e(h_1(ID \| \text{Undeniable}), mpk)^{\widehat{d}_s}}$, $D' = e(D_v, g_1)e(h_1(ID_v), mpk)^{d_v}$.
 - Output **invalid** if $A \neq 1$ and $d_s + d_v = h_4(ID \| ID_v \| A \| B' \| C' \| D' \| m \| \sigma)$, and “ \perp ” otherwise.

S-Convert. For a user's ID , the associated secret verification key svk_{ID} and a message-signature pair $(m, \sigma = (U, V, W))$:

- If $U = e(svk_{ID}, h_2(m))$ and $e(W, g_1) = e(h_1(ID), mpk)e(h_3(U, V), V)$, (m, σ) is a valid message-signature pair under ID . The selective proof $\pi(m, \sigma, 1) = (c, C)$ is generated as following.
 - Choose a random number $r \in \mathbb{Z}_p$ and calculate $R_1 = e(g_1, g_1)^r$, $R_2 = e(h_2(m), g_1)^r$;
 - Set $c = h_4(ID \| R_1 \| R_2 \| m \| \sigma)$ and calculate $C = g_1^r \cdot (svk_{ID})^{-c}$.
- Otherwise, (m, σ) is an invalid message-signature pair under ID . The selective proof $\pi(m, \sigma, 0) = (A, d, D, \widehat{d})$ is generated as following.
 - Choose $z, \alpha, \beta \in_R \mathbb{Z}_p$ and calculate

$$A = \left[\frac{e(svk_{ID}, h_2(m))}{U} \right]^z, B = \frac{[e(h_2(m), g_1)]^\alpha}{U^\beta},$$

$$C = \frac{e(g_1, g_1)^\alpha}{e(h_1(ID \| \text{Undeniable}), mpk)^\beta};$$

- Set $d = h_4(ID_s \| A \| B \| C \| m \| \sigma)$ and calculate $D = g_1^\alpha \cdot (svk_{ID})^{dz}$, $\widehat{d} = \beta + dz \pmod{p}$.

S-Verify: For a selective proof π of the message-signature pair $(m, \sigma = (U, V, W))$ under ID , this algorithm outputs **invalid** if $e(W, g_1) \neq e(h_1(ID), mpk)e(h_3(U, V), V)$. Otherwise,

- If $\pi = (c, C)$, this algorithm outputs **valid** if $c = h_4(ID \| e(C, g_1)e(h_1(ID \| \text{Undeniable}), mpk)^c \| e(C, h_2(m))U^c \| m \| \sigma)$;
- Else, if $\pi = (A, d, D, \widehat{d})$, one calculates $B' = \frac{e(D, h_2(m))}{U^{\widehat{d}} \cdot A^d}$ and $C' = \frac{e(D, g_1)}{e(h_1(ID \| \text{Undeniable}), mpk)^{\widehat{d}}}$. This algorithm outputs **invalid** if $A \neq 1$ and $d = h_4(ID \| A \| B' \| C' \| m \| \sigma)$.

- Otherwise, output “ \perp ”.

The correctness of the our scheme is self-evident. We now prove that our scheme satisfies all the security requirements defined earlier.

3.3.2 Security Analysis: Confirmation and Disavowal

We now show that **Confirmation** and **Disavowal** are zero-knowledge proofs with designated verifier. We will first prove that **Confirmation** satisfies the completeness, soundness and zero-knowledge in the random oracle model.

Completeness of Confirmation.

If $(m, \sigma = (U, V, W))$ is valid under ID , then $U = e(svk_{ID}, h_2(m))$ and $e(W, g_1) = e(h_1(ID), mpk)e(h_3(U, V), V)$. The verifier will output **valid** after receiving the transcript (c_s, c_v, C_s, C_v) of **Confirmation**. To see this, it suffices to show that $R'_1 = R_1$, $R'_2 = R_2$ and $R'_3 = R_3$.

1. As $C_s = g_1^r \cdot (svk_{ID})^{-c_s}$ and $R'_1 = e(C_s, g_1)e(h_1(ID \parallel \text{Undeniable}), mpk)^{c_s}$, we have

$$\begin{aligned}
 R'_1 &= e(g_1^r \cdot (svk_{ID})^{-c_s}, g_1)e(h_1(ID \parallel \text{Undeniable}), mpk)^{c_s} \\
 &= e(g_1, g_1)^r e(svk_{ID}, g_1)^{-c_s} e(h_1(ID \parallel \text{Undeniable}), mpk)^{c_s} \\
 &= e(g_1, g_1)^r e(h_1(ID \parallel \text{Undeniable})^{msk}, g_1)^{-c_s} e(h_1(ID \parallel \text{Undeniable}), mpk)^{c_s} \\
 &= e(g_1, g_1)^r = R_1.
 \end{aligned}$$

2. As $R'_2 = e(C_s, h_2(m))U^{c_s}$, we have

$$\begin{aligned}
 R'_2 &= e(g_1^r \cdot (svk_{ID})^{-c_s}, h_2(m))e(svk_{ID}, h_2(m))^{c_s} \\
 &= e(g_1, h_2(m))^r e(svk_{ID}, h_2(m))^{-c_s} e(svk_{ID}, h_2(m))^{c_s} \\
 &= e(g_1, h_2(m))^r = R_2.
 \end{aligned}$$

3. The calculation of R'_3 is the same as that of R_3 , so we have $R'_3 = R_3$.

This completes the analysis of the completeness of **Confirmation**.

Soundness of Confirmation.

We now prove the soundness of **Confirmation** in the random oracle model. Suppose there is a prover can produce a transcript (c_s, c_v, C_s, C_v) which can make the verifier output **valid**. This means $e(W, g_1) = e(h_1(ID), mpk)e(h_3(U, V), V)$

and $c_s + c_v = h_4(\xi) \pmod{p}$, $\xi = ID \| ID_v \| e(C_s, g_1) e(h_1(ID \| \text{Undeniable}), \text{mpk})^{c_s} \| e(C_s, h_2(m)) U^{c_s} \| e(C_v, g_1) \cdot e(h_1(ID_v), \text{mpk})^{c_v}$.

In the random oracle model, if we assign another value (different from $c_s + c_v$) to $h_4(\xi)$, then this prover will generate another transcript (c'_s, c'_v, C'_s, C'_v) in polynomial time such that $c'_s + c'_v = h_4(\xi) \pmod{p}$, $\xi = ID \| ID_v \| e(C'_s, g_1) e(h_1(ID \| \text{Undeniable}), \text{mpk})^{c'_s} \| e(C'_s, h_2(m)) U^{c'_s} \| e(C'_v, g_1) \cdot e(h_1(ID_v), \text{mpk})^{c'_v}$.

If $c_s + c_v \neq c'_s + c'_v \pmod{p}$, then either $c_s \neq c'_s \pmod{p}$ or $c_v \neq c'_v \pmod{p}$.

1. If $c_s \neq c'_s \pmod{p}$, then σ must be a valid undeniable signature of m under ID . As the input to h_4 is identical, we have

$$e(C_s, g_1) e(h_1(ID \| \text{Undeniable}), \text{mpk})^{c_s} = e(C'_s, g_1) e(h_1(ID \| \text{Undeniable}), \text{mpk})^{c'_s}$$

and $e(C_s, h_2(m)) U^{c_s} = e(C'_s, h_2(m)) U^{c'_s}$. It follows that $(C'_s \cdot C_s^{-1})^{(c_s - c'_s)^{-1}} = \text{svk}_{ID}$ and $U = e(\text{svk}_{ID}, h_2(m))$, which guarantees that σ is a valid signature of m under ID .

2. Otherwise, $c_v \neq c'_v$ and one can extract sk_{ID_v} of ID_v . Again, we have $e(C_v, g_1) \cdot e(h_1(ID_v), \text{mpk})^{c_v} = e(C'_v, g_1) \cdot e(h_1(ID_v), \text{mpk})^{c'_v}$. It follows that $\text{sk}_{ID_v} = (C'_v \cdot C_v^{-1})^{(c_v - c'_v)^{-1}}$.

Zero-Knowledgeness of Confirmation. We show that the verifier with the knowledge sk_{ID_v} can generate the transcript of **Confirmation** for any signature $\sigma \in \mathcal{S}$, which has the same probability distribution (in the random oracle model) as that generated by **Confirmation** with the knowledge of svk_{ID} .

1. We first show that given a valid pair $(m, \sigma = (U, V, W))$, the transcript (c_s, c_v, C_s, C_v) produced by the prover is a random element in $\mathbb{Z}_p^2 \times \mathbb{G}_1^2$ (in the random oracle model).

In the **Confirmation** protocol, c_v is randomly chosen in \mathbb{Z}_p and C_v is a random element in \mathbb{G}_1 . If h_4 is viewed as the random oracle, then its output is independent of c_v, d_v, r , and will be a random element in \mathbb{Z}_p . This leads c_s and C_s to be randomly distributed in \mathbb{Z}_p and \mathbb{G}_1 respectively.

2. With the knowledge sk_{ID_v} , one can also generate a transcript of **Confirmation** for any signature $\sigma = (U, V, W) \in \mathcal{S}$, such that it is a random element in \mathbb{Z}_p^4 (in the random oracle model).

To do this, one first randomly chooses $r, c_s \in \mathbb{Z}_p$ and $C_s \in \mathbb{G}_1$ and calculates

- (a) $R_1 = e(C_s, g_1)e(h_1(ID\|Undeniable), mpk)^{c_s}$;
- (b) $R_2 = e(C_s, h_2(m))U^{c_s}$, $R_3 = e(g_1, g_1)^r$;
- (c) $c = h_4(ID\|ID_v\|R_1\|R_2\|R_3\|m\|\sigma)$, $c_v = c - c_s \pmod{p}$, $C_v = g_1^r \cdot (sk_{ID_v})^{-c_v}$.

The transcript generated by the verifier is (c_s, c_v, C_s, C_v) which is also a random element in $\mathbb{Z}_p^2 \times \mathbb{G}_1^2$ (in the random oracle model).

This completes the analysis that **Confirmation** is a zero-knowledge proof with designated verifier in the random oracle model. The analysis of **Disavowal** is quite similar to that of **Confirmation**, and is thus omitted.

3.3.3 Security Analysis: Unforgeability

The unforgeability of the proposed scheme can be reduced to the hardness of CDH problem defined in Section 1.5.

Theorem 3.1 *The proposed selectively and universally convertible identity-based undeniable signature scheme is $(t, q_{UC}, q_E, q_S, q_H, \varepsilon)$ -existentially unforgeable under an adaptive chosen message and chosen identity attack, if the CDH problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 1), \varepsilon/\mathfrak{e}(q_E + 1))$ -hard on \mathbb{G}_1 . Here $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ is a constant that depends on $(\mathbb{G}_1, \mathbb{G}_T)$, and \mathfrak{e} is the base of the natural logarithm.*

Proof. Suppose there is a forger \mathcal{A} can $(t, q_{UC}, q_E, q_S, q_H, \varepsilon)$ -break the proposed undeniable signature scheme. We show how to construct a $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 1)$ -time algorithm \mathcal{B} that solves CDH on \mathbb{G}_1 with probability at least $\varepsilon/\mathfrak{e}(q_E + 1)$. This will contradict the fact that the CDH problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 1), \varepsilon/\mathfrak{e}(q_E + 1))$ -hard on \mathbb{G}_1 .

Algorithm \mathcal{B} is given $(\mathbb{G}_1, \mathbb{G}_T, e, p, k)$ together with a CDH instance (g_1, g_1^a, g_1^b) , \mathcal{B} 's goal is to obtain $g_1^{ab} \in \mathbb{G}_1$. Algorithm \mathcal{B} simulates the challenger and interacts with forger \mathcal{A} as follows.

Setup: Algorithm \mathcal{B} selects a random element $a \in \mathbb{Z}_p^*$ and calculates $mpk = (g_1^a)^a$.

The adversary is given mpk and the system $param = (\mathbb{G}_1, \mathbb{G}_T, e, p, k, g_1, h_1, h_2, h_3, h_4)$. Here, hash functions are simulated as random oracles as below.

h_1 **Queries:** At any time the forger \mathcal{A} can query the random oracle h_1 . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(ID_j, c_j, \mu_j, \nu_j)$

as explained below. We refer to this list as the h_1 -list. The list is initially empty. For a request ID_i , algorithm \mathcal{B} first checks if the string ID_i ends with “Undeniable”. If so, \mathcal{B} resets ID_i by removing the string “Undeniable” from ID_i .

1. If ID_i already appears on the h_1 -list in a tuple $(ID_i, c_i, \mu_i, \nu_i)$, the algorithm \mathcal{B} responds with $h_1(ID_i) = (g_1^b)^{1-c_i} \cdot g_1^{\mu_i}$ and $h_1(ID_i \parallel \text{Undeniable}) = g_1^{\nu_i}$.
2. Otherwise, ID_i is a fresh query and \mathcal{B} picks a random coin $c_i \in \{0, 1\}$ so that $\Pr[c_i = 0] = \frac{1}{q_E + 1}$.
3. Algorithm \mathcal{B} then randomly chooses $\mu_i, \nu_i \in \mathbb{Z}_p$, and computes $h_1(ID_i) \leftarrow (g_1^b)^{1-c_i} \cdot g_1^{\mu_i} \in \mathbb{G}_1$, $h_1(ID_i \parallel \text{Undeniable}) \leftarrow g_1^{\nu_i}$.
4. At last, \mathcal{B} adds $(ID_i, c_i, \mu_i, \nu_i)$ on the h_1 -list, and sends \mathcal{A} with $h_1(ID_i)$ and $h_1(ID_i \parallel \text{Undeniable})$.

h_2 Queries: At any time, \mathcal{A} can issue an h_2 query for the message m_i . In response, \mathcal{B} will maintain an h_2 -list. For a new query, \mathcal{B} chooses $\omega_i \in_R \mathbb{Z}_p$ and computes $h_2(m_i) = g_1^{\omega_i}$. \mathcal{B} then adds $(m_i, \omega_i, h_2(m_i))$ to h_2 -list and returns $h_2(m_i)$ as the answer. For a query which already appears on the h_2 -list, \mathcal{B} finds the tuple $(m_i, \omega_i, h_2(m_i))$ and responds with $h_2(m_i)$.

h_3 Queries: At any time, \mathcal{A} can issue an h_3 query for the input η_i . In response, \mathcal{B} will maintain an h_3 -list. For a new query, \mathcal{B} first chooses $\ell_i \in_R \mathbb{Z}_p$ and computes $h_3(\eta_i) = g_1^{\ell_i}$. Then, \mathcal{B} adds $(\eta_i, \ell_i, h_3(\eta_i))$ to h_3 -list and returns $h_3(\eta_i)$ as the answer. For a query already appears on the h_3 -list, \mathcal{B} finds the tuple $(\eta_i, \ell_i, h_3(\eta_i))$ and responds with $h_3(\eta_i)$.

h_4 Queries: At any time, \mathcal{A} can issue an h_4 query for the input ξ_i . In response, \mathcal{B} will maintain an h_4 -list. For a new query ξ_i , \mathcal{B} chooses $h_i \in_R \mathbb{Z}_p$ and sets $h_4(\xi_i) = h_i$. \mathcal{B} then adds (ξ_i, h_i) on h_4 -list and returns $h_4(\xi_i)$ as the answer. For a query already appears on the h_4 -list, \mathcal{B} finds the tuple (ξ_i, h_i) and responds with h_i .

User-Create Queries: For a user-create query ID_i , \mathcal{B} will first check h_1 -list.

1. If ID_i already appears in a tuple $(ID_i, c_i, \mu_i, \nu_i)$ on the h_1 -list, \mathcal{B} responds with $svk_i = mpk^{\nu_i}$. Note that svk_i is a correct secret verification key of ID_i as $mpk = g_1^{aa}$ and $h_1(ID_i \parallel \text{Undeniable}) = g_1^{\nu_i}$.

2. Otherwise, \mathcal{B} makes an h_1 request ID_i by itself and there will be a tuple $(ID_i, c_i, \mu_i, \nu_i)$ on h_1 -list. \mathcal{B} responds with $svk_i = mpk^{\nu_i}$.

Signing-Key-Extract Queries: For a signing-key-extract query of a created user ID_i , \mathcal{B} first finds the tuple $(ID_i, c_i, \mu_i, \nu_i)$ on the h_1 -list. Note that $h_1(ID_i) = (g_1^b)^{1-c_i} \cdot g_1^{\mu_i}$ and $h_1(ID_i \parallel \text{Undeniable}) = g_1^{\nu_i}$.

1. If $c_i = 1$, then $h_1(ID_i) = g_1^{\mu_i}$ and the corresponding secret signing key $ssk_i = (mpk^{\mu_i}, mpk^{\nu_i})$.
2. Otherwise, $c_i = 0$ and $h_1(ID_i) = g_1^b \cdot g_1^{\mu_i}$. \mathcal{B} is not able to calculate the corresponding secret key and the simulation fails.

Signature Queries: For a signature query (m_i, ID_i) , \mathcal{B} first finds the tuple $(ID_i, c_i, \mu_i, \nu_i)$ on the h_1 -list.

1. If $c_i = 1$, \mathcal{B} can calculate the secret signing key ssk_i and then calculate the signature by running algorithm **Sign** as described in Section 3.3.1.
2. Otherwise, $c_i = 0$ and \mathcal{B} is not able to calculate the secret signing key in this case, but \mathcal{B} still can generate the signature as described below. We assume that m_i already appears in a tuple $(m_i, \omega_i, h_2(m_i))$ on the h_2 -list (If it does not, \mathcal{B} can make an h_2 request m_i by itself).
 - (a) \mathcal{B} first calculates $svk_i = mpk^{\nu_i}$ and $U_i = e(svk_i, h_2(m_i))$.
 - (b) \mathcal{B} then chooses a random element $k_i \in \mathbb{Z}_p$ and calculates $V_i = (g_1^a)^{-1} \cdot g_1^{k_i}$. If there is a tuple $(\eta_i, \ell_i, h_3(\eta_i))$ on the h_3 -list and $\eta_i = (U_i, V_i)$, \mathcal{B} will rechoose k_i and re-calculate V_i until there is not tuple on the h_3 -list containing (U_i, V_i) .
 - (c) After that, \mathcal{B} chooses a random element $\ell_i \in \mathbb{Z}_p$ and computes $h_3(U_i, V_i) \leftarrow (g_1^b)^a \cdot g_1^{\ell_i}$. \mathcal{B} adds $(\eta_i = (U_i, V_i), \ell_i, h_3(\eta_i))$ on h_3 -list.
 - (d) Finally, \mathcal{B} computes $W_i = (g_1^a)^{a\mu_i - \ell_i} \cdot (g_1^b)^{ak_i} \cdot g_1^{k_i\ell_i}$.

The signature (U_i, V_i, W_i) is returned to \mathcal{A} as the answer. (U_i, V_i, W_i) is a valid signature as $U_i = e(svk_i, h_2(m_i))$ and

$$\begin{aligned}
e(W_i, g_1) &= e((g_1^a)^{a\mu_i - \ell_i} \cdot (g_1^b)^{ak_i} \cdot g_1^{k_i \ell_i}, g_1) \\
&= e((g_1^a)^{a\mu_i - \ell_i} \cdot (g_1^b)^{ak_i} \cdot g_1^{k_i \ell_i} \cdot g_1^{aba} \cdot g_1^{-aba}, g_1) \\
&= e((g_1^b \cdot g_1^{\mu_i})^{aa}, g_1) \cdot e(g_1^{\ell_i(k_i - a)} \cdot (g_1^{ab})^{k_i - a}, g_1) \\
&= e(g_1^b \cdot g_1^{\mu_i}, g_1^{aa}) \cdot e(g_1^{\ell_i} \cdot (g_1^b)^a, (g_1^a)^{-1} \cdot g_1^{k_i}) \\
&= e(h_1(ID_i), mpk) \cdot e(h_3(U_i, V_i), V_i)
\end{aligned}$$

This completes the description of how \mathcal{B} can answer queries from \mathcal{A} . Now we calculate the probability that \mathcal{B} does not abort during the simulation. \mathcal{B} could only fail when \mathcal{A} requests the secret signing key of ID_i and the corresponding $c_i = 0$ in the tuple $(ID_i, c_i, \mu_i, \nu_i)$. As $\Pr[c_i = 0] = \frac{1}{q_E + 1}$, the probability that \mathcal{B} does not fail in the simulation is $(1 - \frac{1}{q_E + 1})^{q_E} \geq 1/\epsilon$. Here, ϵ is the base of natural logarithm.

If \mathcal{B} does not abort, \mathcal{A} will output (m^*, σ^*, ID^*) as the forgery. If $\sigma^* = (U^*, V^*, W^*)$ is a valid signature, then $e(W^*, g_1) = e(h_1(ID^*), mpk)e(h_3(U^*, V^*), V^*)$. Let $((U^*, V^*), \ell^*, h_3(U^*, V^*))$ be the corresponding tuple on h_3 -list. According to the simulation of h_3 , $h_3(U^*, V^*) = g_1^{\ell^*}$. Thus, we have $e(W^*, g_1) = e(h_1(ID^*), mpk)e(V^*, g_1)^{\ell^*}$. It follows that $e(h_1(ID^*), mpk) = e(W^* \cdot (V^*)^{-\ell^*}, g_1)$. Thus, we have $sk_{ID^*} = W^* \cdot (V^*)^{-\ell^*}$. Let $(ID^*, c^*, \mu^*, \nu^*)$ be the corresponding tuple on h_1 -list. If $c^* = 0$ (with probability $\frac{1}{q_E + 1}$), $h_1(ID^*) = g_1^b \cdot g_1^{\mu^*}$ and $sk_{ID^*} = (g_1^b \cdot g_1^{\mu^*})^{aa}$. It follows that $W^* \cdot (V^*)^{-\ell^*} = (g_1^b \cdot g_1^{\mu^*})^{aa}$ and $g_1^{ab} = (W^* \cdot (V^*)^{-\ell^*})^{a^{-1}} \cdot (g_1^a)^{-\mu^*}$.

\mathcal{B} can find the solution g_1^{ab} if

\mathcal{E}_1 : \mathcal{B} does not fail during the simulation; and

\mathcal{E}_2 : \mathcal{A} wins the game by outputting a valid forgery (m^*, σ^*) under ID^* ; and

\mathcal{E}_3 : $c^* = 0$ in the tuple $(ID^*, c^*, \mu^*, \nu^*)$ on h_1 -list.

Therefore, the probability that \mathcal{B} can find the solution g_1^{ab} is $\Pr = [\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] = \epsilon/\epsilon(q_E + 1)$.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to $(q_H + q_{UC} + q_S)$ random oracle queries, q_{UC} user-create queries, q_E key signing-key-extract queries and q_S signature queries and compute g_1^{ab} from σ^* . We assume each takes time at most $c_{(\mathbb{G}_1, \mathbb{G}_T)}$, which is a constant on $(\mathbb{G}_1, \mathbb{G}_T)$. Hence, the total running time is at most $t + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_{UC} + q_E + 2q_S + 1)$. This completes the proof of Theorem 3.1.

Remark: Actually, the proposed scheme achieves the strong unforgeability. To see this, let $\text{Set} = \{(m^*, U^*, V_1, W_1), (m^*, U^*, V_2, W_2), \dots, (m^*, U^*, V_n, W_n)\}$ be the set of \mathcal{B} 's responses to the adversary \mathcal{A} 's signature queries on (m^*, ID^*) . For \mathcal{A} 's forgery (m^*, U^*, V^*, W^*) , if there is a tuple $(m^*, U^*, V_i, W_i) \in \text{Set}$ such that $V_i = V^*$, then $W_i = W^*$ as $e(W_i, g_1) = e(h_1(ID)^*, mpk)e(h_3(U^*, V_i), V_i)$ and $e(W^*, g_1) = e(h_1(ID)^*, mpk)e(h_3(U^*, V^*), V^*)$. It follows that $(m^*, U^*, V_i, W_i) = (m^*, U^*, V^*, W^*)$. As \mathcal{A} is not allowed to output any tuple in Set as the forgery, V^* must be different from any V_i in Set . Thus, we have $h_3(U^*, V^*) = g_1^{\ell^*}$ and \mathcal{B} can output g_1^{ab} with the same probability as described in the proof of Theorem 3.1.

3.3.4 Security Analysis: Invisibility

The invisibility of the proposed scheme can be reduced to the hardness of DBDH problem defined in Section 1.5.

Theorem 3.2 *The proposed selectively and universally convertible identity-based undeniable signature scheme is $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, q_H, \varepsilon)$ -indistinguishable under an adaptive chosen message and chosen identity attack if Decisional Bilinear Diffie-Hellman problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1), \frac{2\varepsilon}{\epsilon^{3(q_E+1)(q_S+1)}})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$. Here $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ is a constant that depends on $(\mathbb{G}_1, \mathbb{G}_T)$, and ϵ is the base of natural logarithm.*

Proof. Suppose there is a distinguisher \mathcal{A} can $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, q_H, \varepsilon)$ -break the invisibility of the proposed undeniable signature scheme. We show how to construct a $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1)$ -time algorithm \mathcal{B} that solves the DBDH problem on $(\mathbb{G}_1, \mathbb{G}_T)$ with advantage at least $\frac{2\varepsilon}{\epsilon^{3(q_E+1)(q_S+1)}}$. This will contradict the fact that DBDH is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1), \frac{2\varepsilon}{\epsilon^{3(q_E+1)(q_S+1)}})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$.

Algorithm \mathcal{B} is given $(\mathbb{G}_1, \mathbb{G}_T, e, p, k)$ together with a DBDH instance $(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h})$, \mathcal{B} 's goal is to determine if $\mathfrak{h} = e(g_1, g_1)^{abc}$. Algorithm \mathcal{B} simulates the challenger and interacts with \mathcal{A} as follows.

Setup: Algorithm \mathcal{B} selects a random element $a \in \mathbb{Z}_p^*$ and calculates $mpk = (g_1^a)^a$.

The adversary is given mpk and the system $param = (\mathbb{G}_1, \mathbb{G}_T, e, p, k, g_1, h_1, h_2, h_3, h_4)$. Here, hash functions are simulated as random oracles as below.

h_1 **Queries:** At any time the distinguisher \mathcal{A} can query the random oracle h_1 . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(ID_j, c_j, \mu_j, \nu_j)$

as explained below. We refer to this list as the h_1 -list. The list is initially empty. For a request ID_i , algorithm \mathcal{B} first checks if the string ID_i ends with “Undeniable”. If so, \mathcal{B} resets ID_i by removing the string “Undeniable” from ID_i .

1. If ID_i already appears on the h_1 -list in a tuple $(ID_i, c_i, \mu_i, \nu_i)$, the algorithm \mathcal{B} responds with $h_1(ID_i) = g_1^{\mu_i}$, $h_1(ID_i \parallel \text{Undeniable}) = (g_1^b)^{1-c_i} \cdot g_1^{\nu_i}$.
2. Otherwise, ID_i is a fresh query and \mathcal{B} picks a random coin $c_i \in \{0, 1\}$ so that $\Pr[c_i = 0] = \frac{1}{q_E+1}$.
3. Algorithm \mathcal{B} then randomly chooses $\mu_i, \nu_i \in \mathbb{Z}_p$, and computes $h_1(ID_i) \leftarrow g_1^{\mu_i}$, $h_1(ID_i \parallel \text{Undeniable}) \leftarrow (g_1^b)^{1-c_i} \cdot g_1^{\nu_i}$.
4. At last, \mathcal{B} adds $(ID_i, c_i, \mu_i, \nu_i)$ on the h_1 -list, and sends $h_1(ID_i)$ and $h_1(ID_i \parallel \text{Undeniable})$ to \mathcal{A} .

h_2 Queries: At any time, \mathcal{A} can issue an h_2 query for the message m_i . In response, \mathcal{B} will maintain an h_2 -list consisting of tuples $(m_j, \varpi_j, \omega_j, h_2(m_j))$. For a query m_i ,

1. If m_i already appears in a tuple $(m_i, \varpi_i, \omega_i, h_2(m_i))$ on the h_2 -list, \mathcal{B} responds with $h_2(m_i)$.
2. Otherwise, m_i is a new query and \mathcal{B} picks a random coin $\varpi_i \in \{0, 1\}$ so that $\Pr[\varpi_i = 0] = \frac{1}{q_S+1}$.
3. Algorithm \mathcal{B} then randomly chooses $\omega_i \in \mathbb{Z}_p$, and computes $h_2(m_i) \leftarrow (g_1^c)^{1-\varpi_i} \cdot g_1^{\omega_i} \in \mathbb{G}_1$.
4. At last, \mathcal{B} adds $(m_i, \varpi_i, \omega_i, h_2(m_i))$ on h_2 -list and responds with $h_2(m_i)$.

h_3 and h_4 queries: \mathcal{B} answers these queries as same as described in the proof of Theorem 3.1.

User-Create Queries: For a user-create query ID_i , \mathcal{B} will first check h_1 -list.

1. If there is no tuple $(ID_i, c_i, \mu_i, \nu_i)$ on h_1 -list containing ID_i , \mathcal{B} makes an h_1 request ID_i by itself. After that, ID_i appears in a tuple $(ID_i, c_i, \mu_i, \nu_i)$ on h_1 -list.
2. \mathcal{B} first calculates $sk_i = mpk^{\mu_i}$. If $c_i = 1$, $h_1(ID_i \parallel \text{Undeniable}) = g_1^{\nu_i}$ and $svk_i = mpk^{\nu_i}$. \mathcal{B} then sets $ssk_i = (sk_i, svk_i)$ and adds $(ID_i, c_i, ssk_i, svk_i)$

on *user-list*. Otherwise, $c_i = 0$ and \mathcal{B} is not able to calculate svk_i . \mathcal{B} will add $(ID_i, c_i, \perp, \perp)$ on *user-list*.

Key-Extract Queries: For a key extract query of a created user ID_i , \mathcal{B} will first find the tuple $(ID_i, c_i, ssk_i, svk_i)$ on *user-list*. If $c_i = 1$, (ssk_i, svk_i) will be given to \mathcal{A} . Otherwise, $c_i = 0$ and \mathcal{B} fails in the simulation.

Signature Queries: For a signature query (m_i, ID_i) , \mathcal{B} first finds the tuple $(ID_i, c_i, \mu_i, \nu_i)$ on the h_1 -list and the tuple $(m_i, \varpi_i, \omega_i, h_2(m_i))$ on the h_2 -list.

1. If $c_i = 1$, \mathcal{B} is able to calculate ssk_i and svk_i . \mathcal{B} thus can calculate the signature (U_i, V_i, W_i) by running algorithm **Sign** as described in Section 3.3.1.
2. Else if $c_i = 0, \varpi_i = 1$, \mathcal{B} cannot calculate svk_i but still can calculate the signature. As $\varpi_i = 1$, $h_2(m_i) = g_1^{\omega_i}$. Therefore, \mathcal{B} can compute $U_i = e(svk_i, h_2(m_i)) = e(svk_i, g_1)^{\omega_i} = e(h_1(ID_i \parallel \text{Undeniable}), mpk)^{\omega_i}$. The calculation of V_i and W_i is the same as in algorithm **Sign** described in Section 3.3.1 as \mathcal{B} can calculate $sk_i = mpk^{\mu_i}$.
3. Otherwise, $c_i = 0, \varpi_i = 0$ and \mathcal{B} fails in the simulation.

Verification Queries: For a verification query (m_i, σ_i, ID_i) where $\sigma_i = (U_i, V_i, W_i)$ and $e(W_i, g_1) = e(h_1(ID_i), mpk)e(h_3(U_i, V_i), V_i)$, we assume that m_i already appears in the tuple $(m_i, \varpi_i, \omega_i, h_2(m_i))$ on the h_2 -list. \mathcal{B} responds as follows.

1. If m_i already appears as one of signature queries and \mathcal{B} does not abort, \mathcal{B} can first calculate $e(svk_i, h_2(m_i))$ as described in **Signature Queries**.
 - (a) If $U_i = e(svk_i, h_2(m_i))$. \mathcal{B} will generate a transcript of **Confirmation** with designated verifier ID_v as follows. \mathcal{B} first randomly picks c_s, c_v from \mathbb{Z}_p and C_s, C_v from \mathbb{G}_1 . After that, \mathcal{B} computes $R_1 = e(C_s, g_1) \cdot e(h_1(ID_i \parallel \text{Undeniable}), mpk)^{c_s}$, $R_2 = e(C_s, h_2(m_i))U_i^{c_s}$, $R_3 = e(C_v, g_1) \cdot e(h_1(ID_v), mpk)^{c_v}$. If there is a tuple on the h_4 -list containing the string $\xi_i = "ID_i \parallel ID_v \parallel R_1 \parallel R_2 \parallel R_3 \parallel m_i \parallel \sigma_i"$, \mathcal{B} will rechoose (c_s, c_v, C_s, C_v) until ξ_i is a fresh h_4 query. \mathcal{B} then adds $(\xi_i, c_s + c_v \pmod{p})$ on h_4 -list and returns (c_s, c_v, C_s, C_v) as the answer.
 - (b) Otherwise, $U_i \neq e(svk_i, h_2(m_i))$. \mathcal{B} will generate a transcript of **Disavowal** with designated verifier ID_v as follows. \mathcal{B} picks a random element $A \neq 1 \in \mathbb{G}_T$, three random elements $d_s, d_v, \hat{d}_s \in \mathbb{Z}_p$ and

two random elements $D_s, D_v \in \mathbb{G}_1$. After that, \mathcal{B} calculates

$$B = \frac{e(D_s, h_2(m))}{U_i^{\hat{d}_s} \cdot A^{d_s}}, C = \frac{e(D_s, g_1)}{e(h_1(ID \parallel \text{Undeniable}), \text{mpk})^{\hat{d}_s}},$$

and $D = e(D_v, g_1)e(h_1(ID_v), \text{mpk})^{d_v}$. If there is a tuple on the h_4 -list containing the string $\xi_i = "ID_i \parallel ID_v \parallel A \parallel B \parallel C \parallel D \parallel m_i \parallel \sigma_i"$, \mathcal{B} will rechoose $(A, d_s, d_v, \hat{d}_s, D_s, D_v)$ until ξ_i is a fresh h_4 query. \mathcal{B} then adds $(\xi_i, d_s + d_v \pmod{p})$ on h_4 -list and returns $(A, d_s, d_v, \hat{d}_s, D_s, D_v)$ as the answer.

2. Otherwise, m_i does not appear as one of signature queries and \mathcal{B} will generate a transcript of **Disavowal** as described above. The answer could be incorrect if σ_i is a valid signature of m_i . According to Theorem 3.1, this happens with probability less than $1/(q_{SC} + q_V + 1)$ if CDH is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 1), \frac{1}{e_{(q_{SC} + q_V + 1) \cdot (q_E + 1)}})$ -hard on \mathbb{G}_1 .

Selectively-Convert Queries: For a selectively-convert query (m_i, σ_i, ID_i) where $\sigma_i = (U_i, V_i, W_i)$ and $e(W_i, g_1) = e(h_1(ID_i), \text{mpk})e(h_3(U_i, V_i), V_i)$, we assume that m_i already appears in the tuple $(m_i, \varpi_i, \omega_i, h_2(m_i))$ on the h_2 -list. \mathcal{B} responds as follows.

1. If m_i already appears as one of signature queries and \mathcal{B} does not abort, \mathcal{B} can first calculate $e(\text{svk}_i, h_2(m_i))$ as described in **Signature Queries**.
 - (a) If $U_i = e(\text{svk}_i, h_2(m_i))$. \mathcal{B} will generate a transcript to prove it as valid using the same technique as it responds **Verification Queries**.
 - (b) Otherwise, $U_i \neq e(\text{svk}_i, h_2(m_i))$. \mathcal{B} will generate a transcript to prove it as invalid using the same technique as it responds **Verification Queries**.
2. Otherwise, m_i does not appear as one of signature queries and \mathcal{B} will generate a transcript to prove it as invalid as described above. The answer could be incorrect if σ_i is a valid signature of m_i . According to Theorem 3.1, this happens with probability less than $1/(q_{SC} + q_V + 1)$ if CDH is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 1), \frac{1}{e_{(q_{SC} + q_V + 1) \cdot (q_E + 1)}})$ -hard on \mathbb{G}_1 .

Output I: \mathcal{A} outputs a message m^* and a created user ID^* . There must exist a tuple $(ID^*, c^*, \mu^*, \nu^*)$ on h_1 -list. We assume that m^* already appears in the

tuple $(m_i, \varpi^*, \omega^*, h_2(m^*))$ on h_2 -list (If it does not, \mathcal{B} can make an h_2 request of m^*). If $c^* = 0$ and $\varpi^* = 0$, then $h_1(ID^* \parallel \text{Undeniable}) = g_1^b \cdot g_1^{\nu^*}$ and $h_2(m^*) = g_1^c \cdot g_1^{\omega^*}$. In a valid undeniable signature (U^*, V^*, W^*) of m^* ,

$$\begin{aligned} U^* &= e(\text{svk}^*, h_2(m^*)) \\ &= e(h_1(ID^* \parallel \text{Undeniable})^{aa}, h_2(m^*)) \\ &= e(g_1^b \cdot g_1^{\nu^*}, g_1^c \cdot g_1^{\omega^*})^{aa} \\ &= e(g_1, g_1)^{aabc} \cdot e(g_1^b, g_1^a)^{a\omega^*} \cdot e(g_1^a, h_2(m^*))^{\nu^*a} \end{aligned}$$

In the simulation, \mathcal{B} computes $U^* \leftarrow \mathfrak{h}^a \cdot e(g_1^b, g_1^a)^{a\omega^*} \cdot e(g_1^a, h_2(m^*))^{\nu^*a}$. After that, \mathcal{B} generates V^*, W^* as described in algorithm **Sign** in Section 3.3.1 with $sk^* = \text{mpk}^{\mu^*}$.

Queries II: \mathcal{A} continues to issue queries and \mathcal{B} responds as same as described above.

Output II: Finally, \mathcal{A} outputs its guess b' .

This completes the simulation. We first calculate the probability that \mathcal{B} does not abort during the simulation.

1. \mathcal{B} could fail in **Key-Extract Queries** if $c_i = 0$. Thus, the probability that \mathcal{B} does not fail in such queries is $(1 - \frac{1}{q_E+1})^{q_E} \geq 1/\mathfrak{e}$.
2. \mathcal{B} could also fail in **Signature Queries** if $c_i = 0$ and $\varpi_i = 0$. Thus, the probability that \mathcal{B} does not fail in such queries is $(1 - \frac{1}{(q_E+1)(q_S+1)})^{q_S} \geq 1/\mathfrak{e}$.
3. \mathcal{B} could also fail in **Verification Queries** or **Selective-Convert Queries** if σ_i is a valid signature of m_i which does not appear as one of signature queries. Thus, the probability that \mathcal{B} does not fail in such queries is $(1 - \frac{1}{q_V+q_{SC}+1})^{q_V+q_{SC}} \geq 1/\mathfrak{e}$.
4. $c^* = 0$ and $\varpi^* = 0$ in **Output I**, which happens with probability $\frac{1}{(q_E+1)(q_S+1)}$.

Therefore, the probability that \mathcal{B} does not fail during the simulation is $\frac{1}{\mathfrak{e}^3(q_E+1)(q_S+1)}$.

If \mathcal{B} does not abort during the simulation, \mathcal{B} will set b' as its own output. If $\mathfrak{h} = e(g_1, g_1)^{abc}$ in the instance of the DBDH problem, then σ^* will be a valid undeniable signature. If \mathcal{A} can $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, \varepsilon)$ -break the invisibility of

the proposed scheme, then $b' = 1$ happens with the probability at least $1/2 + \varepsilon$. Therefore,

$$\Pr[\mathcal{B}(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h}) = 1 : \mathfrak{h} = e(g_1, g_1)^{abc}] \geq 1/2 + \varepsilon.$$

Otherwise, $\mathfrak{h} \neq e(g_1, g_1)^{abc}$, \mathcal{A} will output 1 with probability at most $1/2 - \varepsilon$. Thus,

$$\Pr[\mathcal{B}(g_1, g_1^a, g_1^b, g_1^c, \mathfrak{h}) = 1 : \mathfrak{h} \neq e(g_1, g_1)^{abc}] \leq 1/2 - \varepsilon.$$

It follows that $\text{Adv DBDH}_{\mathcal{B}} \geq 2\varepsilon$ if \mathcal{B} does not abort during the simulation. As the probability that \mathcal{B} does not abort during the simulation is $\frac{1}{\epsilon^3(q_E+1)(q_S+1)}$, the advantage that \mathcal{B} has is $\frac{2\varepsilon}{\epsilon^3(q_E+1)(q_S+1)}$.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to $(q_H + q_{UC} + q_S + q_V + q_{SC})$ random oracle queries, q_{UC} user-create queries, q_E key-extract queries, q_V verification queries, q_{SC} selectively convert queries, q_S signature queries and calculate σ^* . We assume each takes time at most $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ which is a constant on $(\mathbb{G}_1, \mathbb{G}_T)$. Hence, the total running time is at most $t + c_{\mathbb{G}_1}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1)$. This completes the proof of Theorem 3.2.

3.3.5 Security Analysis: S-Convert

The algorithm **S-Convert** is almost the same as **Confirmation** and **Disavowal** protocols, with the only difference that there is no designated verifier in **S-Convert**. Thus, the analysis of the Soundness of **S-Convert** is almost the same as that in **Confirmation** and **Disavowal** protocols. We now prove that the unforgeability of **S-Convert** can be reduced to the hardness of CDH problem defined in Section 1.5.

Theorem 3.3 *The algorithm **S-Convert** in the proposed selectively and universally convertible identity-based undeniable signature scheme is $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, q_H, \varepsilon)$ -existentially unforgeable under an adaptive chosen message and chosen identity attack if CDH problem is $(23q_H q_{SC} t / \varepsilon + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1), \frac{1}{9\epsilon})$ -hard on \mathbb{G}_1 . Here $c_{(\mathbb{G}_1, \mathbb{G}_2)}$ is a constant on $(\mathbb{G}_1, \mathbb{G}_T)$ and ϵ is the base of natural logarithm.*

Proof. Suppose there is a forger \mathcal{A} can $(t, q_{UC}, q_E, q_S, q_V, q_{SC}, q_H, \varepsilon)$ -break the unforgeability of algorithm **S-Convert** in the proposed undeniable signature scheme. We show how to construct a $23q_H q_{SC} t / \varepsilon + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1)$ -time algorithm \mathcal{B} that solves the CDH problem on \mathbb{G}_1 with probability at least $\frac{1}{9\epsilon}$.

This will contradict the fact that CDH is $(23q_H q_{SCt}/\varepsilon + c_{(\mathbb{G}_1, \mathbb{G}_2)}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1), \frac{1}{9\varepsilon})$ -hard on \mathbb{G}_1 .

Algorithm \mathcal{B} is given $(\mathbb{G}_1, \mathbb{G}_T, e, p, k)$ together with a CDH instance (g_1, g_1^a, g_1^b) , \mathcal{B} 's goal is to calculate g_1^{ab} . Algorithm \mathcal{B} simulates the challenger and interacts with \mathcal{A} as follows.

Setup: Algorithm \mathcal{B} selects a random element $a \in \mathbb{Z}_p^*$ and calculates $mpk = (g_1^a)^a$.

The adversary is given mpk and the system $param = (\mathbb{G}_1, \mathbb{G}_T, e, p, k, g_1, h_1, h_2, h_3, h_4)$. Here, hash functions are simulated as random oracles as below.

h_1 Queries: At any time the forger \mathcal{A} can query the random oracle h_1 . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(ID_j, c_j, \mu_j, \nu_j)$ as explained below. We refer to this list as the h_1 -list. The list is initially empty. For a request ID_i , algorithm \mathcal{B} first checks if the string ID_i ends with “Undeniable”. If so, \mathcal{B} resets ID_i by removes the string “Undeniable” from ID_i .

1. If ID_i already appears on the h_1 -list in a tuple $(ID_i, c_i, \mu_i, \nu_i)$, the algorithm \mathcal{B} responds with $h_1(ID_i) = g_1^{\mu_i}$, $h_1(ID_i \parallel \text{Undeniable}) = (g_1^b)^{1-c_i} \cdot g_1^{\nu_i}$.
2. Otherwise, ID_i is a fresh query and \mathcal{B} picks a random coin $c_i \in \{0, 1\}$ so that $\Pr[c_i = 0] = \frac{1}{q_E + 1}$.
3. Algorithm \mathcal{B} then randomly chooses $\mu_i, \nu_i \in \mathbb{Z}_p$, and computes $h_1(ID_i) \leftarrow g_1^{\mu_i}$, $h_1(ID_i \parallel \text{Undeniable}) \leftarrow (g_1^b)^{1-c_i} \cdot g_1^{\nu_i}$.
4. At last, \mathcal{B} adds $(ID_i, c_i, \mu_i, \nu_i)$ on h_1 -list, and sends $h_1(ID_i)$ and $h_1(ID_i \parallel \text{Undeniable})$ to \mathcal{A} .

h_2 Queries: At any time, \mathcal{A} can issue an h_2 query for the message m_i . In response, \mathcal{B} will maintain an h_2 -list. For a new query, \mathcal{B} chooses $\omega_i \in_R \mathbb{Z}_p$ and computes $h_2(m_i) = g_1^{\omega_i}$. \mathcal{B} then adds $(m_i, \omega_i, h_2(m_i))$ to h_2 -list and returns $h_2(m_i)$ as the answer. For a query which already appears on h_2 -list, \mathcal{B} finds the tuple $(m_i, \omega_i, h_2(m_i))$ and responds with $h_2(m_i)$.

h_3 Queries: At any time, \mathcal{A} can issue an h_3 query for the input η_i . In response, \mathcal{B} will maintain an h_3 -list. For a new query, \mathcal{B} first chooses $\ell_i \in_R \mathbb{Z}_p$ and computes $h_3(\eta_i) = g_1^{\ell_i}$. Then, \mathcal{B} adds $(\eta_i, \ell_i, h_3(\eta_i))$ to h_3 -list and returns $h_3(\eta_i)$ as the answer. For a query already appears on h_3 -list, \mathcal{B} finds the tuple $(\eta_i, \ell_i, h_3(\eta_i))$ and responds with $h_3(\eta_i)$.

h_4 Queries: At any time, \mathcal{A} can issue an h_4 query for the input ξ_i . In response, \mathcal{B} will maintain an h_4 -list. For a new query ξ_i , \mathcal{A} chooses $h_i \in_R \mathbb{Z}_p$ and computes $h_4(\xi_i) = h_i$. \mathcal{B} then adds (ξ_i, h_i) on h_4 -list and returns $h_4(\xi_i)$ as the answer. For a query already appears on h_4 -list, \mathcal{B} finds the tuple (ξ_i, h_i) and responds with h_i .

User-Create Queries: For a user-create query ID_i , \mathcal{B} will first check h_1 -list.

1. If there is no tuple $(ID_i, c_i, \mu_i, \nu_i)$ on h_1 -list containing ID_i , \mathcal{B} makes an h_1 request ID_i by itself. After that, ID_i appears in a tuple $(ID_i, c_i, \mu_i, \nu_i)$ on h_1 -list.
2. \mathcal{B} first calculates $sk_i = mpk^{\mu_i}$. If $c_i = 1$, $h_1(ID_i \parallel \text{Undeniable}) = g_1^{\nu_i}$ and $svk_i = mpk^{\nu_i}$. \mathcal{B} then sets $ssk_i = (sk_i, svk_i)$ and adds $(ID_i, c_i, ssk_i, svk_i)$ on $user$ -list. Otherwise, $c_i = 0$ and \mathcal{B} is not able to calculate svk_i . \mathcal{B} will add $(ID_i, c_i, \perp, \perp)$ on $user$ -list.

Key-Extract Queries: For a key extract query of a created user ID_i , \mathcal{B} will first find the tuple $(ID_i, c_i, ssk_i, svk_i)$ on $user$ -list. If $c_i = 1$, (ssk_i, svk_i) will be given to \mathcal{A} . Otherwise, $c_i = 0$ and \mathcal{B} fails in the simulation.

Signature Queries: For a signature query (m_i, ID_i) , \mathcal{B} first finds the tuple $(ID_i, c_i, \mu_i, \nu_i)$ on h_1 -list and the tuple $(m_i, \omega_i, h_2(m_i))$ on h_2 -list.

1. If $c_i = 1$, \mathcal{B} is able to calculate ssk_i and svk_i . \mathcal{B} thus can calculate the signature (U_i, V_i, W_i) by running algorithm **Sign** as described in Section 3.3.1.
2. Otherwise, $c_i = 0$, \mathcal{B} cannot calculate svk_i but still can calculate the signature. As $h_2(m_i) = g_1^{\omega_i}$, \mathcal{B} can compute $U_i = e(svk_i, h_2(m_i)) = e(svk_i, g_1)^{\omega_i} = e(h_1(ID_i \parallel \text{Undeniable}), mpk)^{\omega_i}$. The calculation of V_i and W_i is the same as in algorithm **Sign** described in Section 3.3.1 as \mathcal{B} can calculate $sk_i = mpk^{\mu_i}$.

In either case, \mathcal{B} can calculate a valid signature which will be returned to \mathcal{A} .

Verification Queries: For a verification query (m_i, σ_i, ID_i) where $\sigma_i = (U_i, V_i, W_i)$ and $e(W_i, g_1) = e(h_1(ID_i), mpk)e(h_3(U_i, V_i), V_i)$, we assume that m_i already appears in the tuple $(m_i, \omega_i, h_2(m_i))$ on h_2 -list. \mathcal{B} responds as follows.

1. If $U_i = e(h_1(ID_i \parallel \text{Undeniable}), \text{mpk})^{\omega_i} = e(\text{svk}_i, h_2(m_i))$. \mathcal{B} will generate a transcript of **Confirmation** with designated verifier ID_v as follows. \mathcal{B} first randomly picks c_s, c_v from \mathbb{Z}_p and C_s, C_v from \mathbb{G}_1 . After that, \mathcal{B} computes

$$\begin{aligned} R_1 &= e(C_s, g_1) \cdot e(h_1(ID_i \parallel \text{Undeniable}), \text{mpk})^{c_s}; \\ R_2 &= e(C_s, h_2(m_i)) U_i^{c_s}; \\ R_3 &= e(C_v, g_1) \cdot e(h_1(ID_v), \text{mpk})^{c_v} \end{aligned}$$

If there is a tuple on h_4 -list containing the string

$$\xi_i = "ID_i \parallel ID_v \parallel R_1 \parallel R_2 \parallel R_3 \parallel m_i \parallel \sigma_i",$$

\mathcal{B} will rechoose (c_s, c_v, C_s, C_v) until ξ_i is a fresh h_4 query. \mathcal{B} then adds $(\xi_i, c_s + c_v \pmod{p})$ on h_4 -list and returns (c_s, c_v, C_s, C_v) as the answer.

2. Otherwise, $U_i \neq e(\text{svk}_i, h_2(m_i))$. \mathcal{B} will generate a transcript of **Disavowal** with designated verifier ID_v as follows. \mathcal{B} picks a random element $A \neq 1 \in \mathbb{G}_T$, three random elements $d_s, d_v, \hat{d}_s \in \mathbb{Z}_p$ and two random elements $D_s, D_v \in \mathbb{G}_1$. After that, \mathcal{B} calculates

$$B = \frac{e(D_s, h_2(m))}{U_i^{\hat{d}_s} \cdot A^{d_s}}, C = \frac{e(D_s, g_1)}{e(h_1(ID \parallel \text{Undeniable}), \text{mpk})^{\hat{d}_s}},$$

and $D = e(D_v, g_1) e(h_1(ID_v), \text{mpk})^{d_v}$. If there is a tuple on h_4 -list containing the string $\xi_i = "ID_i \parallel ID_v \parallel A \parallel B \parallel C \parallel D \parallel m_i \parallel \sigma_i"$, \mathcal{B} will rechoose $(A, d_s, d_v, \hat{d}_s, D_s, D_v)$ until ξ_i is a fresh h_4 query. \mathcal{B} then adds $(\xi_i, d_s + d_v \pmod{p})$ on h_4 -list and returns $(A, d_s, d_v, \hat{d}_s, D_s, D_v)$ as the answer.

Selectively-Convert Queries: For a selectively-convert query (m_i, σ_i, ID_i) where $\sigma_i = (U_i, V_i, W_i)$ and $e(W_i, g_1) = e(h_1(ID_i), \text{mpk}) e(h_3(U_i, V_i), V_i)$, we assume that m_i already appears in the tuple $(m_i, \omega_i, h_2(m_i))$ on h_2 -list. \mathcal{B} responds as follows.

1. If $U_i = e(h_1(ID_i \parallel \text{Undeniable}), \text{mpk})^{\omega_i} = e(\text{svk}_i, h_2(m_i))$. \mathcal{B} will generate a transcript of **Confirmation** using the same technique as it responds **Verification Queries**.
2. Otherwise, $U_i \neq e(\text{svk}_i, h_2(m_i))$. \mathcal{B} will generate a transcript of **Disavowal** using the same technique as it responds **Verification Queries**.

Output: Let \mathcal{A} 's output be $(m^*, \sigma^* = (U^*, V^*, W^*), ID^*)$ and a selective-proof π^* .

If \mathcal{A} can win the game with probability at least ε , then $\mathbf{S}\text{-Verify}(param, mpk, ID^*, m^*, \sigma^*, \pi^*) \neq \perp$, ID^* is not one of **Key-Extract Queries** and (m^*, σ^*, ID^*) has never been chosen as one of selectively-convert queries. We distinguish the following two cases: (1) $\mathbf{S}\text{-Verify}(param, mpk, ID^*, m^*, \sigma^*, \pi^*) = \text{valid}$; and (2) $\mathbf{S}\text{-Verify}(param, mpk, ID^*, m^*, \sigma^*, \pi^*) = \text{invalid}$. We will show that in either case, \mathcal{B} can solve the given instance of CDH problem.

Suppose $\mathbf{S}\text{-Verify}(param, mpk, ID^*, m^*, \sigma^*, \pi^*) = \text{valid}$, we have $\pi^* = (c^*, C^*)$ and $c^* = h_2(\xi)$, where $\xi = ID^* \| e(C^*, g_1) e(h_1(ID^* \| \text{Undeniable}), mpk)^{c^*} \| e(C^*, h_2(m^*)) \cdot (U^*)^{c^*} \| m^* \| \sigma^*$. Due to the forking lemma [PS00], if \mathcal{B} assigns another value c' (different from c^*) to $h_2(\xi)$, then \mathcal{A} will generate another proof (c', C') with probability greater than $1/9$ in time $23q_H q_{SC} t / \varepsilon$ such that $c' = h_2(\xi)$ and

$$\xi = ID^* \| e(C', g_1) e(h_1(ID^* \| \text{Undeniable}), mpk)^{c'} \| e(C', h_2(m^*)) (U^*)^{c'} \| m^* \| \sigma^*.$$

As the input to h_4 is identical, we have

$$e(C^*, g_1) e(h_1(ID^* \| \text{Undeniable}), mpk)^{c^*} = e(C', g_1) e(h_1(ID^* \| \text{Undeniable}), mpk)^{c'}.$$

It follows that

$$e(h_1(ID^* \| \text{Undeniable}), mpk)^{c^* - c'} = e(C' \cdot (C^*)^{-1}, g_1), \text{svk}^* = (C' \cdot (C^*)^{-1})^{(c^* - c')^{-1}}.$$

Let $(ID^*, c_{ID^*}, \mu^*, \nu^*)$ be the tuple on h_1 -list containing ID^* . If $c_{ID^*} = 0$, then $h_1(ID^* \| \text{Undeniable}) = g_1^b \cdot g_1^{\nu^*}$ and the corresponding $\text{svk}^* = h_1(ID^* \| \text{Undeniable})^{aa} = (g_1^{ab})^a \cdot (g_1^a)^{a\nu^*}$. Thus, \mathcal{B} can output $g_1^{ab} = (C' \cdot (C^*)^{-1})^{a^{-1}(c^* - c')^{-1}} \cdot (g_1^a)^{-\nu^*}$ as the solution of the given CDH problem. For the case $\mathbf{S}\text{-Verify}(param, mpk, ID^*, m^*, \sigma^*, \pi^*) = \text{invalid}$, \mathcal{B} can compute g_1^{ab} using the same technique.

It remains to calculate the probability that \mathcal{B} does not fail in the simulation. As \mathcal{B} could only fail in responding **Key-Extract** queries, the probability that \mathcal{B} does not fail is $(1 - \frac{1}{q_E + 1})^{q_E} \geq 1/e$. Here, e is the base of natural logarithm. Thus, the probability \mathcal{B} can compute g_1^{ab} is $\frac{1}{9e}$.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to $(q_H + q_{UC} + q_S + q_V + q_{SC})$ random oracle queries, q_{UC} user-create queries, q_E key-extract queries, q_V verification queries, q_S signature queries, q_{SC} selectively convert queries and calculate g_1^{ab} . We assume each takes time at most $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ which is a constant on $(\mathbb{G}_1, \mathbb{G}_T)$. Hence, the total running time is at most $23q_H q_{SV} t / \varepsilon + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + 2q_{UC} + q_E + 2q_S + 2q_V + 2q_{SC} + 1)$. This completes the proof of Theorem 3.3.

3.4 Conclusion

This chapter introduced the first construction of selectively and universally convertible undeniable signature where there is no certificate and a user's public key is his/her identity. The construction is based on the bilinear map over elliptic curves. Its unforgeability can be reduced to the hardness of Computational Diffie-Hellman problem in the random oracle model, and its invisibility can be reduced to the hardness of Decisional Bilinear Diffie-Hellman problem.

Chapter 4

Short (Identity-based) Strong Designated Verifier Signatures

This chapter describes two new constructions of strong designated verifier signatures with short signature length. The designated verifier signature scheme in Section 4.3 and the identity-based designated verifier signature scheme in Section 4.4.3 were presented at *ISPEC 2006* [HSMZ06b] and the full version of [HSMZ06b] was published in the *International Journal of Network Security* [HSMZ08].

4.1 Introduction

The concept of undeniable signatures was introduced by Chaum and van Antwerpen [CA89] for signers to have a complete control over their signatures. The validity of undeniable signatures can only be verified via the *confirmation/disavowal* protocol in collaboration with the signer. As shown in Section 1.1.2, it is also important that the prover must know to whom he/she is proving the validity of a signature; otherwise undeniable signatures may be vulnerable to blackmailing [DY91, Jak94] and mafia attacks [DGB87]. Jakobsson, Sako and Impagliazzo proposed the notion of *designated-verifier signatures* (DVS) [JSI96], where signatures provide authentication of messages *without* providing the non-repudiation property of ordinary digital signatures. By using a designated verifier signature scheme, a signer can designate a verifier (say, Cindy) such that only Cindy can be convinced about the signer's commitment on a message. This is because Cindy is able to construct a signature whose designated verifier is herself, and designated verifier signatures produced by Cindy are indistinguishable from those produced by the signer. In [JSI96], Jakobsson *et al.* briefly discussed a stronger notion called *strong designated verifier signatures* (SDVS). The *strongness* property requires that given a designated

verifier signature and two potential signing public keys, it is *computationally infeasible* for an eavesdropper to determine under which of the two corresponding secret keys the signature was performed. This notion was formally defined by Saeednia, Kremer and Markowitch [SKM03] and further strengthened by Laguillaumie and Vergnaud [LV04a]. In [SZM04], Susilo, Zhang and Mu proposed an identity-based SDVS scheme from pairings. The definition of ring signatures was introduced by Rivest, Shamir and Tauman in [RST01]. They also showed how to achieve a designated verifier signature scheme where two participants in a ring signature collaborate and generate a signature. Designated verifier signature schemes constructed in this way cannot provide the strongness property, as the validity of ring signatures can be verified with the public keys of ring members. Due to the construction in [LV04a], by applying an additional IND-CCA2 public-key encryption, one can convert a designated verifier signature scheme into a strong designated verifier signature scheme. At Crypto'03 rump session [Des03], Desmedt suggested the notion of multi-designated verifiers signatures. Laguillaumie and Vergnaud formally defined this notion in [LV04b], where an efficient generic construction of multi-designated verifiers signatures and an efficient strong designated verifier signature scheme with two designated verifiers were proposed.

In this chapter, we will propose a construction of short strong designated verifier signature scheme. Compared to the existing schemes [JSI96, SKM03, LV04a], our scheme is very efficient in terms of both the signature generation and the signature length. We also extend our scheme to construct a short identity-based strong designated verifier signature scheme. In contrast to the previous construction in [SZM04], our scheme produces a significantly shorter signature length, and requires less computational operations.

Organization

The rest of this chapter is organized as follows. In the next section, we will review the definitions of strong designated verifier signatures. In Section 4.3, we describe the construction of short strong designated verifier signature and its security analysis. A short identity-based strong designated verifier signature scheme is proposed in Section 4.4. We compare the proposed schemes with other schemes in Section 4.5. Section 4.6 concludes this chapter.

4.2 Definitions of Strong Designated Verifier Signatures

This section defines the syntax and the security of strong designated verifier signatures.

Definition 4.1 *A strong designated verifier signature scheme is made up of four algorithms, KeyGen, A-Sign, Verify and B-Sign. For a fixed security parameter $param$, these algorithms work as follows:*

$\text{KeyGen}(param) \rightarrow (sk, pk)$.

This algorithm takes $param$ as input and returns a secret key sk and a public key pk . pk includes the description of the message space \mathcal{M} and the signature space \mathcal{S} . In particular, the signer generates the key pair (sk_A, pk_A) and the verifier generates the key pair (sk_B, pk_B) .

$\text{A-Sign}(param, sk_A, pk_B, m) \rightarrow \sigma$.

This algorithm takes $param$, the signer's secret key sk_A , the verifier's public key pk_B and a message m as inputs, and returns a signature σ for designated verifier with public key pk_B .

$\text{Verify}(param, sk_B, pk_A, m, \sigma) \rightarrow \{\text{valid}, \text{invalid}\}$.

This algorithm takes $param$, the verifier's secret key sk_B , the signer's public key pk_A and a message-signature pair (m, σ) as inputs, and returns **valid** or **invalid**. σ is said to be a valid designated verifier signature on message m if $\text{Verify}(param, sk_B, pk_A, m, \sigma) = \text{valid}$.

$\text{B-Sign}(param, sk_B, pk_A, m) \rightarrow \sigma$.

This algorithm takes $param$, the verifier's secret key sk_B , the signer's public key pk_A and a message m as inputs, and returns a signature σ for designated verifier with public key pk_B .

In addition to the above main algorithms, we also require the following.

- *Correctness.* Any designated verifier signature produced by A-Sign and B-Sign algorithm must be valid. That is,

$\text{Verify}(param, sk_B, pk_A, m, \text{A-Sign}(param, sk_A, pk_B, m)) = \text{valid}$; and

$$\text{Verify}(param, sk_B, pk_A, m, \text{B-Sign}(param, sk_B, pk_A, m)) = \text{valid}.$$

- *Unconditional Ambiguity.* For any message m , designated verifier signatures produced by $\text{A-Sign}(param, sk_A, pk_B, m)$ must have the identical probability distribution as those produced by $\text{B-Sign}(param, sk_B, pk_A, m)$.

Remark: A designated verifier signature scheme but not strong designated verifier signature scheme can be defined in a similar way, with the only difference that algorithm Verify requires $param, m, \sigma, pk_A$, and pk_B as input.

4.2.1 Unforgeability of Strong Designated Verifier Signatures

The forger against strong designated verifier signatures can make signature queries as in the standard notion of security for a signature scheme [GMR88]. As the validity of strong designated verifier signature is not publicly verifiable, the adversary must be allowed to make verification queries as well. The unforgeability of strong designated verifier signatures is defined using the following game between a challenger and an adversary:

Setup: The challenger runs the algorithm $\text{KeyGen}(param)$ to obtain the signer's key pair (sk_A, pk_A) and the verifier's key pair (sk_B, pk_B) . The adversary is given $param$ as well as pk_A and pk_B .

Signature Queries: Proceeding adaptively, the adversary requests signatures on at most q_S messages m_1, \dots, m_{q_S} of its choice. For such a message m_i , the challenger responds with a signature $\sigma_i = \text{A-Sign}(param, sk_A, pk_B, m_i)$.

Verification Queries: Proceeding adaptively, the adversary makes at most q_V verification queries of its choice. For such a pair (m_i, σ_i) , the challenger responds with the output of $\text{Verify}(param, sk_B, pk_A, m_i, \sigma_i)$.

Output: Eventually, the adversary outputs a pair (m^*, σ^*) and wins the game if $m^* \notin \{m_1, \dots, m_{q_S}\}$ and $\text{Verify}(param, sk_B, pk_A, m^*, \sigma^*) = \text{valid}$.

Let $\text{F-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 4.2 A forger \mathcal{A} is said to $(t, q_S, q_V, \varepsilon)$ -break the unforgeability of a strong designated verifier signature scheme if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_S signature queries, q_V verification queries and $\text{F-Adv}_{\mathcal{A}}$ is at least ε . A strong designated verifier signature scheme is $(t, q_S, q_V, \varepsilon)$ -existentially unforgeable under an adaptive chosen message attack if there exists no forger that $(t, q_S, q_V, \varepsilon)$ -breaks it.

In the random oracle model, we add the fifth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

4.2.2 Privacy of Signer's Identity

As in Definition 4.1, unconditional ambiguity is a necessary property for (strong) designated verifier signatures. Even with this property, however, it could be possible to check that there are only two potential signature generators for a given designated verifier signature. One example is designated verifier signature schemes where the verification only requires the public information of the signer and the verifier. If signatures are captured on the line before reaching the verifier, an eavesdropper will believe that the designated verifier did not produce the signature. Therefore, Jakobsson *et al.* [JSI96] suggested a stronger notion “strong designated verifier signature”, which requires the “privacy of signer's identity” (also known as “strongness” of designated verifier signatures). In this thesis, this property is defined as below.

Setup: The challenger runs the algorithm $\text{KeyGen}(param)$ to obtain the potential signers' key pairs $(sk_{A_0}, pk_{A_0}), (sk_{A_1}, pk_{A_1})$ and the verifier's key pair (sk_B, pk_B) . The adversary is given $param, pk_{A_0}, pk_{A_1}$, and pk_B .

Signature Queries: Proceeding adaptively, the adversary can make at most q_S signature queries m_1, \dots, m_{q_S} associated with the signer $A \in \{A_0, A_1\}$. For such a query (m_i, A) , the challenger responds with a signature $\sigma_i = \text{A-Sign}(param, sk_A, pk_B, m_i)$.

Verification Queries: Proceeding adaptively, the adversary makes at most q_V verification queries of its choice. For such a query (m_i, σ_i, A) and $A \in \{A_0, A_1\}$, the challenger responds with the output of $\text{Verify}(param, sk_B, pk_A, m_i, \sigma_i)$.

Output I: The adversary outputs a message m^* with the only restriction that m^* is not one of signature queries if A-Sign and B-Sign are deterministic algorithms.

The challenger responds by picking a random $b \in \{1, 0\}$ and calculating $\sigma^* = \text{A-Sign}(param, sk_{A_b}, pk_B, m^*)$. In either case, σ^* is returned to the adversary.

Queries II: The adversary continues to make signature queries and verification queries with the restriction that (m^*, σ^*) cannot appear as one of verification queries, and m^* is not one of signature queries if A-Sign and B-Sign are deterministic algorithms.

Output II: Eventually, the adversary outputs its guess b' and wins the game if $b' = b$.

Let $\text{D-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 4.3 *A distinguisher \mathcal{A} is said to $(t, q_S, q_V, \varepsilon)$ -break the privacy of signer's identity of a strong designated verifier signature scheme if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_S signature queries, q_V verification queries and $\text{D-Adv}_{\mathcal{A}}$ is at least $1/2 + \varepsilon$. A strong designated verifier signature scheme has $(t, q_S, q_V, \varepsilon)$ -privacy of signer's identity under an adaptive chosen message attack if there exists no distinguisher that $(t, q_S, q_V, \varepsilon)$ -breaks it.*

In the random oracle model, we add the fifth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

4.3 A Short Strong Designated Verifier Signature Scheme

In this section, we describe a concrete construction of strong designated verifier signature with short signature length.

4.3.1 The Description of Our Scheme

Let p be a large prime and q a prime divisor of $p - 1$. Let G be a subgroup in \mathbb{Z}_p^* with prime order q . The generator of G is denoted as g . Let $p, q \geq 2^\ell$, where ℓ is the security parameter. The system wide parameter $param = \{p, q, G, g, \ell, h\}$ where $h : \{0, 1\}^* \times G \rightarrow \mathbb{Z}_q^*$ is a hash function modeled as a random oracle in the security analysis.

- **KeyGen:** The signer randomly selects her secret key $sk_A \in \mathbb{Z}_q^*$ and calculates her public key $pk_A = g^{sk_A} \pmod{p}$. Similarly, the designated verifier selects his secret key $sk_B \in \mathbb{Z}_q^*$ and sets his public key as $pk_B = g^{sk_B} \pmod{p}$. The message space is $\mathcal{M} = \{0, 1\}^*$ and the signature space is $\mathcal{S} = \mathbb{Z}_q^*$.
- **A-Sign:** Given a message $m \in \{0, 1\}^*$, the signer generates the signature $\sigma = h(m, pk_B^{sk_A} \pmod{p})$ with designated verifier with public key pk_B .
- **Verify:** Given a message-signature pair (m, σ) together with the signer's public key pk_A and the verifier's secret key sk_B , this algorithm outputs **valid** if $\sigma = h(m, pk_A^{sk_B} \pmod{p})$. Otherwise, outputs **invalid**.
- **B-Sign:** Given a message $m \in \{0, 1\}^*$ together with the signer's public key pk_A and the verifier's secret key sk_B , one is able to generate a designated verifier signature σ by computing $\sigma = h(m, pk_A^{sk_B} \pmod{p})$.

We show that the proposed scheme satisfies the following properties.

- *Correctness.* The correctness of algorithm **Verify** is due to

$$pk_B^{sk_A} = pk_A^{sk_B} = g^{sk_A sk_B} \pmod{p}.$$

Thus, $\text{Verify}(param, sk_B, pk_A, m, \text{A-Sign}(param, sk_A, pk_B, m)) = \text{Verify}(param, sk_B, pk_A, m, \text{B-Sign}(param, sk_B, pk_A, m)) = \text{valid}$.

- *Unconditional Ambiguity.* It is evident that $\text{A-Sign}(param, sk_A, pk_B, m) = \text{B-Sign}(param, sk_B, pk_A, m)$ as $pk_B^{sk_A} = pk_A^{sk_B} = g^{sk_A sk_B} \pmod{p}$.

4.3.2 Security Analysis: Unforgeability

We now prove that the unforgeability of the proposed scheme is based on the hardness of Gap Diffie-Hellman problem defined in Section 1.4.

Theorem 4.1 *The proposed short strong designated verifier signature scheme in Section 4.3.1 is $(t, q_S, q_V, q_H, \varepsilon)$ -existentially unforgeable under an adaptive chosen message attack, if Gap Diffie-Hellman problem is $(t + c_p(q_H + q_S + q_V + 1), \varepsilon/\epsilon)$ -hard. Here c_p is a constant that depends on \mathbb{Z}_p , and ϵ is the base of natural logarithm.*

Proof. Suppose there is a forger \mathcal{A} can $(t, q_S, q_V, q_H, \varepsilon)$ -break the proposed short designated verifier signature scheme with $param = \{p, q, G, g, \ell, h\}$. We show how

to construct a $t + c_p(q_H + q_S + q_V + 1)$ -time algorithm \mathcal{B} that solves Gap Diffie-Hellman (GDH) problem on G with probability at least ε/ϵ . This will contradict the fact that the GDH problem is $(t + c_p(q_H + q_S + q_V + 1), \varepsilon/\epsilon)$ -hard on G .

Given a random instance (g, g^a, g^b) of the GDH problem together with a Decisional Diffie-Hellman oracle, we will show how \mathcal{B} can use \mathcal{A} to obtain the value of g^{ab} . Algorithm \mathcal{B} simulates the challenger and interacts with forger \mathcal{A} as follows.

Setup: \mathcal{B} selects two random integers $r_A, r_B \in \mathbb{Z}_q^*$ and sets $pk_A = (g^a)^{r_A} \pmod{p}$, $pk_B = (g^b)^{r_B} \pmod{p}$. The adversary \mathcal{A} is given pk_A and pk_B . In our proof, the hash function h will be modeled as random oracle which is simulated by \mathcal{B} as described below.

h Queries: At any time the forger \mathcal{A} can query the random oracle h . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(m_j, k_j, \sigma_j, c_j)$ as explained below. We refer to this list as the h -list. The list is initially empty. When \mathcal{A} makes a request (m_i, k_i) to the oracle h , algorithm \mathcal{B} first submits (g, pk_A, pk_B, k_i) to the Decisional Diffie-Hellman oracle which will inform \mathcal{B} whether (g, pk_A, pk_B, k_i) is a DH-tuple.

1. If (g, pk_A, pk_B, k_i) is not a DH-tuple, \mathcal{B} will search h -list.
 - (a) If (m_i, k_i) already appears on the h -list in a tuple $(m_i, k_i, \sigma_i, c_i)$, algorithm \mathcal{B} responds with $h(m_i, k_i) = \sigma_i$.
 - (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = 0$, which indicates that (g, pk_A, pk_B, k_i) is not a DH-tuple and σ_i thus is an invalid designated verifier signature of message m_i . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h -list and responds with $h(m_i, k_i) = \sigma_i$.
2. Otherwise (g, pk_A, pk_B, k_i) is a DH-tuple, \mathcal{B} will search the h -list.
 - (a) If $(m_i, 1)$ already appears on the h -list in a tuple either with the form $(m_i, k_i, \sigma_i, 1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, 1)$ (as the result of signature queries), algorithm \mathcal{B} updates that tuple with $(m_i, k_i, \sigma_i, 1)$ if necessary and responds with $h(m_i, k_i) = \sigma_i$.
 - (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = 1$, which indicates that (g, pk_A, pk_B, k_i) is a DH-tuple and σ_i is a valid designated verifier signature of message m_i . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h -list and responds with $h(m_i, k_i) = \sigma_i$.

Signature Queries: For a signature query m_i chosen by \mathcal{A} , algorithm \mathcal{B} searches the h -list and responds as follows:

1. If $(m_i, 1)$ already appears on the h -list in a tuple either with form $(m_i, k_i, \sigma_i, 1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, 1)$ (as the result of signature queries), algorithm \mathcal{B} responds with σ_i .
2. Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = 1$, which indicates that σ_i is a valid designated verifier signature of message m_i . \mathcal{B} then adds $(m_i, \top, \sigma_i, c_i)$ on the h -list and responds with σ_i .

Verification Queries: For a verification query (m_i, σ) , \mathcal{B} first searches the h -list.

1. If $(m_i, 1)$ already appears on the h -list in a tuple either with form $(m_i, k_i, \sigma_i, 1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, 1)$ (as the result of signature queries), \mathcal{B} responds with **valid** if $\sigma_i = \sigma$, and **invalid** otherwise.
2. Otherwise, \mathcal{B} responds with **invalid**. This answer could be incorrect when \mathcal{A} makes a signature query of m_i and σ is returned as the answer at some time later. However, this only happens with probability $\frac{1}{q-1}$ as m_i 's valid signature is randomly chosen in \mathbb{Z}_q^* .

Output: Eventually, algorithm \mathcal{A} produces a pair (m^*, σ^*) .

If σ^* is the valid signature of m^* (happens with probability ε), then $(m^*, k^*, \sigma^*, 1)$ must exist on the h -list with probability $1 - \frac{1}{q-1}$ as hash function h is simulated as the random oracle. It follows that (g, pk_A, pk_B, k^*) is a DH tuple. As $pk_A = (g^a)^{r_A}$ and $pk_B = (g^b)^{r_B}$, k^* must be equal to $(g^{ab})^{r_A r_B}$ and $g^{ab} = (k^*)^{(r_A r_B)^{-1} \pmod{q}}$. Algorithm \mathcal{B} outputs $(k^*)^{(r_A r_B)^{-1} \pmod{q}}$ as the answer of the given GDH instance.

\mathcal{B} can output the answer if and only if

1. The simulation is correct, which happens with the probability $(1 - \frac{1}{q-1})^{q_V}$;
2. σ^* is the valid signature of m^* , which happens with probability ε ; and
3. $(m^*, k^*, \sigma^*, 1)$ appears on the h -list, which happens with the probability $1 - \frac{1}{q-1}$.

The overall success probability is

$$\varepsilon \cdot (1 - \frac{1}{q-1})^{q_V+1} \geq \varepsilon/\mathfrak{e}.$$

Here, e is the base of natural logarithm.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to q_H random oracle queries, q_S signature queries, q_V verification queries and calculate g^{ab} from σ^* . We assume each query requires time at most c_p . Hence, the total running time is at most $t + c_p(q_H + q_S + q_V + 1)$. This completes the proof of Theorem 4.1.

4.3.3 Security Analysis: Privacy of Signer's Identity

We now prove that the proposed scheme has the privacy of signer's identity, which is also based on the hardness of Gap Diffie-Hellman problem.

Theorem 4.2 *The proposed short strong designated verifier signature scheme in Section 4.3.1 has $(t, q_S, q_V, q_H, \varepsilon)$ -privacy of signer's identity under an adaptive chosen message attack, if Gap Diffie-Hellman problem is $(t + c_p(q_H + q_S + q_V + 1), \varepsilon/e)$ -hard. Here c_p is a constant that depends on \mathbb{Z}_p , and e is the base of the natural logarithm.*

Proof. Suppose there is a distinguisher \mathcal{A} can $(t, q_S, q_V, q_H, \varepsilon)$ -break the privacy of signer's identity of the proposed short designated verifier signature scheme with $param = \{p, q, G, g, \ell, h\}$. We show how to construct a $t + c_p(q_H + q_S + q_V + 1)$ -time algorithm \mathcal{B} that solves Gap Diffie-Hellman (GDH) problem on G with probability at least ε/e . This will contradict the fact that the GDH problem is $(t + c_p(q_H + q_S + q_V + 1), \varepsilon/e)$ -hard on G .

Given a random instance (g, g^a, g^b) of GDH problem together with a Decisional Diffie-Hellman oracle, we will show how \mathcal{B} can use \mathcal{A} to obtain g^{ab} . Algorithm \mathcal{B} simulates the challenger and interacts with forger \mathcal{A} as follows.

Setup: \mathcal{B} selects three random integers $r_0, r_1, r_2 \in \mathbb{Z}_q^*$ and sets $pk_{A_0} = (g^a)^{r_0} \pmod{p}$, $pk_{A_1} = (g^a)^{r_1} \pmod{p}$, $pk_B = (g^b)^{r_2} \pmod{p}$. The adversary \mathcal{A} is given pk_{A_0}, pk_{A_1} and pk_B . In the proof, the hash function h will be modeled as random oracle which is simulated by \mathcal{B} as described below.

h Queries: At any time the forger \mathcal{A} can query the random oracle h . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(m_j, k_j, \sigma_j, c_j)$ as explained below. We refer to this list as the h -list. The list is initially empty. When \mathcal{A} makes a request (m_i, k_i) to the oracle h , algorithm \mathcal{B} first

checks if (g, pk_{A_0}, pk_B, k_i) or (g, pk_{A_1}, pk_B, k_i) is a DH-tuple with the help of the Decisional Diffie-Hellman oracle.

1. If neither of them is a DH-tuple, \mathcal{B} will search the h -list. If (m_i, k_i) already appears on the h -list in a tuple $(m_i, k_i, \sigma_i, c_i)$, algorithm \mathcal{B} responds with $h(m_i, k_i) = \sigma_i$. Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = 0$, which indicates that σ_i is an invalid designated verifier signature of message m_i either under (pk_{A_0}, pk_B) or (pk_{A_1}, pk_B) . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h -list and responds with $h(m_i, k_i) = \sigma_i$.
2. Else, if (g, pk_{A_0}, pk_B, k_i) is a DH-tuple, \mathcal{B} will search the h -list.
 - (a) If (m_i, A_0) already appears on the h -list in a tuple either with form $(m_i, k_i, \sigma_i, A_0)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0)$ (as the result of signature queries), algorithm \mathcal{B} updates that tuple with $(m_i, k_i, \sigma_i, A_0)$ if necessary and responds with $h(m_i, k_i) = \sigma_i$.
 - (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = A_0$, which indicates that σ_i is a valid designated verifier signature of message m_i between public keys pk_{A_0} and pk_B . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h -list and responds with $h(m_i, k_i) = \sigma_i$.
3. Otherwise, (g, pk_{A_1}, pk_B, k_i) is a DH-tuple, \mathcal{B} will search the h -list.
 - (a) If (m_i, A_1) already appears on the h -list in a tuple either with form $(m_i, k_i, \sigma_i, A_1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_1)$ (as the result of signature queries), algorithm \mathcal{B} updates that tuple with $(m_i, k_i, \sigma_i, A_1)$ if necessary and responds with $h(m_i, k_i) = \sigma_i$.
 - (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = A_1$, which indicates that σ_i is a valid designated verifier signature of message m_i between public keys pk_{A_1} and pk_B . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h -list and responds with $h(m_i, k_i) = \sigma_i$.

Signature Queries: For a signature query with form (m_i, A_0) , algorithm \mathcal{B} searches the h -list and responds as follows:

1. If (m_i, A_0) already appears on the h -list in a tuple either with form $(m_i, k_i, \sigma_i, A_0)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0)$ (as the result of signature queries), algorithm \mathcal{B} responds with σ_i .

2. Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = A_0$, which indicates that σ_i is a valid designated verifier signature of message m_i between pk_{A_0} and pk_B . \mathcal{B} then adds $(m_i, \top, \sigma_i, c_i)$ on the h -list and responds with σ_i .

For a signature query with form (m_i, A_1) , algorithm \mathcal{B} searches the h -list and responds as follows:

1. If (m_i, A_1) already appears on the h -list in a tuple either with form $(m_i, k_i, \sigma_i, A_1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_1)$ (as the result of signature queries), algorithm \mathcal{B} responds with σ_i .
2. Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_q^*$ and sets $c_i = A_1$, which indicates that σ_i is a valid designated verifier signature of message m_i between pk_{A_1} and pk_B . \mathcal{B} then adds $(m_i, \top, \sigma_i, c_i)$ on the h -list and responds with σ_i .

Verification Queries: For a verification query with form (m_i, σ, A_0) , \mathcal{B} first searches the h -list.

1. If (m_i, A_0) already appears on the h -list in a tuple either with form $(m_i, k_i, \sigma_i, A_0)$ or $(m_i, \top, \sigma_i, A_0)$, \mathcal{B} responds with **valid** if $\sigma_i = \sigma$, and **invalid** otherwise.
2. Otherwise, \mathcal{B} responds with **invalid**. The answer could be incorrect when \mathcal{A} makes a signature query of (m_i, A_0) and σ is returned as the answer at some time later. However, this only happens with probability $\frac{1}{q-1}$ as m_i 's valid signature is randomly chosen in \mathbb{Z}_q^* .

For a verification query with form (m_i, σ, A_1) , \mathcal{B} first searches the h -list.

1. If (m_i, A_1) already appears on the h -list in a tuple with form $(m_i, k_i, \sigma_i, A_1)$ or $(m_i, \top, \sigma_i, A_1)$, \mathcal{B} responds with **valid** if $\sigma_i = \sigma$, and **invalid** otherwise.
2. Otherwise, \mathcal{B} responds with **invalid**. The answer could be incorrect when \mathcal{A} makes a signature query of (m_i, A_1) and σ is returned as the answer at some time later. However, this only happens with probability $\frac{1}{q-1}$ as m_i 's valid signature is randomly chosen in \mathbb{Z}_q^* .

Output I: \mathcal{A} outputs the challenge message m^* , which cannot be one of signature queries as **A-Sign** and **B-Sign** are deterministic algorithms in our scheme. In response, \mathcal{B} first picks a random $b \in \{1, 0\}$ and makes a signature query (m^*, A_b) by itself. Let the result be σ^* , which is returned to \mathcal{A} .

Queries II: \mathcal{A} continues to make signature queries and verification queries with restrictions that (m^*, σ^*) cannot appear as one of verification queries and m^* is not one of signature queries as **A-Sign** and **B-Sign** are deterministic algorithms in our scheme. \mathcal{B} responds these queries as same as described above.

Output II: Eventually algorithm \mathcal{A} outputs its guess b' .

If $b' = b$ happens with probability greater than $1/2$, then either $(m^*, k_0^*, \sigma^*, A_0)$ or $(m^*, k_1^*, \sigma^*, A_1)$ must exist on the h -list with probability $1 - \frac{1}{q-1}$, as hash function h is simulated as the random oracle. If $(m^*, k_0^*, \sigma^*, A_0)$ appears on the h -list, then $(g, pk_{A_0}, pk_B, k_0^*)$ is a DH tuple. As $pk_{A_0} = (g^a)^{r_0}$ and $pk_B = (g^b)^{r_2}$, k_0^* must be equal to $(g^{ab})^{r_0 r_2}$ and $g^{ab} = (k_0^*)^{(r_0 r_2)^{-1} \pmod{q}}$. Algorithm \mathcal{B} outputs $(k_0^*)^{(r_0 r_2)^{-1} \pmod{q}}$ as the answer of the given GDH instance. For the case $(m^*, k_1^*, \sigma^*, A_1)$ appears on the h -list, \mathcal{B} can output $(k_1^*)^{(r_1 r_2)^{-1} \pmod{q}}$ as the answer of the given GDH instance as well.

\mathcal{B} can output the answer if and only if

1. The simulation is correct, which happens with the probability $(1 - \frac{1}{q-1})^{q_V}$;
2. \mathcal{A} outputs a correct guess $b' = b$, which happens with probability $1/2 + \varepsilon$; and
3. Either $(m^*, k_0^*, \sigma^*, A_0)$ or $(m^*, k_1^*, \sigma^*, A_1)$ appears on the h -list, which happens with the probability $1 - \frac{1}{q-1}$.

The overall success probability is

$$\varepsilon(1 - \frac{1}{q-1})^{q_V+1} \geq \varepsilon/\mathfrak{e}.$$

Here, \mathfrak{e} is the base of natural logarithm.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to q_H random oracle queries, q_S signature queries, q_V verification queries and calculate g^{ab} from σ^* . We assume each query requires time at most c_p . Hence, the total running time is at most $t + c_p(q_H + q_S + q_V + 1)$. This completes the proof of Theorem 4.2.

4.4 Short Identity-based Strong Designated Verifier Signatures

In this section, we describe a concrete construction of short identity-based strong designated verifier signature where a user's public key is his/her identity. We first review the definitions of identity-based strong designated verifier signature as below.

Definition 4.4 *An identity-based strong designated verifier signature scheme is made up of five algorithms, Setup, KeyGen, A-Sign, Verify and B-Sign. For a fixed security parameter $param$, these algorithms work as follows:*

Setup($param$) $\rightarrow (msk, mpk)$.

This algorithm takes $param$ as input and returns master secret key msk and master public key mpk .

Key-Extract($param, msk, ID$) $\rightarrow sk$.

This algorithm takes $param$, the master secret key msk and an entity's identity ID as inputs, and returns the secret key sk of ID . Private Key Generator (PKG) runs this algorithm to generate the secret key sk_A for the signer with identity ID_A , and the secret key sk_B for the verifier with identity ID_B .

A-Sign($param, sk_A, ID_B, m$) $\rightarrow \sigma$.

This algorithm takes $param$, the signer's secret key sk_A , the verifier's identity ID_B and a message m as inputs, and returns a signature σ for designated verifier with identity ID_B .

Verify($param, sk_B, ID_A, m, \sigma$) $\rightarrow \{\text{valid}, \text{invalid}\}$.

This algorithm takes $param$, the verifier's secret key sk_B , the signer's identity ID_A and a message-signature pair (m, σ) as inputs, and returns **valid** or **invalid**.

B-Sign($param, sk_B, ID_A, m$) $\rightarrow \sigma$.

This algorithm takes $param$, the verifier's secret key sk_B , the signer's identity ID_A and a message m as inputs, and returns a signature σ for designated verifier with identity ID_B .

In addition to the above main algorithms, we also require the following.

- *Correctness.* Any designated verifier signature produced by A-Sign and B-Sign is valid. That is,

$$\text{Verify}(param, sk_B, ID_A, m, \text{A-Sign}(param, sk_A, ID_B, m)) = \text{valid}; \text{ and}$$

$$\text{Verify}(param, sk_B, ID_A, m, \text{B-Sign}(param, sk_B, ID_A, m)) = \text{valid}.$$

- *Unconditional Ambiguity.* For a message m , designated verifier signatures produced by $\text{A-Sign}(param, sk_A, ID_B, m)$ must have the identical probability distribution as those produced by $\text{B-Sign}(param, sk_B, ID_A, m)$.

4.4.1 Unforgeability of Identity-based Strong Designated Verifier Signatures

The existential unforgeability of identity-based strong designated verifier signatures is quite similar to that defined in Section 4.2.1. It is defined by the game as below.

Setup: The challenger first generates the system parameter $param$ and the master secret/public key pair (msk, mpk) . The adversary is given $param$ and mpk .

Extract Queries: Proceeding adaptively, the adversary requests secret keys of at most q_E identities $\{ID_1, \dots, ID_{q_E}\}$ of its choice. For such an identity ID_i , the challenger responds with $sk_i = \text{Key-Extract}(param, msk, ID_i)$.

Signature Queries: Proceeding adaptively, the adversary makes at most q_S signature requests with the form (m_i, ID_S, ID_V) ($ID_S \neq ID_V$) of its choice. For such a query (m_i, ID_S, ID_V) , the challenger responds with a signature $\sigma_i = \text{A-Sign}(param, sk_s, ID_V, m_i)$.

Verification Queries: Proceeding adaptively, the adversary makes at most q_V verification queries of its choice. For such a query $(m_i, \sigma_i, ID_S, ID_V)$ ($ID_S \neq ID_V$), the challenger responds with the output of $\text{Verify}(param, sk_V, ID_S, m_i, \sigma_i)$.

Output: Eventually, the adversary outputs $(m^*, \sigma^*, ID_A^*, ID_B^*)$ ($ID_A^* \neq ID_B^*$) and wins the game if

1. Neither (m^*, ID_A^*, ID_B^*) nor (m^*, ID_B^*, ID_A^*) is one of signature queries;
2. Neither ID_A^* nor ID_B^* appears as one of extract queries;
3. $\text{Verify}(param, sk_B^*, ID_A^*, m^*, \sigma^*) = \text{valid}$.

Let $\text{F-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 4.5 *A forger \mathcal{A} is said to $(t, q_E, q_S, q_V, \varepsilon)$ -break the unforgeability of an identity-based strong designated verifier signature scheme if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_E extract queries, q_S signature queries, q_V verification queries and $\text{F-Adv}_{\mathcal{A}}$ is at least ε . An identity-based strong designated verifier signature scheme is $(t, q_E, q_S, q_V, \varepsilon)$ -existentially unforgeable under an adaptive chosen identity and chosen message attack if there exists no forger that $(t, q_E, q_S, q_V, \varepsilon)$ -breaks it.*

In the random oracle model, we add the sixth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

4.4.2 Privacy of Signer's Identity in Identity-based Strong Designated Verifier Signatures

The privacy of signer's identity in identity-based strong designated verifier signature scheme is defined as follows:

Setup: The challenger first generates the system parameter $param$ and the master secret/public key pair (msk, mpk) . The adversary is given $param$ and mpk .

Extract Queries: Proceeding adaptively, the adversary requests secret keys of at most q_E identities $\{ID_1, \dots, ID_{q_E}\}$ of its choice. For such an identity ID_i , the challenger responds with $sk_i = \text{Key-Extract}(param, msk, ID_i)$.

Signature Queries: Proceeding adaptively, the adversary can make at most q_S signature requests with the form (m_i, ID_S, ID_V) ($ID_S \neq ID_V$) of its choice. For such a query (m_i, ID_S, ID_V) , the challenger responds with a signature $\sigma_i = \text{A-Sign}(param, sk_S, ID_V, m_i)$.

Verification Queries: Proceeding adaptively, the adversary makes at most q_V verification queries of its choice. For such a query $(m_i, \sigma_i, ID_S, ID_V)$ ($ID_S \neq ID_V$), the challenger responds with the output of $\text{Verify}(param, sk_V, ID_S, m_i, \sigma_i)$.

Output I: The adversary outputs $(m^*, ID_{A_0}^*, ID_{A_1}^*, ID_B^*)$ ($ID_{A_0}^*, ID_{A_1}^*, ID_B^*$ are all different) with restrictions that

1. None of $\{ID_{A_0}^*, ID_{A_1}^*, ID_B^*\}$ appears as one of extract queries; and

2. None of $\{(m^*, ID_{A_0}^*, ID_B^*), (m^*, ID_B^*, ID_{A_0}^*), (m^*, ID_{A_1}^*, ID_B^*), (m^*, ID_B^*, ID_{A_1}^*)\}$ appears as one of signature queries if A-Sign and B-Sign are deterministic algorithms.

The challenger responds by picking a random $b \in \{1, 0\}$ and computing $\sigma^* = \text{A-Sign}(param, sk_{A_b}^*, ID_B^*, m^*)$. In either case, σ^* is returned to the adversary.

Queries II: The adversary continues to make extract queries, signature queries and verification queries with restrictions as follows:

1. None of $\{ID_{A_0}^*, ID_{A_1}^*, ID_B^*\}$ appears as one of extract queries; and
2. None of $\{(m^*, ID_{A_0}^*, ID_B^*), (m^*, ID_B^*, ID_{A_0}^*), (m^*, ID_{A_1}^*, ID_B^*), (m^*, ID_B^*, ID_{A_1}^*)\}$ appears as one of signature queries if A-Sign and B-Sign are deterministic algorithms.
3. None of $\{(m^*, \sigma^*, ID_{A_0}^*, ID_B^*), (m^*, \sigma^*, ID_B^*, ID_{A_0}^*), (m^*, \sigma^*, ID_{A_1}^*, ID_B^*), (m^*, \sigma^*, ID_B^*, ID_{A_1}^*)\}$ appears as one of verification queries.

Output II: Eventually, the adversary outputs its guess b' and wins the game if $b' = b$.

Let $\text{D-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 4.6 *A distinguisher \mathcal{A} is said to $(t, q_E, q_S, q_V, \varepsilon)$ -break the privacy of signer's identity of an identity-based strong designated verifier signature scheme if \mathcal{A} runs in time at most t , \mathcal{A} makes at most q_E extract queries, q_S signature queries, q_V verification queries and $\text{D-Adv}_{\mathcal{A}}$ is at least $1/2 + \varepsilon$. An identity-based strong designated verifier signature scheme has $(t, q_E, q_S, q_V, \varepsilon)$ -privacy of signer's identity under an adaptive chosen identity and chosen message attack if there exists no distinguisher that $(t, q_E, q_S, q_V, \varepsilon)$ -breaks it.*

In the random oracle model, we add the sixth parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle.

4.4.3 The Proposed Short Identity-based Strong Designated Verifier Signature Scheme

Let e be the pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ described in Section 1.5. The generator of \mathbb{G}_1 is denoted as g_1 . The order of \mathbb{G}_1 is a prime p . Let h_{ID} be a hash function

$h_{\text{ID}} : \{0, 1\}^* \rightarrow \mathbb{G}_1$. Let h_p be a hash function $h_p : \{0, 1\}^* \times \mathbb{G}_T \rightarrow \mathbb{Z}_p^*$. Both of them are modeled as random oracles in the security analysis. The system parameter is $param = \{\mathbb{G}_1, \mathbb{G}_T, p, e, g_1, h_{\text{ID}}, h_p\}$.

Setup. The PKG in the identity-based system picks a random element $msk \in \mathbb{Z}_p^*$ and calculates the master public key $mpk = g_1^{msk}$.

Key-Extract. Given an identity ID , PKG calculates the secret key as $h_{\text{ID}}(ID)^{msk}$. In particular, for the signer's identity ID_A , PKG calculates the secret key as $sk_A = h_{\text{ID}}(ID_A)^{msk}$. For the verifier's identity ID_B , PKG calculates the secret key as $sk_B = h_{\text{ID}}(ID_B)^{msk}$.

A-Sign. Given a message $m \in \{0, 1\}^*$, the signer generates the signature $\sigma = h_p(m, e(sk_A, h_{\text{ID}}(ID_B)))$ for designated verifier with identity ID_B .

Verify. Given a message-signature pair (m, σ) together with the signer's identity ID_A and the verifier's secret key sk_B , this algorithm outputs **valid** if $\sigma = h_p(m, e(sk_B, h_{\text{ID}}(ID_A)))$. Otherwise, outputs **invalid**.

B-Sign. Given a message $m \in \{0, 1\}^*$ together with the signer's identity ID_A and the verifier's secret key sk_B , one can generate a designated verifier signature by computing $\sigma = h_p(m, e(sk_B, h_{\text{ID}}(ID_A)))$.

We show that the proposed scheme satisfies the following properties.

- *Correctness.* The correctness of algorithm **Verify** is due to the symmetry of bilinear mapping $e(\cdot, \cdot)$. That is, $e(sk_A, h_{\text{ID}}(ID_B)) = e(h_{\text{ID}}(ID_A)^{msk}, h_{\text{ID}}(ID_B)) = e(h_{\text{ID}}(ID_B)^{msk}, h_{\text{ID}}(ID_A)) = e(sk_B, h_{\text{ID}}(ID_A))$.
- *Unconditional Ambiguity.* It is evident that $\text{A-Sign}(param, sk_A, ID_B, m) = \text{B-Sign}(param, sk_B, ID_A, m)$ as $e(sk_A, h_{\text{ID}}(ID_B)) = e(sk_B, h_{\text{ID}}(ID_A))$.

4.4.4 Security Analysis: Unforgeability

We now prove that the unforgeability of the proposed short identity-based strong designated verifier signature scheme is based on the hardness of Gap Bilinear Diffie-Hellman problem defined in Section 1.5.

Theorem 4.3 *The proposed short identity-based strong designated verifier signature scheme in Section 4.4.3 is $(t, q_E, q_S, q_V, q_H, \varepsilon)$ -existentially unforgeable under an*

adaptive chosen identity and chosen message attack, if Gap Bilinear Diffie-Hellman problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1), \frac{2\varepsilon}{\epsilon^3 q_H^2})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$. Here $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ is a constant on $(\mathbb{G}_1, \mathbb{G}_T)$, and ϵ is the base of natural logarithm.

Proof. Suppose there is a forger \mathcal{A} can $(t, q_E, q_S, q_V, q_H, \varepsilon)$ -break the proposed short identity-based designated verifier signature scheme with $param = \{\mathbb{G}_1, \mathbb{G}_T, g_1, p, e, h_{ID}, h_p\}$. We show how to construct a $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1)$ -time algorithm \mathcal{B} that solves Gap Bilinear Diffie-Hellman (GBDH) on $(\mathbb{G}_1, \mathbb{G}_T)$ with probability at least $\frac{2\varepsilon}{\epsilon^3 q_H^2}$. This will contradict the fact that the GBDH problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1), \frac{2\varepsilon}{\epsilon^3 q_H^2})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$.

Given a random instance $(g_1, g_1^a, g_1^b, g_1^c)$ of the GBDH problem together with a Decisional Bilinear Diffie-Hellman oracle, we will show how \mathcal{B} can use \mathcal{A} to obtain the value of $e(g_1, g_1)^{abc}$. Algorithm \mathcal{B} simulates the challenger and interacts with forger \mathcal{A} as follows.

Setup: \mathcal{B} selects a random integer $r_c \in \mathbb{Z}_p^*$ and calculates $mpk = (g_1^c)^{r_c}$. The corresponding $msk = cr_c$, which is not known to algorithm \mathcal{B} . The adversary \mathcal{A} is given $param = \{\mathbb{G}_1, \mathbb{G}_T, p, e, g_1, h_{ID}, h_p\}$ and mpk . In our proof, hash functions h_{ID}, h_p will be modeled as random oracles which are simulated by \mathcal{B} as described below.

h_{ID} Queries: To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(ID_i, r_i, h_{ID}(ID_i))$ as explained below. We refer to this list as the h_{ID} -list. The list is initially empty. We say a query ID_i is fresh if it has not appeared before. If a query ID_i is not fresh, \mathcal{B} searches the h_{ID} -list and responds with $h_{ID}(ID_i)$. We assume there are at most $q_{h_{ID}}$ fresh queries. \mathcal{B} first guesses two numbers $A, B \in \{1, \dots, q_{h_{ID}}\}$.

1. For the A^{th} fresh query ID_A , \mathcal{B} picks a random integer $r_A \in \mathbb{Z}_p^*$ and calculates $h_{ID}(ID_A) = (g_1^a)^{r_A}$. \mathcal{B} then adds $(ID_A, r_A, h_{ID}(ID_A))$ on the h_{ID} -list and responds with $h_{ID}(ID_A)$.
2. For the B^{th} fresh query ID_B , \mathcal{B} picks a random integer $r_B \in \mathbb{Z}_p^*$ and calculates $h_{ID}(ID_B) = (g_1^b)^{r_B}$. \mathcal{B} then adds $(ID_B, r_B, h_{ID}(ID_B))$ on the h_{ID} -list and responds with $h_{ID}(ID_B)$.
3. Otherwise, for a fresh query ID_i , \mathcal{B} picks a random integer $r_i \in \mathbb{Z}_p^*$ and calculates $h_{ID}(ID_i) = g_1^{r_i}$. \mathcal{B} then adds $(ID_i, r_i, h_{ID}(ID_i))$ on the h_{ID} -list and responds with $h_{ID}(ID_i)$.

Extract Queries: For an extract query ID_i , algorithm \mathcal{B} first searches the h_{ID} -list.

If ID_i does not appear on the h_{ID} -list, \mathcal{B} makes an h_{ID} query by itself and thus there is a tuple $(ID_i, r_i, h_{\text{ID}}(ID_i))$ on the h_{ID} -list. If $ID_i \in \{ID_A, ID_B\}$, \mathcal{B} aborts and the simulation fails. Otherwise, \mathcal{B} responds with $mpk^{r_i} = (g_1^e)^{r_i} = h_{\text{ID}}(ID_i)^{cr_i} = h_{\text{ID}}(ID_i)^{msk}$.

h_p queries. At any time the forger \mathcal{A} can query the random oracle h_p . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(m_j, k_j, \sigma_j, c_j)$ as explained below. We refer to this list as the h_p -list. The list is initially empty. When \mathcal{A} makes a request (m_i, k_i) to the oracle h_p , algorithm \mathcal{B} first submits $(g_1, mpk, h_{\text{ID}}(ID_A), h_{\text{ID}}(ID_B), k_i)$ to the Decisional Bilinear Diffie-Hellman oracle which will inform \mathcal{B} whether $(g_1, mpk, h_{\text{ID}}(ID_A), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple.

1. If $(g_1, mpk, h_{\text{ID}}(ID_A), h_{\text{ID}}(ID_B), k_i)$ is not a BDH-tuple, \mathcal{B} will search the h_p -list.
 - (a) If (m_i, k_i) already appears on the h_p -list in a tuple $(m_i, k_i, \sigma_i, c_i)$, algorithm \mathcal{B} responds with $h_p(m_i, k_i) = \sigma_i$.
 - (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = 0$, which indicates that $(g_1, mpk, h_{\text{ID}}(ID_A), h_{\text{ID}}(ID_B), k_i)$ is not a BDH-tuple and σ_i thus is an invalid designated verifier signature of message m_i between ID_A and ID_B . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h_p -list and responds with $h_p(m_i, k_i) = \sigma_i$.
2. Otherwise $(g_1, mpk, h_{\text{ID}}(ID_A), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple, \mathcal{B} will search the h_p -list.
 - (a) If $(m_i, 1)$ already appears on the h_p -list in a tuple either with the form $(m_i, k_i, \sigma_i, 1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, 1)$ (as the result of signature queries), algorithm \mathcal{B} updates that tuple with $(m_i, k_i, \sigma_i, 1)$ if necessary and responds with $h_p(m_i, k_i) = \sigma_i$.
 - (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = 1$, which indicates that $(g_1, mpk, h_{\text{ID}}(ID_A), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple and σ_i is a valid designated verifier signature of message m_i between ID_A and ID_B . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h_p -list and responds with $h_p(m_i, k_i) = \sigma_i$.

Signature Queries: For a signature query (m_i, ID_S, ID_V) chosen by \mathcal{A} , if $(ID_S, ID_V) \notin \{(ID_A, ID_B), (ID_B, ID_A)\}$, \mathcal{B} can calculate either sk_{ID_S} or sk_{ID_V} and thus is able to calculate $k_{SV} = e(sk_{ID_S}, h_{ID}(ID_V))$. \mathcal{B} then makes an h_p request (m_i, k_{SV}) and returns $h_p(m_i, k_{SV})$ to \mathcal{A} . For the case $(ID_S, ID_V) \in \{(ID_A, ID_B), (ID_B, ID_A)\}$, algorithm \mathcal{B} searches the h_p -list and responds as follows:

1. If $(m_i, 1)$ already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, 1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, 1)$ (as the result of signature queries), algorithm \mathcal{B} responds with σ_i .
2. Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = 1$, which indicates that σ_i is a valid designated verifier signature of message m_i between ID_A and ID_B . \mathcal{B} then adds $(m_i, \top, \sigma_i, c_i)$ on the h_p -list and responds with σ_i .

Verification Queries: For a verification query $(m_i, \sigma, ID_S, ID_V)$, if $(ID_S, ID_V) \notin \{(ID_A, ID_B), (ID_B, ID_A)\}$, \mathcal{B} can calculate either sk_{ID_S} or sk_{ID_V} and thus is able to calculate $k_{SV} = e(sk_{ID_S}, h_{ID}(ID_V))$. \mathcal{B} then makes an h_p request (m_i, k_{SV}) and obtains $h_p(m_i, k_{SV})$. \mathcal{B} responds with **valid** if $h_p(m_i, k_{SV}) = \sigma$, and **invalid** otherwise. For the case $(ID_S, ID_V) \in \{(ID_A, ID_B), (ID_B, ID_A)\}$, \mathcal{B} first searches the h_p -list.

1. If $(m_i, 1)$ already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, 1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, 1)$ (as the result of signature queries), \mathcal{B} responds with **valid** if $\sigma_i = \sigma$, and **invalid** otherwise.
2. Otherwise, \mathcal{B} responds with **invalid**. The answer could be incorrect when \mathcal{A} makes a signature query of (m_i, ID_S, ID_V) and σ is returned as the answer at some time later. However, this only happens with probability $\frac{1}{p-1}$ as m_i 's valid signature is randomly chosen in \mathbb{Z}_p^* .

Output: Eventually algorithm \mathcal{A} produces a pair $(m^*, \sigma^*, ID_A^*, ID_B^*)$.

If σ^* is the valid signature of m^* and $(ID_A^*, ID_B^*) \in \{(ID_A, ID_B), (ID_B, ID_A)\}$ (happens with probability at least $\frac{2\varepsilon}{q_{h_{ID}}(q_{h_{ID}}-1)}$), then $(m^*, k^*, \sigma^*, 1)$ must exist on the h_p -list with probability $1 - \frac{1}{p-1}$ as hash function h_p is simulated as a random oracle.

It follows that $(g_1, mpk, h_{\text{ID}}(ID_A), h_{\text{ID}}(ID_B), k^*)$ is a BDH tuple. As $h_{\text{ID}}(ID_A) = (g_1^a)^{r_A}$, $h_{\text{ID}}(ID_B) = (g_1^b)^{r_B}$ and $mpk = (g_1^c)^{r_C}$, k^* must be equal to $(e(g_1, g_1)^{\text{abc}})^{r_A r_B r_C}$ and $e(g_1, g_1)^{\text{abc}} = (k^*)^{(r_A r_B r_C)^{-1} \pmod{p}}$. Algorithm \mathcal{B} outputs $(k^*)^{(r_A r_B r_C)^{-1} \pmod{p}}$ as the answer of the given GBDH instance.

\mathcal{B} can output the correct answer if and only if

1. The simulation is correct, which happens with probability $(1 - \frac{2}{q_{\text{ID}}})^{q_E} \cdot (1 - \frac{1}{p-1})^{q_V}$;
2. σ^* is the valid signature of m^* between ID_A and ID_B , which happens with probability $\frac{2\varepsilon}{q_{\text{ID}}(q_{\text{ID}}-1)}$; and
3. $(m^*, k^*, \sigma^*, 1)$ appears on the h_p -list, which happens with probability $1 - \frac{1}{p-1}$.

The overall success probability \mathcal{B} can output the correct answer is

$$(1 - \frac{2}{q_{\text{ID}}})^{q_E} \cdot (1 - \frac{1}{p-1})^{q_V+1} \cdot \frac{2\varepsilon}{q_{\text{ID}}(q_{\text{ID}}-1)} \geq \frac{2\varepsilon}{\mathfrak{e}^3 q_H^2}.$$

Here, \mathfrak{e} is the base of natural logarithm.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to q_H random oracle queries, q_E extract queries, q_S signature queries, q_V verification queries and compute $e(g_1, g_1)^{\text{abc}}$ from σ^* . We assume each query requires time at most $c_{(\mathbb{G}_1, \mathbb{G}_T)}$. Hence, the total running time is at most $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1)$. This completes the proof of Theorem 4.3.

4.4.5 Security Analysis: Privacy of Signer's Identity

We now prove that the privacy of signer's identity of the proposed short identity-based strong designated verifier signature scheme is based on the hardness of Gap Bilinear Diffie-Hellman problem.

Theorem 4.4 *The proposed short identity-based strong designated verifier signature scheme in Section 4.4.3 has $(t, q_E, q_S, q_V, q_H, \varepsilon)$ -privacy of signer's identity under an adaptive chosen identity and chosen message attack, if Gap Bilinear Diffie-Hellman problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1), \frac{2\varepsilon}{\mathfrak{e}^4 q_H^3})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$. Here $c_{(\mathbb{G}_1, \mathbb{G}_T)}$ is a constant on $(\mathbb{G}_1, \mathbb{G}_T)$, and \mathfrak{e} is the base of natural logarithm.*

Proof. Suppose there is an adversary \mathcal{A} can $(t, q_E, q_S, q_V, q_H, \varepsilon)$ -break the privacy of signer's identity of the proposed short identity-based strong designated verifier

signature scheme with $param = \{\mathbb{G}_1, \mathbb{G}_T, g_1, p, e, h_{\text{ID}}, h_p\}$. We show how to construct a $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1)$ -time algorithm \mathcal{B} that solves Gap Bilinear Diffie-Hellman (GBDH) on $(\mathbb{G}_1, \mathbb{G}_T)$ with probability at least $\frac{2\varepsilon}{e^4 q_H^3}$. This will contradict the fact that the GBDH problem is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1), \frac{2\varepsilon}{e^4 q_H^3})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$.

Given a random instance $(g_1, g_1^a, g_1^b, g_1^c)$ of the GBDH problem together with a Decisional Bilinear Diffie-Hellman oracle, we will show how \mathcal{B} can use \mathcal{A} to obtain the value of $e(g_1, g_1)^{abc}$. Algorithm \mathcal{B} simulates the challenger and interacts with distinguisher \mathcal{A} as follows.

Setup: \mathcal{B} selects a random integer $r_c \in \mathbb{Z}_p^*$ and calculates $mpk = (g_1^c)^{r_c}$. The corresponding $msk = cr_c$, which is not known to algorithm \mathcal{B} . The adversary \mathcal{A} is given $param = \{\mathbb{G}_1, \mathbb{G}_T, p, e, g_1, h_{\text{ID}}, h_p\}$ and mpk . In our proof, hash functions h_{ID}, h_p will be modeled as random oracles which are simulated by \mathcal{B} as described below.

h_{ID} Queries: To respond to these queries, algorithm \mathcal{B} maintains a list of tuples $(ID_i, r_i, h_{\text{ID}}(ID_i))$ as explained below. We refer to this list as the h_{ID} -list. The list is initially empty. We say a query ID_i is fresh if it has not appeared before. If a query ID_i is not fresh, \mathcal{B} searches the h_{ID} -list and responds with $h_{\text{ID}}(ID_i)$. We assume there are at most $q_{h_{\text{ID}}}$ fresh queries. \mathcal{B} first guesses three numbers $A_0, A_1, B \in \{1, \dots, q_{h_{\text{ID}}}\}$.

1. For the A_0^{th} fresh query ID_{A_0} , \mathcal{B} picks a random integer $r_{A_0} \in \mathbb{Z}_p^*$ and calculates $h_{\text{ID}}(ID_{A_0}) = (g_1^a)^{r_{A_0}}$. \mathcal{B} then adds $(ID_{A_0}, r_{A_0}, h_{\text{ID}}(ID_{A_0}))$ on h_{ID} -list and responds with $h_{\text{ID}}(ID_{A_0})$.
2. For the A_1^{th} fresh query ID_{A_1} , \mathcal{B} picks a random integer $r_{A_1} \in \mathbb{Z}_p^*$ and calculates $h_{\text{ID}}(ID_{A_1}) = (g_1^a)^{r_{A_1}}$. \mathcal{B} then adds $(ID_{A_1}, r_{A_1}, h_{\text{ID}}(ID_{A_1}))$ on the h_{ID} -list and responds with $h_{\text{ID}}(ID_{A_1})$.
3. For the B^{th} fresh query ID_B , \mathcal{B} picks a random integer $r_B \in \mathbb{Z}_p^*$ and calculates $h_{\text{ID}}(ID_B) = (g_1^b)^{r_B}$. \mathcal{B} then adds $(ID_B, r_B, h_{\text{ID}}(ID_B))$ on the h_{ID} -list and responds with $h_{\text{ID}}(ID_B)$.
4. Otherwise, for a fresh query ID_i , \mathcal{B} picks a random integer $r_i \in \mathbb{Z}_p^*$ and calculates $h_{\text{ID}}(ID_i) = g_1^{r_i}$. \mathcal{B} then adds $(ID_i, r_i, h_{\text{ID}}(ID_i))$ on the h_{ID} -list and responds with $h_{\text{ID}}(ID_i)$.

Extract Queries: For an extract query ID_i , algorithm \mathcal{B} first searches the h_{ID} -list. If ID_i does not appear on the h_{ID} -list, \mathcal{B} makes an h_{ID} query by itself and thus there is a tuple $(ID_i, r_i, h_{\text{ID}}(ID_i))$ on the h_{ID} -list. If $ID_i \in \{ID_{A_0}, ID_{A_1}, ID_B\}$, \mathcal{B} aborts and the simulation fails. Otherwise, \mathcal{B} responds with $mpk^{r_i} = (g_1^c)^{r_i} = h_{\text{ID}}(ID_i)^{c r_i} = h_{\text{ID}}(ID_i)^{msk}$.

h_p queries. At any time the adversary \mathcal{A} can query the random oracle h_p . To respond to these queries algorithm \mathcal{B} maintains a list of tuples $(m_j, k_j, \sigma_j, c_j)$ as explained below. We refer to this list as the h_p -list. The list is initially empty. When \mathcal{A} makes a request (m_i, k_i) to the oracle h_p , algorithm \mathcal{B} first checks if $(g_1, mpk, h_{\text{ID}}(ID_{A_0}), h_{\text{ID}}(ID_{A_1}), k_i)$, $(g_1, mpk, h_{\text{ID}}(ID_{A_0}), h_{\text{ID}}(ID_B), k_i)$ or $(g_1, mpk, h_{\text{ID}}(ID_{A_1}), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple with the help of Decisional Bilinear Diffie-Hellman oracle.

1. If none of them is a BDH tuple, algorithm \mathcal{B} searches the h_p -list. If (m_i, k_i) already appears on the h_p -list in a tuple $(m_i, k_i, \sigma_i, c_i)$, algorithm \mathcal{B} responds with $h(m_i, k_i) = \sigma_i$. Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = 0$. \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h_p -list and responds with $h_p(m_i, k_i) = \sigma_i$.
2. Else, if $(g_1, mpk, h_{\text{ID}}(ID_{A_0}), h_{\text{ID}}(ID_{A_1}), k_i)$ is a BDH-tuple, algorithm \mathcal{B} searches the h_p -list.
 - (a) If (m_i, A_0A_1) already appears on the h_p -list in a tuple either with the form $(m_i, k_i, \sigma_i, A_0A_1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0A_1)$ (as the result of signature queries), algorithm \mathcal{B} updates that tuple with $(m_i, k_i, \sigma_i, A_0A_1)$ if necessary and responds with $h_p(m_i, k_i) = \sigma_i$.
 - (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = A_0A_1$, which indicates that $(g_1, mpk, h_{\text{ID}}(ID_{A_0}), h_{\text{ID}}(ID_{A_1}), k_i)$ is a BDH-tuple and σ_i is a valid designated verifier signature of message m_i between ID_{A_0} and ID_{A_1} . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h_p -list and responds with $h_p(m_i, k_i) = \sigma_i$.
3. Else, if $(g_1, mpk, h_{\text{ID}}(ID_{A_0}), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple, algorithm \mathcal{B} searches the h_p -list.
 - (a) If (m_i, A_0B) already appears on the h_p -list in a tuple either with the form $(m_i, k_i, \sigma_i, A_0B)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0B)$

- (as the result of signature queries), algorithm \mathcal{B} updates that tuple with $(m_i, k_i, \sigma_i, A_0B)$ if necessary and responds with $h_p(m_i, k_i) = \sigma_i$.
- (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = A_0B$, which indicates that $(g_1, mpk, h_{\text{ID}}(ID_{A_0}), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple and σ_i is a valid designated verifier signature of message m_i between ID_{A_0} and ID_B . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h_p -list and responds with $h_p(m_i, k_i) = \sigma_i$.
4. Otherwise, $(g_1, mpk, h_{\text{ID}}(ID_{A_1}), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple, algorithm \mathcal{B} searches the h_p -list.
- (a) If (m_i, A_1B) already appears on the h_p -list in a tuple either with the form $(m_i, k_i, \sigma_i, A_1B)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_1B)$ (as the result of signature queries), algorithm \mathcal{B} updates that tuple with $(m_i, k_i, \sigma_i, A_1B)$ if necessary and responds with $h_p(m_i, k_i) = \sigma_i$.
- (b) Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = A_1B$, which indicates that $(g_1, mpk, h_{\text{ID}}(ID_{A_1}), h_{\text{ID}}(ID_B), k_i)$ is a BDH-tuple and σ_i is a valid designated verifier signature of message m_i between ID_{A_1} and ID_B . \mathcal{B} then adds $(m_i, k_i, \sigma_i, c_i)$ on the h_p -list and responds with $h_p(m_i, k_i) = \sigma_i$.

Signature Queries: For a signature query (m_i, ID_S, ID_V) chosen by \mathcal{A} , if $\{ID_S, ID_V\} \not\subseteq \{ID_{A_0}, ID_{A_1}, ID_B\}$, \mathcal{B} can calculate either sk_{ID_S} or sk_{ID_V} and thus is able to calculate $k_{SV} = e(sk_{ID_S}, h_{\text{ID}}(ID_V))$. \mathcal{B} then makes an h_p request (m_i, k_{SV}) and returns $h_p(m_i, k_{SV})$ to \mathcal{A} . Otherwise $\{ID_S, ID_V\} \subseteq \{ID_{A_0}, ID_{A_1}, ID_B\}$, we distinguish the following cases.

1. $(ID_S, ID_V) \in \{(ID_{A_0}, ID_{A_1}), (ID_{A_1}, ID_{A_0})\}$.

For this case, if (m_i, A_0A_1) already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, A_0A_1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0A_1)$ (as the result of signature queries), algorithm \mathcal{B} responds with σ_i . Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = A_0A_1$, which indicates that σ_i is a valid designated verifier signature of message m_i between ID_{A_0} and ID_{A_1} . \mathcal{B} then adds $(m_i, \top, \sigma_i, c_i)$ on the h_p -list and responds with σ_i .

2. $(ID_S, ID_V) \in \{(ID_{A_0}, ID_B), (ID_B, ID_{A_0})\}$.

For this case, if (m_i, A_0B) already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, A_0B)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0B)$ (as the result of signature queries), algorithm \mathcal{B} responds with σ_i . Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = A_0B$, which indicates that σ_i is a valid designated verifier signature of message m_i between ID_{A_0} and ID_B . \mathcal{B} then adds $(m_i, \top, \sigma_i, c_i)$ on the h_p -list and responds with σ_i .

3. $(ID_S, ID_V) \in \{(ID_{A_1}, ID_B), (ID_B, ID_{A_1})\}$.

For this case, if (m_i, A_1B) already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, A_1B)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_1B)$ (as the result of signature queries), algorithm \mathcal{B} responds with σ_i . Otherwise, \mathcal{B} picks a random element $\sigma_i \in \mathbb{Z}_p^*$ and sets $c_i = A_1B$, which indicates that σ_i is a valid designated verifier signature of message m_i between ID_{A_1} and ID_B . \mathcal{B} then adds $(m_i, \top, \sigma_i, c_i)$ on the h_p -list and responds with σ_i .

Verification Queries: For a verification query $(m_i, \sigma, ID_S, ID_V)$, if $\{ID_S, ID_V\} \not\subseteq \{ID_{A_0}, ID_{A_1}, ID_B\}$, \mathcal{B} can calculate either sk_{ID_S} or sk_{ID_V} and thus is able to calculate $k_{SV} = e(sk_{ID_S}, h_{ID}(ID_V))$. \mathcal{B} then makes an h_p request (m_i, k_{SV}) and obtains $h_p(m_i, k_{SV})$. \mathcal{B} responds with **valid** if $h_p(m_i, k_{SV}) = \sigma$, and **invalid** otherwise. Otherwise $\{ID_S, ID_V\} \subseteq \{ID_{A_0}, ID_{A_1}, ID_B\}$, we distinguish the following cases.

1. $(ID_S, ID_V) \in \{(ID_{A_0}, ID_{A_1}), (ID_{A_1}, ID_{A_0})\}$.

For this case, if (m_i, A_0A_1) already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, A_0A_1)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0A_1)$ (as the result of signature queries), \mathcal{B} responds with **valid** if $\sigma_i = \sigma$, and **invalid** if $\sigma_i \neq \sigma$. Otherwise, (m_i, A_0A_1) does not appear on the h_p -list and \mathcal{B} responds with **invalid**. The answer could be incorrect when \mathcal{A} makes a signature query of $(m_i, ID_{A_0}, ID_{A_1})$ and σ is returned as the answer at some time later. However, this only happens with probability $\frac{1}{p-1}$ as m_i 's valid signature is randomly chosen in \mathbb{Z}_p^* .

2. $(ID_S, ID_V) \in \{(ID_{A_0}, ID_B), (ID_B, ID_{A_0})\}$.

Similarly, if (m_i, A_0B) already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, A_0B)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_0B)$

(as the result of signature queries), \mathcal{B} responds with **valid** if $\sigma_i = \sigma$, and **invalid** if $\sigma_i \neq \sigma$. Otherwise, (m_i, A_0B) does not appear on the h_p -list and \mathcal{B} responds with **invalid**. The answer could be incorrect when \mathcal{A} makes a signature query of (m_i, ID_{A_0}, ID_B) and σ is returned as the answer at some time later. However, this only happens with probability $\frac{1}{p-1}$ as m_i 's valid signature is randomly chosen in \mathbb{Z}_p^* .

3. $(ID_S, ID_V) \in \{(ID_{A_1}, ID_B), (ID_B, ID_{A_1})\}$.

Similarly, if (m_i, A_1B) already appears on the h_p -list in a tuple either with form $(m_i, k_i, \sigma_i, A_1B)$ (as the result of hash queries) or $(m_i, \top, \sigma_i, A_1B)$ (as the result of signature queries), \mathcal{B} responds with **valid** if $\sigma_i = \sigma$, and **invalid** if $\sigma_i \neq \sigma$. Otherwise, (m_i, A_1B) does not appear on the h_p -list and \mathcal{B} responds with **invalid**. The answer could be incorrect when \mathcal{A} makes a signature query of (m_i, ID_{A_1}, ID_B) and σ is returned as the answer at some time later. However, this only happens with probability $\frac{1}{p-1}$ as m_i 's valid signature is randomly chosen in \mathbb{Z}_p^* .

Output I: The adversary outputs $(m^*, ID_{A_0}^*, ID_{A_1}^*, ID_B^*)$, algorithm \mathcal{B} will abort if $\{ID_{A_0}^*, ID_{A_1}^*\} \neq \{ID_{A_0}, ID_{A_1}\}$ or $ID_B^* \neq ID_B$. If \mathcal{B} does not abort, \mathcal{B} first picks a random $b \in \{1, 0\}$ and makes a signature query (m^*, ID_{A_b}, ID_B) by itself. Let the result be σ^* , which is returned to \mathcal{A} .

Queries II: \mathcal{A} continues to make queries with restrictions described in Section 4.4.2. \mathcal{B} responds these queries as same as described above.

Output II: Eventually algorithm \mathcal{A} outputs its guess b' .

If $b' = b$ happens with probability greater than $1/2$, then either $(m^*, k^*, \sigma^*, A_0B)$ or $(m^*, k^*, \sigma^*, A_1B)$ must exist on the h_p -list with probability $1 - \frac{1}{p-1}$, as hash function h_p is simulated as a random oracle. If $(m^*, k^*, \sigma^*, A_0B)$ appears on the h_p -list, then $(g_1, mpk, h_{ID}(ID_{A_0}), h_{ID}(ID_B), k^*)$ is a BDH tuple. As $h_{ID}(ID_{A_0}) = (g_1^a)^{r_{A_0}}$, $h_{ID}(ID_B) = (g_1^b)^{r_B}$ and $mpk = (g_1^c)^{r_C}$, k^* must be equal to $(e(g_1, g_1)^{abc})^{r_{A_0}r_Br_C}$ and $e(g_1, g_1)^{abc} = (k^*)^{(r_{A_0}r_Br_C)^{-1} \pmod{p}}$. Algorithm \mathcal{B} outputs $(k^*)^{(r_{A_0}r_Br_C)^{-1} \pmod{p}}$ as the answer of the given GBDH instance. Otherwise, $(m^*, k^*, \sigma^*, A_1B)$ appears on the h_p -list and $(g_1, mpk, h_{ID}(ID_{A_1}), h_{ID}(ID_B), k^*)$ is a BDH tuple. Algorithm \mathcal{B} can output $(k^*)^{(r_{A_1}r_Br_C)^{-1} \pmod{p}}$ as the answer of the given GBDH instance.

\mathcal{B} can output a correct answer if and only if

1. The simulation is correct, which happens with the probability

$$\left(1 - \frac{3}{q_{h_{ID}}}\right)^{q_E} \cdot \left(1 - \frac{1}{p-1}\right)^{q_V} \cdot \frac{2}{q_{h_{ID}}(q_{h_{ID}} - 1)(q_{h_{ID}} - 2)};$$

2. \mathcal{A} outputs a correct guess $b' = b$, which happens with probability $1/2 + \varepsilon$; and
3. Either $(m^*, k^*, \sigma^*, A_0 B)$ or $(m^*, k^*, \sigma^*, A_1 B)$ appears on the h_p -list, which happens with the probability $1 - \frac{1}{p-1}$.

The overall success probability that \mathcal{B} can output a correct answer is

$$\left(1 - \frac{3}{q_{h_{ID}}}\right)^{q_E} \cdot \left(1 - \frac{1}{p-1}\right)^{q_V+1} \cdot \frac{2\varepsilon}{q_{h_{ID}}(q_{h_{ID}} - 1)(q_{h_{ID}} - 2)} \geq \frac{2\varepsilon}{e^4 q_H^3}.$$

Here, e is the base of the natural logarithm.

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond to q_H random oracle queries, q_E extract queries, q_S signature queries, q_V verification queries and compute $e(g_1, g_1)^{abc}$ from σ^* . We assume each query requires time at most $c_{(\mathbb{G}_1, \mathbb{G}_T)}$. Hence, the total running time is at most $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_H + q_E + q_S + q_V + 1)$. This completes the proof of Theorem 4.4.

4.5 Efficiency Comparison

In this section, we compare the efficiency of the proposed schemes with other existing schemes. Our comparison is based on the length of the signature and the number of operations required.

We first compare our scheme with the other two efficient strong DVS schemes proposed in [SKM03] and [LV04a]. Under the same security number where q is a 160-bit prime, the scheme in [SKM03] has signature length around 480 bits and the scheme in [LV04a] has signature length around 271 bits, whereas ours only requires 160 bits. In the aspect of computational cost, our scheme only requires two exponentiations in \mathbb{Z}_p^* and two hash operations for signature generation and verification, while the scheme in [SKM03] needs more than 4 exponentiations in \mathbb{Z}_p^* and the scheme in [LV04a] requires pairings over elliptic curves. The comparison is summarized in Table 4.1.

Next, we compare our identity-based strong designated verifier signature scheme with another similar scheme proposed in [SZM04]. The signature of our scheme is only an element in \mathbb{Z}_p , while the signature in [SZM04] consists of two elements in \mathbb{Z}_p .

Table 4.1: Comparison with Two Efficient DVS Schemes [SKM03, LV04a]

Scheme	Signature Length	Exp. in \mathbb{Z}_p^*	Pairing
<i>The Scheme in [SKM03]</i>	480 bits	4	0
<i>The Scheme in [LV04a]</i>	271 bits	0	4
<i>Our Scheme in Section 4.3</i>	160 bits	2	0

Exp.: Exponentiation.

Table 4.2: Comparison with An ID-based DVS Scheme [SZM04]

Scheme	Signature Length	Pairings
<i>The Scheme in [SZM04]</i>	491 bits	3
<i>Our Scheme in Section 4.4</i>	160 bits	2

and one element in \mathbb{G}_1 . Here, p is the order of bilinear groups. When p is a 160-bit prime, the signature length of the scheme in [SZM04] is around 491 bits. For the computational cost, our scheme requires 2 pairing operations and 2 hash operations for signature generation and verification, while the scheme in [SZM04] needs more than 3 pairings. The comparison is summarized in Table 4.2.

4.6 Conclusion

In this chapter, we described a short strong designated verifier signature scheme and its identity-based variant. Our schemes outperform others in the literature, both in signature length and in computational cost. The security of our first scheme can be reduced to the hardness of Gap Diffie-Hellman problem in the random oracle model, and the security of our second scheme can be reduced to the hardness of Gap Bilinear Diffie-Hellman problem in the random oracle model. We note that both schemes proposed in this chapter are delegatable, details of which will be given in Section 5.5. Due to the analysis in Section 5.5 and in [LLP05, LWB05], there is no concrete construction of strong designated verifier signature with provable non-delegatability.

Chapter 5

New Constructions of Universal Designated Verifier Signatures

This chapter describes two new constructions of universal designated verifier signatures. Our first construction in Section 5.4 was published in the *International Journal of Information Security* [HSMW08]. It can be proven secure without random oracles and has the unconditional non-transferability property. Our second construction in Section 5.6 was presented at *ICICS 2006* [HSMW06]. It is the first universal designated verifier signature scheme with the non-delegatability property.

5.1 Introduction

In an ordinary digital signature scheme, one can verify the signature of a message with only public information. This is known as the public verifiability of ordinary digital signatures. To prove that the signer has signed the message, the signature holder can simply present the signature to the interested verifier. In an electronic world, the ease of copying and transmitting digital signatures may make the verifier easily disseminate the message and the signature, which will make an unlimited number of users believe the information contained on the message. This possibility poses a serious threat to the privacy of signature holder, especially when the message contains the private information such as hospital records, income summary, etc.

Universal designated verifier signature (UDVS), as introduced by Steinfeld, Bull, Wang and Pieprzyk [SBWP03] at Asiacrypt 2003, is an important tool to protect the privacy of signature holder. To prove to a verifier that the signer has signed the message, the signature holder does not need to send the signature to the verifier. Instead, the signature holder can convert the signer's ordinary signature to a UDVS which is designated to a verifier, such that only this designated verifier can believe the signer's signature on the message. Any other third parties will not believe it as

the designated verifier can use his/her secret key to create a valid UDVS, which is indistinguishable from the one produced by the signature holder. Given a UDVS, the designated verifier can believe that the signer has signed the message, as he/she did not generate that UDVS. When the signature holder and the signature signer are the same user, a universal designated verifier signature will form a designated verifier signature, which was introduced by Jakobsson *et al.* [JSI96]. Therefore, universal designated verifier signatures can be viewed as an application of general designated verifier signatures.

Related Work. From Boneh-Lynn-Shacham (or, BLS for short) signature [BLS01], Steinfeld, Bull, Wang and Pieprzyk [SBWP03] proposed the first construction of UDVS at Asiacrypt 2003. Another different construction of UDVS from BLS signature was presented by Vergnaud in [Ver06]. Steinfeld, Wang and Pieprzyk also showed how to obtain a UDVS scheme from the Schnorr/RSA signature scheme in PKC 2004 [SWP04]. Zhang, Susilo, Mu and Chen [ZSMC05] extended the notion of universal designated verifier signatures to identity-based cryptography and proposed two identity-based universal designated verifier signature schemes. However, all those UDVS schemes are proven secure in the random oracle model [BR93]. The first UDVS scheme with provable security without random oracles was proposed by Zhang, Furukawa and Imai [ZFI05]. Their scheme is based on Boneh-Boyen (or, BB for short) [BB04] signature without random oracles. Another UDVS scheme based on BB signature [BB04] was proposed by Vergnaud in [Ver06], where the universal designated verifier signature is generated in a different way from that in [ZFI05]. Based on Waters signature [Wat05], Laguillaumie, Libert and Quisquater [LLQ06] introduced two UDVS schemes without random oracles, whose security can be reduced to more classical complexity assumptions than those in [Ver06, ZFI05]. Laguillaumie and Vergnaud [LV07] pointed out that the notion “restricted universal designated verifier signatures” introduced in [HSMZ06a] is generically infeasible, as a signature holder may generically provide a proof that he/she has a certain signature without being punished. In Asiacrypt 2005, Baek, Safavi-Naini and Susilo [BSNS05] introduced the notion of universal designated verifier signature proof, which removes the requirement that the designated verifier must create a public key using the parameters of signer’s public key system. Baek *et al.* also provided two interactive protocols [BSNS05] based on BLS signature [BLS01] and BB signature [BB04], respectively.

Organization of This Chapter. The rest of this chapter is organized as follows. In the next section, we review the definitions of universal designated verifier signatures. We then revise the definition of non-transferability and analyze a universal designated verifier signature scheme without random oracles [ZFI05] in Section 5.3. We show that their scheme does not satisfy the non-transferability defined in this chapter. Section 5.4 describes a new construction of universal designated verifier signatures with non-transferability and provable security without random oracles.

In the second part of this chapter, we focus on the property “Non-delegatability” in universal designated verifier signatures. In Section 5.5, we review the definition of non-delegatability with some concrete examples. After that, Section 5.6 describes the first construction of universal designated verifier signatures with non-delegatability, which is proven secure in the random oracle model. Section 5.7 concludes this chapter.

5.2 Definitions of Universal Designated Verifier Signatures

This section will define the syntax and the security of universal designated verifier signatures.

Definition 5.1 *A universal designated verifier signature scheme is made up of six algorithms, KeyGen, PSign, PVerify, DSign, $\overline{\text{DSign}}$ and DVerify. For a fixed security parameter param , these algorithms work as follows:*

$\text{KeyGen}(\text{param}) \rightarrow (sk, pk)$.

This algorithm takes a common parameter param as input and returns a secret-public key pair (sk, pk) . In particular, the signer runs this algorithm to generate his/her key pair (sk_S, pk_S) and the verifier runs this algorithm to generate the key pair (sk_V, pk_V) . The public key includes the description of the message space and the signature space.

$\text{PSign}(\text{param}, sk_S, m) \rightarrow \sigma_{PV}$.

This algorithm takes the common parameter param , the signer’s secret key sk_S and the message m as inputs, and returns the signer’s publicly verifiable (PV) signature σ_{PV} .

$\text{PVerify}(param, pk_S, m, \sigma_{PV}) \rightarrow \{\text{valid}, \text{invalid}\}.$

This algorithm takes the common parameter $param$, the signer's public key pk_S , the signed message m and the PV signature σ_{PV} as inputs, and returns the verification decision **valid** or **invalid**.

$\text{DSign}(param, pk_S, pk_V, m, \sigma_{PV}) \rightarrow \sigma_{DV}.$

This algorithm takes the common parameter $param$, the signer's public key pk_S , the verifier's public key pk_V , the signed message m and the PV signature σ_{PV} as inputs, and outputs a universal designated verifier (DV) signature σ_{DV} . The signature holder of σ_{PV} runs this algorithm to generate σ_{DV} for the verifier with the public key pk_V .

$\overline{\text{DSign}}(param, pk_S, sk_V, m) \rightarrow \sigma_{DV}.$

This algorithm takes the common parameter $param$, the signer's public key pk_S , the verifier's secret key sk_V and the message m as inputs, and outputs the designated verifier (DV) signature σ_{DV} which is designated to the verifier with the corresponding public key pk_V .

$\text{DVerify}(param, pk_S, pk_V, sk_V, m, \sigma_{DV}) \rightarrow \{\text{valid}, \text{invalid}\}.$

This algorithm takes the common parameter $param$, the signer's public key pk_S , the verifier's public key pk_V , (possibly) the verifier's secret key sk_V , the signed message m and the DV signature σ_{DV} as inputs, and outputs the verification decision **valid** or **invalid**.

Remarks:

Compared with the models defined in [SBWP03, ZFI05], we add an additional algorithm $\overline{\text{DSign}}$ to describe how a designated verifier can create a valid universal designated verifier signature which is designated to himself/herself. It is also for the convenience to define the property *non-transferability*.

Correctness. Let $(sk_S, pk_S), (sk_V, pk_V)$ be the output of algorithm $\text{KeyGen}(param)$, the correctness of a universal designated verifier signature scheme includes:

1. Any publicly verifiable signature produced by **Sign** is valid. That is,

$$\text{PVerify}(param, pk_S, m, \text{PSign}(param, sk_S, m)) = \text{valid}.$$

2. Any universal designated verifier signature produced by **DSign** is valid. That is, for any $\sigma_{PV} = \text{PSign}(param, sk_S, m)$,

$$\text{DVerify}(param, pk_S, pk_V, sk_V, m, \text{DSign}(param, pk_S, pk_V, m, \sigma_{PV})) = \text{valid}.$$

3. Any universal designated verifier signature produced by $\overline{\text{DSign}}$ is valid. That is,

$$\text{DVerify}(param, pk_S, pk_V, sk_V, m, \overline{\text{DSign}}(param, pk_S, sk_V, m)) = \text{valid}.$$

Unconditional Non-Transferability. Let $(sk_S, pk_S), (sk_V, pk_V)$ be the output of algorithm $\text{KeyGen}(param)$ and σ_{PV} be the output of $\text{PSign}(param, sk_S, m)$, the unconditional non-transferability property requires that universal designated verifier signatures produced by $\text{DSign}(param, pk_S, pk_V, m, \sigma_{PV})$ have the identical probability distribution as those produced by $\overline{\text{DSign}}(param, pk_S, sk_V, m)$.

Remark: The unconditional non-transferability states that it should be unconditionally infeasible to determine if a universal designated verifier signature was produced by the signature holder or the designated verifier. Other researchers have defined that property in different ways and with different names. For instance, the property *source hiding* in [LLQ06] is defined as: There exists an algorithm that takes as input only the secret key of the designated verifier and which produces bit strings which are perfectly indistinguishable (even knowing all secret keys) from the distribution of actual designated verifier signatures. The property *source hiding* in [Ver06] is defined as: $\text{DSign}(param, pk_S, pk_V, m, \text{PSign}(param, sk_S, m)) = \overline{\text{DSign}}(param, pk_S, sk_V, m)$. In Section 5.3, we will show the difference among these definitions.

5.2.1 Unforgeability of Universal Designated Verifier Signatures

There are in fact two types of unforgeability properties in universal designated verifier signatures [SBWP03]. The first property, *publicly verifiable signature unforgeability* (PV-Unforgeability), is just the usual existential unforgeability notion under chosen message attacks [GMR88] for the standard publicly verifiable signature scheme PSign , which states that anyone should not be able to forge a PV signature of the signer on a new message. The second property, *designated verifier signature unforgeability* (DV-Unforgeability) is defined as below.

Setup: The challenger \mathcal{C} first runs algorithm $\text{KeyGen}(param)$ to obtain the signer's key pair (sk_S, pk_S) . Additionally, the challenger runs $\text{KeyGen}(param)$ several

times to obtain n potential verifier's key pairs (sk_{V_i}, pk_{V_i}) . The adversary \mathcal{A} is given $param, pk_S$ and $pk_{V_i}, i \in \{1, 2, \dots, n\}$.

PSign Queries: Proceeding adaptively, \mathcal{A} can ask the publicly verifiable signature σ_{PV} on the message m chosen by itself. The challenger \mathcal{C} responds with $\sigma_{PV} = \text{PSign}(param, sk_S, m)$.

DSign Queries: Proceeding adaptively, \mathcal{A} can ask the designated verifier signature σ_{DV} on the message m and the verifier's public key $pk_V \in \{pk_{V_1}, pk_{V_2}, \dots, pk_{V_n}\}$. In response, \mathcal{C} first runs $\text{PSign}(param, sk_S, m)$ algorithm to obtain a publicly verifiable signature σ_{PV} . Then, \mathcal{C} responds with $\sigma_{DV} = \text{DSign}(param, pk_S, pk_V, m, \sigma_{PV})$.

DVerify Queries: Proceeding adaptively, \mathcal{A} can ask the designated verification result of the message-signature pair (m, σ_{DV}) with the designated verifier's public key $pk_V \in \{pk_{V_1}, pk_{V_2}, \dots, pk_{V_n}\}$. In response, \mathcal{C} runs $\text{DV}(param, pk_S, pk_V, sk_V, m, \sigma_{DV})$ and returns the verification result to \mathcal{A} .

Key-Extract Queries: \mathcal{A} can make the secret key query of the public key $pk \in \{pk_{V_1}, pk_{V_2}, \dots, pk_{V_n}\}$. In response, \mathcal{C} returns the corresponding secret key sk to \mathcal{A} .

Output: Eventually, \mathcal{A} outputs a forged message-signature pair (m^*, σ_{DV}^*) with a public key $pk^* \in \{pk_{V_1}, pk_{V_2}, \dots, pk_{V_n}\}$ and wins the game if

1. $\text{DVerify}(param, pk_S, pk^*, sk^*, m^*, \sigma_{DV}^*) = \text{valid}$;
2. m^* is not one of PSign queries;
3. (m^*, pk^*) is not one of DSign queries; and
4. pk^* is not one of Key-Extract queries.

Let $\text{F-Adv}_{\mathcal{A}}$ be the probability that the adversary \mathcal{A} wins in the above game, taken over the coin tosses made by \mathcal{A} and the challenger.

Definition 5.2 *A forger \mathcal{A} is said to $(t, q_{PS}, q_{DS}, q_{DV}, q_{KE}, \varepsilon)$ -break the DV-unforgeability of a UDVS scheme if \mathcal{A} runs in time at most t , makes at most q_{PS} PSign queries, q_{DS} DSign queries, q_{DV} DVerify queries, q_{KE} Key-Extract queries and $\text{F-Adv}_{\mathcal{A}}$ is at least ε . A UDVS scheme is $(t, q_{PS}, q_{DS}, q_{DV}, q_{KE}, \varepsilon)$ -existentially unforgeable under an adaptive chosen public key and chosen message attack if there exists no forger that $(t, q_{PS}, q_{DS}, q_{DV}, q_{KE}, \varepsilon)$ -breaks it.*

Remark: In the random oracle model, we add an additional parameter q_H to denote an upper bound on the number of queries that the adversary \mathcal{A} makes to the random oracle. The unforgeability model defined here is not the strong unforgeability model in the sense of [BB04, ZFI05]. However, we note that this model is *practical* and has been *widely used* [GMR88]. In addition, as mentioned in [SBWP03], DV-Unforgeability always implies the PV-Unforgeability. Thus, it is sufficient to consider only DV-Unforgeability in universal designated verifier signatures.

5.3 Analysis of A UDVS Scheme without Random Oracles in [ZFI05]

Recently, Zhang *et al.* [ZFI05] proposed the first construction of UDVS without random oracles, where the PSign algorithm is BB short signature scheme [BB04]. This section will give a security analysis of Zhang *et al.*'s scheme. We first briefly review their scheme as following.

5.3.1 Review of Zhang *et al.*'s [ZFI05] UDVS Scheme

Let $\sigma_{PV} = (\sigma, r)$ denote BB's signature of a message m from a signer whose public key is $pk_S = (u_1, v_1)$. Algorithms DSign and $\overline{\text{DSign}}$ in [ZFI05] work as following:

DSign. $\sigma_{DV} = (\sigma_{DV_1}, \sigma_{DV_2}, \sigma_{DV_3}) = (\sigma, g_1^r, e(u_3, v_3^r))$, where (u_3, v_3) is the public key of the designated verifier. Here, for convenience, we let $\mathbb{G}_1 = \mathbb{G}_2$ and the generator of $\mathbb{G}_1(\mathbb{G}_2)$ be g_1 in their scheme.

$\overline{\text{DSign}}$. The designated verifier can output a valid UDVS signature for himself/herself using the secret key $sk_V = (x_3, y_3) \in \mathbb{Z}_p \times \mathbb{Z}_p$. For a message m , one chooses $s \in_R \mathbb{Z}_p$ and calculates:

$$\sigma_{DV_1} = g_1^s, \sigma_{DV_2} = g_1^{s^{-1}} u_1^{-1} v_1^{-m}, \sigma_{DV_3} = e(g_1, \sigma_{DV_2})^{x_3 y_3}.$$

Remark: Actually, Zhang *et al.* did not give the description of algorithm $\overline{\text{DSign}}$ directly. However, we can extract this algorithm from their proof of non-transferability privacy from the Theorem 3 in [ZFI05].

DVerify: Given the signer's public key (u_1, v_1) , the verifier's secret key (x_3, y_3) , the signed message m and the DV signature $\sigma_{DV} = (\sigma_{DV_1}, \sigma_{DV_2}, \sigma_{DV_3})$, this algorithm outputs **valid** if and only if

$$e(\sigma_{DV_1}, u_1 \sigma_{DV_2} v_1^m) \stackrel{?}{=} e(g_1, g_1) \text{ and } \sigma_{DV_3} \stackrel{?}{=} e(g_1, \sigma_{DV_2})^{x_3 y_3}.$$

5.3.2 Analysis of Non-Transferability

In this section, we will analyze the non-transferability of Zhang *et al.*'s scheme. We first show that their scheme does not satisfy the unconditional non-transferability defined in Section 5.2.

Algorithm DSign in [ZFI05] is a deterministic algorithm for a given publicly verifiable signature $\sigma_{PV} = (\sigma, r)$ and a designated verifier's public key $pk_V = (u_3, v_3)$. That is, there is only one universal designated verifier signature for the given input. On the other hand, the algorithm $\overline{\text{DSign}}$ is probabilistic due to the random salt s , which makes the output of $\text{DSign}(param, pk_S, pk_V, m, \sigma_{PV})$ has different probability distribution from that of $\overline{\text{DSign}}(param, pk_S, sk_V, m)$. Thus, Zhang *et al.*'s scheme [ZFI05] does not satisfy unconditional non-transferability defined in Section 5.2.

We now show how Zhang *et al.*'s scheme [ZFI05] could also violate the property of non-transferability in practice. Suppose the signature holder wants to prove two verifiers V_A and V_B that he/she has a publicly verifiable signature σ_{PV} on message m . By using Zhang *et al.*'s scheme [ZFI05], the signature holder can produce two universal designated verifier signatures $\sigma_{(DV, V_A)}$ and $\sigma_{(DV, V_B)}$. Given $\sigma_{(DV, V_A)}$ and $\sigma_{(DV, V_B)}$, one will believe that, with overwhelming probability, those two signatures are created by the signature holder using the same publicly verifiable signature.

1. If $\sigma_{(DV, V_A)}$ and $\sigma_{(DV, V_B)}$ are created by the signature holder using the same publicly verifiable signature, then the first two parts of $\sigma_{(DV, V_A)}$ and $\sigma_{(DV, V_B)}$ must be identical. Namely, $\sigma_{(DV_1, V_A)} = \sigma_{(DV_1, V_B)} = \sigma$ and $\sigma_{(DV_2, V_A)} = \sigma_{(DV_2, V_B)} = g^r$ where (σ, r) is the publicly verifiable signature on the message m . Therefore

$$\begin{aligned} \Pr [& \sigma_{(DV_1, V_A)} = \sigma_{(DV_1, V_B)} \wedge \sigma_{(DV_2, V_A)} = \sigma_{(DV_2, V_B)} | \\ & \sigma_{(DV, V_A)} = \text{DS}(pk_S, pk_{v_A}, m, \sigma, r) \wedge \sigma_{(DV, V_B)} = \text{DS}(pk_S, pk_{v_B}, m, \sigma, r)] = 1. \end{aligned}$$

2. However, if $\sigma_{(DV, V_A)}$ is produced by $\overline{\text{DSign}}$ using the secret key of verifier V_A

and $\sigma_{(\text{DV}, V_B)}$ is produced by $\overline{\text{DSign}}$ using the secret key of verifier V_B , then

$$\Pr \left[\begin{aligned} &\sigma_{(\text{DV}_1, V_A)} = \sigma_{(\text{DV}_1, V_B)} \wedge \sigma_{(\text{DV}_2, V_A)} = \sigma_{(\text{DV}_2, V_B)} \\ &\sigma_{(\text{DV}, V_A)} = \overline{\text{DSign}}(pk_S, sk_{v_A}, m) \wedge \sigma_{(\text{DV}, V_B)} = \overline{\text{DSign}}(pk_S, sk_{v_B}, m) \end{aligned} \right] = 1/p.$$

As p is always large enough, this probability is negligible.

Therefore, if V_A and V_B do not collude together, one only needs to check the equality of the first two parts of $\sigma_{(\text{DV}, V_A)}$ and $\sigma_{(\text{DV}, V_B)}$, and will find out who created them. We note that if such collusion happens, V_A and V_B can use the same random number s in $\overline{\text{DSign}}$ algorithm to generate $\sigma_{(\text{DV}, V_A)}$ and $\sigma_{(\text{DV}, V_B)}$, then the first two parts of these two signatures will be identical with probability 1.

In [ZFI05], Zhang *et al.* use BB signature scheme [BB04] as the PSign algorithm. BB signature [BB04] is strongly unforgeable, that is, given a valid signature σ_{PV} of a message m , it is computationally infeasible to create another signature σ'_{PV} such that σ'_{PV} is a valid signature on the message m while $\sigma'_{\text{PV}} \neq \sigma_{\text{PV}}$. Therefore, the signature holder must use the same signature to generate different universal designated verifier signatures for different verifiers. This is the reason why in [ZFI05] the first two components of different universal designated verifier signatures with different designated verifiers are *identical*. We note that the other two UDVS schemes without random oracles in [LLQ06] do not have the unconditional non-transferability either, as the algorithm DSign is deterministic while the algorithm $\overline{\text{DSign}}$ is probabilistic.

If the PSign algorithm is not strongly unforgeable but is only existentially unforgeable against chosen message attacks as defined in [GMR88], then given a valid signature σ_{PV} of the message m , the signature holder can create many different valid signatures σ_{PVs} of the same message m . Therefore the signature holder can use different σ_{PVs} to create universal designated verifier signatures for different verifiers. Following this idea, we will show in Section 5.4 how to create a universal designated verifier signature scheme with unconditional non-transferability defined in Section 5.2.

5.4 A New Construction of UDVS without Random Oracles

In this section, we incorporate Waters signature scheme [Wat05] to obtain a new universal designated verifier signature scheme without random oracles. The formal

security analysis of the proposed scheme is also provided in this section.

5.4.1 The Proposed Scheme

We now describe our construction of universal designated verifier signature without random oracles. Let e be the pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ described in Section 1.5. The generator of \mathbb{G}_1 is denoted as g_1 . The order of \mathbb{G}_1 is a prime p . The messages m to be signed in this scheme will be represented as bitstrings of length n , a separate parameter unrelated to p . Furthermore, there are $n + 1$ random elements $u', u_1, u_2, \dots, u_n \in_R \mathbb{G}_1$ and set $\vec{u} = (u_1, u_2, \dots, u_n)$. Then the common parameter $param = (\mathbb{G}_1, \mathbb{G}_T, p, g_1, e, n, u', \vec{u})$.

KeyGen. The signer picks two secret values $x_s, y_s \in_R \mathbb{Z}_p^*$ and sets the secret key $sk_S = (x_s, y_s)$. Then the signer calculates the public key $pk_S = (pk_{sx}, pk_{sy}) = (g_1^{x_s}, g_1^{y_s})$. Similarly, the verifier picks two secret values $x_v, y_v \in_R \mathbb{Z}_p^*$, sets the secret key $sk_V = (x_v, y_v)$ and calculates the public key $pk_V = (pk_{vx}, pk_{vy}) = (g_1^{x_v}, g_1^{y_v})$.

PSign. Let m be an n -bit message to be signed by the signer, m_i denote the i^{th} bit of m , and $\mathcal{M} \in \{1, \dots, n\}$ be the set of all i for which $m_i = 1$, a Waters' signature is generated as follows. First, a random $r \in \mathbb{Z}_p$ is chosen. Then the signature is constructed as: $\sigma_{PV} = (\sigma_{PV_1}, \sigma_{PV_2}) = (g_1^{x_s y_s} (u' \prod_{i \in \mathcal{M}} u_i)^r, g_1^r)$.

PVerify. Given $\sigma_{PV} = (\sigma_{PV_1}, \sigma_{PV_2})$, a message m and the signer's public key $pk_S = (pk_{sx}, pk_{sy})$. This algorithm outputs **valid** if $e(\sigma_{PV_1}, g_1) / e(u' \prod_{i \in \mathcal{M}} u_i, \sigma_{PV_2}) = e(pk_{sx}, pk_{sy})$, and **invalid** otherwise.

DSign. Given the designated verifier's public key $pk_V = (pk_{vx}, pk_{vy})$, the signature holder selects $r' \in_R \mathbb{Z}_p$ and calculates $\sigma_{DV_1} = e(\sigma_{PV_1} \cdot (u' \prod_{i \in \mathcal{M}} u_i)^{r'}, pk_{vx}) = e(g_1^{x_s y_s} \cdot (u' \prod_{i \in \mathcal{M}} u_i)^{r+r'}, pk_{vx})$ and $\sigma_{DV_2} = \sigma_{PV_2} \cdot g_1^{r'} = g_1^{r+r'}$. Then, the signature holder sends $\sigma_{DV} = (\sigma_{DV_1}, \sigma_{DV_2})$ to the designated verifier with the public key pk_V .

DSign. The designated verifier can also produce a universal designated verifier signature for himself/herself. To to that, one only needs to select a random $r \in \mathbb{Z}_p$ and calculates $\sigma_{DV_2} = g_1^r$ and $\sigma_{DV_1} = e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}} u_i, \sigma_{DV_2})^{x_v}$.

DVerify. Given the signer's public key $pk_S = (pk_{sx}, pk_{sy})$, a message m , and a signature $(\sigma_{DV_1}, \sigma_{DV_2})$, check if $\sigma_{DV_1} = e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}} u_i, \sigma_{DV_2})^{x_v}$. If

the equality holds, this algorithm outputs **valid**. Otherwise, the output is **invalid**.

Correctness. The correctness of the proposed scheme is shown as below:

1. If the publicly verifiable signature $\sigma_{\text{PV}} = (\sigma_{\text{PV}_1}, \sigma_{\text{PV}_2})$ of message m is generated by algorithm **PSign**, then

$$\begin{aligned} \frac{e(\sigma_{\text{PV}_1}, g_1)}{e(u' \prod_{i \in \mathcal{M}} u_i, \sigma_{\text{PV}_2})} &= \frac{e(g_1^{x_s y_s} (u' \prod_{i \in \mathcal{M}} u_i)^r, g_1)}{e(u' \prod_{i \in \mathcal{M}} u_i, g_1^r)} \\ &= e(g_1^{x_s y_s}, g_1) = e(pk_{sx}, pk_{sy}). \end{aligned}$$

Therefore, $\text{PVerify}(param, pk_S, m, \text{PSign}(param, sk_S, m)) = \text{valid}$.

2. If the universal designated verifier signature $\sigma_{\text{DV}} = (\sigma_{\text{DV}_1}, \sigma_{\text{DV}_2})$ is generated by algorithm **DSign**, then

$$\begin{aligned} \sigma_{\text{DV}_1} &= e(\sigma_{\text{PV}_1} \cdot (u \prod_{i \in \mathcal{M}} u_i)^{r'}, pk_{vx}) \\ &= e(g_1^{x_s y_s} (u' \prod_{i \in \mathcal{M}} u_i)^r (u' \prod_{i \in \mathcal{M}} u_i)^{r'}, g_1^{x_v}) \\ &= e(g_1^{x_s y_s}, g_1^{x_v}) e((u' \prod_{i \in \mathcal{M}} u_i)^{r+r'}, g_1^{x_v}) \\ &= e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}} u_i, g_1^{r+r'})^{x_v} \\ &= e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}} u_i, \sigma_{\text{DV}_2})^{x_v}. \end{aligned}$$

Therefore, $\text{DVerify}(param, pk_S, sk_V, m, \text{DSign}(param, pk_S, pk_V, m, \sigma_{\text{PV}})) = \text{valid}$.

3. If the designated verifier signature $\sigma_{\text{DV}} = (\sigma_{\text{DV}_1}, \sigma_{\text{DV}_2})$ is generated by algorithm **DSign**, then $\sigma_{\text{DV}_1} = e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}'} u_i, \sigma_{\text{DV}_2})^{x_v}$. Therefore, $\text{DVerify}(param, pk_S, sk_V, m, \text{DSign}(param, pk_S, sk_V, m)) = \text{valid}$.

5.4.2 Security Analysis: Unforgeability

This section will show that the unforgeability of the proposed scheme is based on the hardness of Gap Bilinear Diffie-Hellman problem defined in Section 1.5.

Theorem 5.1 *The proposed universal designated verifier signature scheme is $(t, q_{\text{PS}}, q_{\text{DS}}, q_{\text{DV}}, q_{\text{KE}}, \varepsilon)$ -existentially unforgeable under an adaptive chosen public key and*

chosen message attack, if Gap Bilinear Diffie-Hellman problem is (t', ε') -hard on $(\mathbb{G}_1, \mathbb{G}_T)$. Here,

$$t' = t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_{PS} + q_{DS} + q_{DV} + q_{KE} + 1),$$

$$\varepsilon' = \frac{\varepsilon}{8\mathfrak{e}(q_{KE} + 1)(n + 1)(q_{PS} + q_{DS} + q_{DV})},$$

$c_{(\mathbb{G}_1, \mathbb{G}_T)}$ is a constant on $(\mathbb{G}_1, \mathbb{G}_T)$ and \mathfrak{e} is the base of the natural logarithm.

Proof. Suppose there exists an attacker \mathcal{A} who can $(t, q_{PS}, q_{DS}, q_{DV}, q_{KE}, \varepsilon)$ break our proposed UDVS scheme. We show how to construct a $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_{PS} + q_{DS} + q_{DV} + q_{KE} + 1)$ -time algorithm \mathcal{B} which can solve the Gap Bilinear Diffie-Hellman (GBDH) problem on $(\mathbb{G}_1, \mathbb{G}_T)$ with probability at least $\frac{\varepsilon}{8\mathfrak{e}(q_{KE} + 1)(n + 1)(q_{PS} + q_{DS} + q_{DV})}$. This will contradict the assumption that GBDH is $(t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_{PS} + q_{DS} + q_{DV} + q_{KE} + 1, \frac{\varepsilon}{8\mathfrak{e}(q_{KE} + 1)(n + 1)(q_{PS} + q_{DS} + q_{DV})})$ -hard on $(\mathbb{G}_1, \mathbb{G}_T)$.

Given a random instance $(g_1, g_1^a, g_1^b, g_1^c)$ of the GBDH problem together with a Decisional Bilinear Diffie-Hellman oracle, we will show how \mathcal{B} can use \mathcal{A} to obtain $e(g_1, g_1)^{abc}$. Algorithm \mathcal{B} simulates the challenger and interacts with the forger \mathcal{A} as follows.

Setup: \mathcal{B} sets an integer $\ell = 4(q_{PS} + q_{DS} + q_{DV})$, and chooses an integer, k , uniformly at random between 0 and n . It then chooses a value x' and a random n -vector, $\vec{x} = (x_i)$ where $x', x_i \in_R \mathbb{Z}_\ell$. Additionally, \mathcal{B} chooses a value y' and a random n -vector $\vec{y} = (y_i)$ where $y', y_i \in_R \mathbb{Z}_p$. \mathcal{B} keeps all values secret.

For a message m , we let $\mathcal{M} \subseteq \{1, 2, \dots, n\}$ be the set of all i for which $m_i = 1$. To make the notation easy to follow, we define three functions $F(m)$, $J(m)$ and $K(m)$ as in [Wat05]:

1. $F(m) = (p - \ell k) + x' + \sum_{i \in \mathcal{M}} x_i$;
2. $J(m) = y' + \sum_{i \in \mathcal{M}} y_i$;
3. $K(m) = \begin{cases} 0, & \text{if } x' + \sum_{i \in \mathcal{M}} x_i \equiv 0 \pmod{\ell} \\ 1, & \text{otherwise} \end{cases}$

\mathcal{B} sets the public keys of users and the common parameter as following.

1. Firstly, \mathcal{B} assigns the signer's public key $pk_S = (pk_{sx}, pk_{sy}) = (g_1^a, g_1^b)$.
2. Secondly, \mathcal{B} maintains a list L of all secret-public key pairs of the verifiers.
To generate the i^{th} verifier V_i 's secret/public key pair, \mathcal{B} chooses a random

coin $c_i \in \{0, 1\}$ such that $\Pr[c_i = 1] = \delta$ (the value of the δ will be determined later).

- If $c_i = 0$, \mathcal{B} chooses two random numbers $d_i, e_i \in \mathbb{Z}_p$ and computes

$$pk_{v_i} = (pk_{v_i x}, pk_{v_i y}) = (g_1^{d_i}, g_1^{e_i}).$$

Then \mathcal{B} adds $(pk_{v_i x}, pk_{v_i y}, c_i, d_i, e_i)$ on list L .

- Otherwise, $c_i = 1$, \mathcal{B} chooses two random numbers $d_i, e_i \in \mathbb{Z}_p$ and computes

$$pk_{v_i} = (pk_{v_i x}, pk_{v_i y}) = ((g_1^c)^{d_i}, (g_1^c)^{e_i}).$$

\mathcal{B} then adds $(pk_{v_i x}, pk_{v_i y}, c_i, \perp, \perp)$ on list L . Here the notation \perp means that \mathcal{B} does not know the corresponding value.

3. Thirdly, \mathcal{B} assigns $u' = pk_{sy}^{p-k\ell+x'} g_1^{y'}$, $u_i = pk_{sy}^{x_i} g_1^{y_i}$ and sets $\vec{u} = (u_1, u_2, \dots, u_n)$.

\mathcal{B} returns the signer's public key pk_S , all verifiers' public keys pk_{v_i} and the common parameter $param = (\mathbb{G}_1, \mathbb{G}_T, p, g_1, e, n, u', \vec{u})$ to \mathcal{A} . From the perspective of the adversary all the distributions are identical to the real construction.

PSign Queries: Suppose \mathcal{A} makes a PSign query for the message m . If $K(m) \neq 0$ (If we have $K(m) \neq 0$ this implies $F(m) \neq 0 \pmod{p}$, since we can assume $p > n\ell$ for any reasonable values of p, n , and ℓ [Wat05]), \mathcal{B} can construct the publicly verifiable signature by choosing a random $r \in \mathbb{Z}_p$ and calculates:

$$\sigma_{PV} = (\sigma_{PV_1}, \sigma_{PV_2}) = \left(pk_{sx}^{\frac{-J(m)}{F(m)}} (u' \prod_{i \in \mathcal{M}} u_i)^r, pk_{sx}^{\frac{-1}{F(m)}} g_1^r \right).$$

σ_{PV} is a valid signature as

$$\begin{aligned} \sigma_{PV_1} &= pk_{sx}^{\frac{-J(m)}{F(m)}} (u' \prod_{i \in \mathcal{M}} u_i)^r \\ &= pk_{sx}^{\frac{-J(m)}{F(m)}} (pk_{sy}^{F(m)} g_1^{J(m)})^r \\ &= pk_{sy}^a (pk_{sy}^{F(m)} g_1^{J(m)})^{\frac{-a}{F(m)}} (pk_{sy}^{F(m)} g_1^{J(m)})^r \\ &= pk_{sy}^a (pk_{sy}^{F(m)} g_1^{J(m)})^{r - \frac{a}{F(m)}} \\ &= pk_{sy}^a (pk_{sy}^{F(m)} g_1^{J(m)})^{\tilde{r}} = pk_{sy}^a (u' \prod_{i \in \mathcal{M}} u_i)^{\tilde{r}}. \end{aligned}$$

Note that $\sigma_{PV_2} = pk_{sx}^{\frac{-1}{F(m)}} g_1^r = g_1^{\frac{-a}{F(m)}} g_1^r = g_1^{r - \frac{a}{F(m)}} = g_1^{\tilde{r}}$.

Otherwise, $K(m) = 0$. \mathcal{B} terminates the simulation and reports failure.

DSign Queries: Suppose \mathcal{A} makes a DSign query for a message m and the designated verifier pk_{v_i} . If $K(m) \neq 0$, \mathcal{B} can obtain the publicly verifiable signature $\sigma_{PV} = (\sigma_{PV_1}, \sigma_{PV_2})$ as described in PSign Queries. Then \mathcal{B} chooses a random $r' \in \mathbb{Z}_p$ and calculates the universal designated verifier signature as

$$\sigma_{DV_1} = e(\sigma_{PV_1} \cdot (u' \prod_{i \in \mathcal{M}} u_i)^{r'}, pk_{v_{ix}}), \sigma_{DV_2} = \sigma_{PV_2} \cdot g_1^{r'}.$$

and returns $(\sigma_{DV_1}, \sigma_{DV_2})$ to \mathcal{A} as the answer. Else, if $K(m) = 0$ and \mathcal{B} checks the tuple $(pk_{v_{ix}}, pk_{v_{iy}}, c_i, d_i, e_i)$ on list L . If $c_i = 0$, \mathcal{B} can generate the universal designated verifier signature by running algorithm $\overline{\text{DSign}}$. Otherwise, $K(m) = 0$ and $c_i = 1$. For this case, \mathcal{B} terminates the simulation and reports failure.

DVerify Queries: Suppose \mathcal{A} makes a DVerify query for the message-signature pair $(m, \sigma_{DV_1}, \sigma_{DV_2})$ and the designated verifier whose public key is $pk_{v_i} = (pk_{v_{ix}}, pk_{v_{iy}})$. Algorithm \mathcal{B} responds as follows:

1. If $K(m) \neq 0$, \mathcal{B} can calculate a valid universal designated verifier signature on this message as it responds to DSign queries. Let $(\sigma'_{DV_1}, \sigma'_{DV_2})$ denote the signature computes by \mathcal{B} . Then \mathcal{B} submits

$$\left(g_1, u' \prod_{i \in \mathcal{M}} u_i, pk_{v_{ix}}, \frac{\sigma_{DV_2}}{\sigma'_{DV_2}}, \frac{\sigma_{DV_1}}{\sigma'_{DV_1}} \right)$$

to the DBDH oracle $\mathcal{O}_{\text{DBDH}}$. \mathcal{B} outputs **valid** to \mathcal{A} if the above tuple is a valid BDH tuple, otherwise outputs **invalid**.

Correctness:

If $(\sigma_{DV_1}, \sigma_{DV_2} = g_1^r)$ is a valid signature, then

$$\sigma_{DV_1} = e(g_1^{\text{ab}}, pk_{v_{ix}}) e(u' \prod_{i \in \mathcal{M}} u_i, pk_{v_{ix}})^r.$$

Similarly, since $(\sigma'_{DV_1}, \sigma'_{DV_2} = g_1^{r'})$ is another valid signature produced by \mathcal{B} , then

$$\sigma'_{DV_1} = e(g_1^{\text{ab}}, pk_{v_{ix}}) e(u' \prod_{i \in \mathcal{M}} u_i, pk_{v_{ix}})^{r'}$$

Therefore,

$$\frac{\sigma_{DV_1}}{\sigma'_{DV_1}} = e(u' \prod_{i \in \mathcal{M}} u_i, pk_{v_{ix}})^{r-r'} \text{ and } \frac{\sigma_{DV_2}}{\sigma'_{DV_2}} = g_1^{r-r'}.$$

It follows that

$$\left(g_1, u' \prod_{i \in \mathcal{M}} u_i, pk_{v_i x}, \frac{\sigma_{DV_2}}{\sigma'_{DV_2}}, \frac{\sigma_{DV_1}}{\sigma'_{DV_1}} \right)$$

is a valid BDH tuple.

2. Else $K(m) = 0$ and $F(m) = 0$, \mathcal{B} submits

$$\left(g_1, pk_{sx}, pk_{sy}, pk_{v_i x}, \frac{\sigma_{DV_1}}{e(pk_{v_i x}, \sigma_{DV_2})^{J(m)}} \right).$$

to the DBDH oracle \mathcal{O}_{DBDH} . \mathcal{B} outputs **valid** to \mathcal{A} if the above tuple is a valid BDH tuple, otherwise outputs **invalid**.

Correctness:

If $(\sigma_{DV_1}, \sigma_{DV_2} = g_1^r)$ is a valid signature, then

$$\sigma_{DV_1} = e(g_1^{ab}, pk_{v_i x}) e(u' \prod_{i \in \mathcal{M}} u_i, pk_{v_i x})^r.$$

Note that $F(m) = 0$, which means that $u' \prod_{i \in \mathcal{M}} u_i = g_1^{J(m)}$. Therefore, $\sigma_{DV_1} = e(g_1^{ab}, pk_{v_i x}) e(\sigma_{DV_2}, pk_{v_i x})^{J(m)}$, which means

$$\left(g_1, pk_{sx}, pk_{sy}, pk_{v_i x}, \frac{\sigma_{DV_1}}{e(\sigma_{DV_2}, pk_{v_i x})^{J(m)}} \right)$$

is a valid BDH tuple.

3. Otherwise, $K(m) = 0$ and $F(m) \neq 0$, \mathcal{B} terminates the simulation and reports failure.

Key-Extract Queries: Suppose \mathcal{A} requests the secret key of the verifier V_i . In response, \mathcal{B} checks the tuple $(pk_{v_i x}, pk_{v_i y}, c_i, d_i, e_i)$ on L .

1. If $c_i = 0$, which means $pk_{v_i x} = g_1^{d_i}, pk_{v_i y} = g_1^{e_i}$, \mathcal{B} returns d_i, e_i to \mathcal{A} .
2. Otherwise, \mathcal{B} terminates the simulation and reports failure.

If \mathcal{B} does not abort during the simulation, \mathcal{A} will output a valid universal designated verifier signature $(\sigma_{DV_1}^*, \sigma_{DV_2}^*)$ under the message m^* and the designated verifier V^* 's public key pk_v^* with success probability ε .

1. If $F(m^*) \neq 0$, \mathcal{B} will abort.
2. Else, \mathcal{B} checks the tuple $(pk_{v^* x}, pk_{v^* y}, c^*, d^*, e^*)$ on list L . If $c^* = 0$, \mathcal{B} will abort.

3. Otherwise, $F(m^*) = 0$ and $c^* = 1$ which means $pk_{v^*x} = (g_1^f)^{d^*}$. \mathcal{B} computes

$$\begin{aligned}
& \left(\frac{\sigma_{\text{DV}_1}^*}{e(pk_{v^*x}, \sigma_{\text{DV}_2}^*)^{J(m^*)}} \right)^{(d^*)^{-1}} \\
&= \left(\frac{e(pk_{sx}, pk_{sy})^{cd^*} e(u' \prod_{i \in \mathcal{M}} u_i, \sigma_{\text{DV}_2}^*)^{cd^*}}{e(pk_{v^*x}, \sigma_{\text{DV}_2}^*)^{J(m^*)}} \right)^{(d^*)^{-1}} \\
&= \left(\frac{e(pk_{sx}, pk_{sy})^{cd^*} e((g_1^b)^{F(m^*)} g_1^{J(m^*)}, \sigma_{\text{DV}_2}^*)^{cd^*}}{e((g_1^f)^{d^*}, \sigma_{\text{DV}_2}^*)^{J(m^*)}} \right)^{(d^*)^{-1}} \\
&= \left(\frac{e(g_1^a, g_1^b)^{cd^*} e(g_1^{J(m^*)}, \sigma_{\text{DV}_2}^*)^{cd^*}}{e(g_1^{cd^*}, \sigma_{\text{DV}_2}^*)^{J(m^*)}} \right)^{(d^*)^{-1}} \\
&= e(g_1, g_1)^{abc}.
\end{aligned}$$

This completes the description of the simulation. It remains to analyze the probability of \mathcal{B} not aborting. \mathcal{B} will not abort if all the following events happen:

- A : \mathcal{B} does not abort during PSign, DSign and DVerify queries;
- B : $c_i = 0$ during KeyExtract queries;
- C : $c^* = 1$;
- D : $F(m^*) = 0 \pmod{p}$.

The success probability is $\varepsilon' = \Pr[A \wedge B \wedge C \wedge D] \varepsilon$. This can be further written as $\varepsilon' = \Pr[B \wedge C] \Pr[A \wedge D] = \delta(1 - \delta)^{q_{\text{KE}}} \Pr[A \wedge D] \varepsilon$. If δ is set as $\frac{1}{q_{\text{KE}} + 1}$, then $\varepsilon' \geq \frac{1}{e^{(q_{\text{KE}} + 1)}} \Pr[A \wedge D] \varepsilon$, where e is the base of natural logarithm. Let $q_{\text{PS}} + q_{\text{DS}} + q_{\text{DV}} = q$, then

$$\begin{aligned}
& \Pr[A \wedge D] = \Pr[A] \Pr[D|A] \\
& \geq \Pr\left[\bigwedge_{i=1}^q K(m_i) \neq 0\right] \Pr\left[x + \sum_{i \in \mathcal{M}^*} x_i = \ell k | A\right] \\
&= (1 - \Pr\left[\bigvee_{i=1}^q K(m_i) = 0\right]) \Pr\left[x + \sum_{i \in \mathcal{M}^*} x_i = \ell k | A\right] \\
&\geq (1 - \frac{q}{\ell}) \Pr\left[x + \sum_{i \in \mathcal{M}^*} x_i = \ell k | A\right] \\
&= \frac{1}{n+1} (1 - \frac{q}{\ell}) \Pr[K(m^*) = 0 | A] \\
&= \frac{1}{n+1} (1 - \frac{q}{\ell}) \frac{\Pr[K(m^*) = 0]}{\Pr[A]} \Pr[A | K(m^*) = 0]
\end{aligned}$$

$$\begin{aligned}
&\geq \frac{1}{(n+1)\ell} \left(1 - \frac{q}{\ell}\right) \Pr[A|K(m^*) = 0] \\
&\geq \frac{1}{(n+1)\ell} \left(1 - \frac{q}{\ell}\right) \left(1 - \Pr\left[\bigvee_{i=1}^q K(m_i) = 0 \mid K(m^*) = 0\right]\right) \\
&= \frac{1}{(n+1)\ell} \left(1 - \frac{q}{\ell}\right)^2 \geq \frac{1}{(n+1)\ell} \left(1 - \frac{2q}{\ell}\right).
\end{aligned}$$

Therefore, \mathcal{B} can solve the given instance of GBDH problem with probability $\varepsilon' \geq \frac{1}{\mathfrak{e}(q_{\text{KE}}+1)} \frac{1}{(n+1)\ell} \left(1 - \frac{2q}{\ell}\right) \varepsilon$. We can optimize it by setting $\ell = 4q = 4(q_{\text{PS}} + q_{\text{DS}} + q_{\text{DV}})$, then

$$\varepsilon' \geq \frac{\varepsilon}{8\mathfrak{e}(q_{\text{KE}}+1)(n+1)(q_{\text{PS}} + q_{\text{DS}} + q_{\text{DV}})}.$$

Algorithm \mathcal{B} 's running time is the same as \mathcal{A} 's running time plus the time it takes to respond the queries from \mathcal{A} and calculate $e(g_1, g_1)^{\text{abc}}$ from \mathcal{A} 's output. We assume each operation requires time at most $c_{(\mathbb{G}_1, \mathbb{G}_T)}$, which depends on $(\mathbb{G}_1, \mathbb{G}_T)$. Hence, the total running time is at most $t + c_{(\mathbb{G}_1, \mathbb{G}_T)}(q_{\text{PS}} + q_{\text{DS}} + q_{\text{DV}} + q_{\text{KE}} + 1)$. This completes the proof of Theorem 5.1.

5.4.3 Security Analysis: Non-Transferability

We now show that in our scheme universal designated verifier signatures produced by $\text{DSign}(param, pk_S, pk_V, m, \sigma_{\text{PV}})$ has the identical probability distribution as those produced by $\overline{\text{DSign}}(param, pk_S, sk_V, m)$. We first review these two algorithms together with DVerify as below.

1. **DSign.** Given the designated verifier's public key $pk_V = (pk_{vx}, pk_{vy})$, the signature holder selects $r' \in_R \mathbb{Z}_p$ and calculates $\sigma_{\text{DV}_1} = e(\sigma_{\text{PV}_1} \cdot (u' \prod_{i \in \mathcal{M}} u_i)^{r'}, pk_{vx}) = e(g_1^{x_{sy}} \cdot (u' \prod_{i \in \mathcal{M}} u_i)^{r+r'}, pk_{vx})$ and $\sigma_{\text{DV}_2} = \sigma_{\text{PV}_2} \cdot g_1^{r'} = g_1^{r+r'}$. Then, the signature holder sends $\sigma_{\text{DV}} = (\sigma_{\text{DV}_1}, \sigma_{\text{DV}_2})$ to the designated verifier.
2. **$\overline{\text{DSign}}$.** The designated verifier can also produce a signature for himself/herself. He/she only needs to select a random $r \in \mathbb{Z}_p$ and computes $\sigma_{\text{DV}_2} = g_1^r$ and $\sigma_{\text{DV}_1} = e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}'} u_i, g_1^r)^{x_v}$.
3. **DVerify:** Given the signer's public key $pk_S = (pk_{sx}, pk_{sy})$, a message m and a signature $(\sigma_{\text{DV}_1}, \sigma_{\text{DV}_2})$, check if $\sigma_{\text{DV}_1} = e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}} u_i, \sigma_{\text{DV}_2})^{x_v}$. If the equality holds, the algorithm outputs **valid**, and otherwise outputs **invalid**.

For any valid universal designated verifier signature $(\sigma_{\text{DV}_1}^*, \sigma_{\text{DV}_2}^*) = (e(pk_{sx}, pk_{sy})^{x_v} \cdot e(u' \prod_{i \in \mathcal{M}} u_i, pk_{vx})^{r^*}, g_1^{r^*})$ where $r^* \in \mathbb{Z}_p$, we now show that

$$\Pr[(\sigma_{\text{DV}_1}^*, \sigma_{\text{DV}_2}^*) = \text{DSign}(param, pk_S, pk_V, m, \sigma_{\text{PV}})] = 1/p$$

and

$$\Pr[(\sigma_{\text{DV}_1}^*, \sigma_{\text{DV}_2}^*) = \overline{\text{DSign}}(param, pk_S, sk_V, m)] = 1/p.$$

Let $(\sigma_{\text{PV}_1}, \sigma_{\text{PV}_2}) = (g_1^{x_s y_s} \cdot (u' \prod_{i \in \mathcal{M}} u_i)^r, g_1^r)$ be a publicly verifiable signature. Given this as input, DSign will output $(\sigma_{\text{DV}_1}, \sigma_{\text{DV}_2}) = (e(g_1^{x_s y_s} \cdot (u' \prod_{i \in \mathcal{M}} u_i)^{r+r'}, pk_{vx}), g_1^{r+r'})$. It is evident that $(\sigma_{\text{DV}_1}^*, \sigma_{\text{DV}_2}^*) = (\sigma_{\text{DV}_1}, \sigma_{\text{DV}_2})$ if and only if $r^* = r + r'$. As r' is randomly chosen in \mathbb{Z}_p , $\Pr[(\sigma_{\text{DV}_1}^*, \sigma_{\text{DV}_2}^*) = \text{DSign}(param, pk_S, pk_V, m, \sigma_{\text{PV}})] = 1/p$.

Similarly, algorithm $\overline{\text{DSign}}$ first chooses a random integer $r \in \mathbb{Z}_p$ and calculates the signature as $(e(pk_{sx}, pk_{sy})^{x_v} e(u' \prod_{i \in \mathcal{M}'} u_i, g_1^r)^{x_v}, g_1^r)$. Therefore, for a given valid universal designated verifier signature $(\sigma_{\text{DV}_1}^*, \sigma_{\text{DV}_2}^*)$, it will be an output of $\overline{\text{DSign}}$ if and only if $r = r^*$. As r is randomly chosen in \mathbb{Z}_p , $\Pr[(\sigma_{\text{DV}_1}^*, \sigma_{\text{DV}_2}^*) = \overline{\text{DSign}}(param, pk_S, sk_V, m)] = 1/p$. This completes the analysis of the unconditional non-transferability of the proposed scheme.

5.4.4 Comparison with Other Schemes

In this section, we will compare our scheme with others in the literature, which have provable security without random oracles.

We first compare our scheme with the constructions in [ZFI05, Ver06] from BB signature [BB04]. As BB signature [BB04] is more computationally efficient than Waters signature [Wat05], the UDVS schemes in [ZFI05, Ver06] require less computational cost than ours. However, the construction in [ZFI05] does not have the property non-transferability as shown in Section 5.3. For the UDVS scheme without random oracles proposed in [Ver06], a universal designated verifier signature consists of three elements in \mathbb{G}_1 and one element in \mathbb{Z}_p , while ours only has two elements in \mathbb{G}_1 . Furthermore, the security of their UDVS scheme can only be reduced to the knowledge-of-exponent assumption [BP04], while ours is based on a more classical assumption: Computational Diffie-Hellman assumption. The comparison is summarized in Table 5.1.

We now compare the proposed scheme with the constructions in [LLQ06]. Although the schemes in [LLQ06] and our scheme are inspired by essentially same idea, they were designed independently. Our paper “Secure Universal Designated Verifier

Table 5.1: Comparison with A UDVS Scheme Without Random Oracles [Ver06]

Scheme	Signature Length	Complexity Assumption
<i>The Scheme in [Ver06]</i>	$3 \mathbb{G}_1 + \mathbb{Z}_p$	Knowledge-of-Exponent
<i>Our Scheme</i>	$2 \mathbb{G}_1 $	CDH

Signature without Random Oracles” was submitted to the International Journal of Information Security on June 9, 2006, while the paper “Universal Designated Verifier Signatures Without Random Oracles or Non-black Box Assumptions” [LLQ06] was presented at SCN 2006, the proceeding of which is not publicly available in springer database until August 31, 2006. In addition to that, as shown in Section 5.3, constructions in [LLQ06] do not have unconditional non-transferability defined in this chapter, while ours has that property but at the cost of a little more computations.

5.5 Delegatability of (Universal) Designated Verifier Signatures

Let $(sk_S, pk_S), (sk_V, pk_V)$ denote the secret-public key pairs of the signer and the designated verifier, respectively. Delegatability of a (universal) designated verifier signature scheme [LWB05] refers that the signer *delegates* the signing right to some other party \mathcal{A} by disclosing some side information $y_{sv} = f_s(sk_S, pk_V)$ that will enable \mathcal{A} to generate valid (universal) designated verifier signatures. Analogously, the designated verifier could also delegate this signing right by disclosing some side information $y'_{sv} = f_v(sk_V, pk_S)$. The implication of the delegatability of (universal) designated verifier signatures will confuse the designated verifier. When the designated verifier receives a valid (universal) designated verifier signature that is not generated by himself/herself, he/she can only be convinced that the signature was generated by someone who knows either y_{sv} or y'_{sv} . The notion non-delegatability was formally defined by Lipmaa, Wang and Bao in [LWB05], under the assumption that the signer (or, the verifier) maybe dishonest. Under this assumption, many existing (universal) designated verifier signature schemes are delegatable. Take the scheme in Section 5.4.1 for example, one can produce valid universal designated verifier signatures of any messages once he/she obtains $e(pk_{sx}, pk_{sy})^{x_v}$, which can be calculated either by the signer using x_s or by the verifier using x_v . Our designated

verifier signature schemes in Section 4.3 and Section 4.4.3 are also delegatable, as one can produce valid signatures with $g^{sk_A sk_B}$ and $e(h_{ID}(ID_A), h_{ID}(ID_B))^{msk}$, respectively. To explain the delegatability more clearly, we analyze the following two UDVS schemes which were proposed by Vergnaud [Ver06] in ICALP 2006. Please refer to [LLP05, LWB05] for the delegatability of other (universal) designated verifier signature schemes.

5.5.1 Vergnaud's UDVS-BB [Ver06]

In [Ver06], the author proposed two UDVS schemes which are designed for the devices with constrained computation capabilities as algorithms PSign and DSign are pairing-free. The first UDVS scheme in [Ver06] employs BB signature [BB04] to obtain a UDVS scheme without random oracles. The UDVS-BB [Ver06] consists of the following algorithms. (We rewrite their scheme with different notations in order to keep the consistence of the whole chapter).

Let e be the pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ described in Section 1.5. The generator of \mathbb{G}_1 is denoted as g_1 . The order of \mathbb{G}_1 is a prime $p \geq 2^k$, where k is the security parameter. The message space $D_{\mathcal{M}} = \mathbb{Z}_p^*$. The system parameter $param = \{\mathbb{G}_1, \mathbb{G}_T, p, k, e, g_1, D_{\mathcal{M}}\}$ which is shared by all users in the system.

KeyGen: The signer S picks two secret numbers $u_a, v_a \in_R \mathbb{Z}_p^*$ and sets the secret key $sk_S = (u_a, v_a)$. Then S calculates the public key $pk_S = (U_a, V_a) = (g_1^{u_a}, g_1^{v_a})$. Similarly, the verifier V 's secret-public key pair is $sk_V = (u_b, v_b), pk_V = (U_b, V_b) = (g_1^{u_b}, g_1^{v_b})$, where u_b, v_b are randomly chosen in \mathbb{Z}_p^* .

PSign: For a message $m \in D_{\mathcal{M}}$ to be signed, S chooses $r \in \mathbb{Z}_p^*$ and calculates the publicly verifiable signature $\sigma_{PV} = (\sigma_{PV_1}, \sigma_{PV_2}) = (r, g_1^{\frac{1}{u_a + m + v_a r}})$.

PVerify: Given a message m , a signature $\sigma_{PV} = (\sigma_{PV_1}, \sigma_{PV_2})$ and S 's public key pk_S , one can check whether $e(\sigma_{PV_2}, U_a \cdot g_1^m \cdot V_a^{\sigma_{PV_1}}) \stackrel{?}{=} e(g_1, g_1)$. If the equality holds, outputs **valid**. Otherwise, this algorithm outputs **invalid**.

DSign: Given the publicly verifiable signature $\sigma_{PV} = (\sigma_{PV_1}, \sigma_{PV_2}) = (r, g_1^{\frac{1}{u_a + m + v_a r}})$ and the verifier's public key pk_V , the signature holder selects $t \in_R \mathbb{Z}_p^*$ and computes $Q_1 = g_1^{\frac{t}{u_a + m + v_a r}}$, $Q_2 = (U_b)^t$ and $Q_3 = g_1^t$. The signature holder sends the universal designated verifier signature $\sigma_{DV} = (r, Q_1, Q_2, Q_3)$ to the verifier V .

DSign: Given the signer's public key pk_S and the message m , the verifier V chooses $t, r \in_R \mathbb{Z}_p^*$ and calculates $R = (U_a \cdot g_1^m \cdot V_a^r)^t$. The universal designated verifier signature generated by the verifier is $\sigma_{DV} = (r, Q_1, Q_2, Q_3)$ where $Q_1 = g_1^t$, $Q_2 = R^{u_b}$ and $Q_3 = R$.

DVerify: Given the designated verifier signature (r, Q_1, Q_2, Q_3) , the verifier checks whether $e(Q_1, U_a \cdot g_1^m \cdot V_a^r) \stackrel{?}{=} e(Q_3, g_1)$ and $e(Q_3, g_1^{u_b}) \stackrel{?}{=} e(Q_2, g_1)$. If both equalities hold, output **valid**. Otherwise, this algorithm outputs **invalid**.

Delegatability. We will show that the knowledge of $y_{sv} := (g_1^{u_b u_a}, g_1^{u_b v_a})$ is sufficient to generate a valid universal designated verifier signature of Vergnaud's UDVS-BB scheme. Given a message m and y_{sv} , anyone can choose $t, r \in_R \mathbb{Z}_q^*$ and calculate $R = (U_a \cdot g_1^m \cdot V_a^r)^t$. After that, one calculates $Q_1 = g_1^t$, $Q_2 = (g_1^{u_b u_a} \cdot U_b^m \cdot (g_1^{u_b v_a})^r)^t$ and $Q_3 = R$. Note that (r, Q_1, Q_2, Q_3) is a valid signature of Vergnaud's UDVS-BB since

$$\begin{aligned} e(Q_1, U_a \cdot g_1^m \cdot V_a^r) &= e(g_1^t, U_a \cdot g_1^m \cdot V_a^r) \\ &= e(g_1^{t(u_a + m + rv_a)}, g_1) = e((U_a \cdot g_1^m \cdot V_a^r)^t, g_1) \\ &= e(R, g_1) = e(Q_3, g_1). \end{aligned}$$

and

$$\begin{aligned} e(Q_3, g_1^{u_b}) &= e(R, g_1^{u_b}) = e(R^{u_b}, g_1) \\ &= e((U_a \cdot g_1^m \cdot V_a^r)^{tu_b}, g_1) \\ &= e((g_1^{u_a u_b} \cdot U_b^m \cdot (g_1^{u_b v_a})^r)^t, g_1) = e(Q_2, g_1). \end{aligned}$$

Both the signer and the verifier can calculate y_{sv} . The signer can use his/her secret key u_a, v_a to calculate $y_{sv} = (U_b^{u_a}, U_b^{v_a})$ where U_b is a part of the verifier's public key $pk_V = (U_b, V_b)$. Similarly, the verifier also can use his/her secret key (u_b, v_b) to calculate $y_{sv} = (U_a^{u_b}, V_a^{u_b})$ where (U_a, V_a) is the public key of the signer. This completes the analysis of the delegatability of UDVS-BB in [Ver06].

5.5.2 Vergnaud's UDVS-BLS [Ver06]

The second UDVS scheme UDVS-BLS in [Ver06] combines BLS short signature [BLS01] to obtain a UDVS with a shorter signature length. It consists of the following algorithms. (Again, we rewrite their scheme with different notations in order to keep the consistence of the whole chapter)

Let e be the pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ described in Section 1.5. The generator of \mathbb{G}_1 is denoted as g_1 . The order of \mathbb{G}_1 is a prime $p \geq 2^k$, where k is the security parameter. Let $h : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ be a secure cryptographic hash function. The message space $D_{\mathcal{M}} = \{0, 1\}^*$. The system parameter $param = \{\mathbb{G}_1, \mathbb{G}_T, p, k, g_1, e, h, D_{\mathcal{M}}\}$ which is shared by all users in the system.

KeyGen: The signer S picks a secret value $x_s \in_R \mathbb{Z}_p^*$ and sets the secret key $sk_S = x_s$. Then, S calculates the public key $pk_S = g_1^{x_s}$. Similarly, the verifier V 's secret-public key pair is $(sk_V, pk_V) = (x_v, g_1^{x_v})$ where x_v is randomly chosen in \mathbb{Z}_p^* .

PSign: For a message m to be signed, the signer S calculates the publicly verifiable signature $\sigma_{PV} = h(m)^{sk_S} \in \mathbb{G}_1$.

PVerify: Given a message m , a signature σ_{PV} and S 's public key pk_S , one can check the equation $e(\sigma_{PV}, g_1) \stackrel{?}{=} e(h(m), pk_S)$. If the equality holds, outputs **valid**. Otherwise, this algorithm outputs **invalid**.

DSign: Given a publicly verifiable signature σ_{PV} and the verifier's public key pk_V , the signature holder selects $t \in_R \mathbb{Z}_p^*$ and calculates $Q_1 = \sigma_{PV}^t$ and $Q_2 = pk_V^{t^{-1}}$. Then the signature holder sends the universal designated verifier signature $\sigma_{DV} = (Q_1, Q_2)$ to the verifier V .

DSign: Given the signer's public key pk_S and the message m , the verifier V chooses $t \in_R \mathbb{Z}_p^*$ and calculates $Q_1 = h(m)^{t^{-1}}$ and $Q_2 = (pk_S^{sk_V})^t$. The universal designated verifier signature generated by the verifier is $\sigma_{DV} = (Q_1, Q_2)$.

DVerify: Given the designated verifier signature (Q_1, Q_2) , the verifier checks whether $e(Q_1, Q_2) \stackrel{?}{=} e(h(m), pk_S^{sk_V})$. If the equality holds, outputs **valid**. Otherwise, this algorithm outputs **invalid**.

Delegatability:

We will show that the knowledge of $y_{sv} := g_1^{sk_S sk_V}$ is sufficient to generate a valid universal designated verifier signature of UDVS-BLS. Given a message m and y_{sv} , anyone can choose $t \in_R \mathbb{Z}_p^*$ and calculate $Q_1 = h(m)^t$, $Q_2 = y_{sv}^{t^{-1}}$. Note that (Q_1, Q_2) is a valid signature of UDVS-BLS since $e(Q_1, Q_2) = e(h(m)^t, y_{sv}^{t^{-1}}) = e(h(m), g_1^{sk_S sk_V}) = e(h(m), pk_S^{sk_V})$. Note that both the signer and the verifier can calculate y_{sv} . The signer can use his/her secret key sk_S to compute $y_{sv} = pk_V^{sk_S}$. Similarly, the verifier

also can use his/her secret key sk_V to compute $y_{sv} = pk_S^{sk_V}$. This completes the analysis of the delegatability of UDVS-BLS in [Ver06].

5.5.3 Definition of Non-Delegatability

A universal designated verifier signature can be regarded as non-interactive system of proofs of knowledge of the signer S 's publicly verifiable signature σ_{PV} or the verifier's secret key sk_V . Thus, both the signature holder and the designated verifier are able to generate that proof. As pointed out in [LWB05], the definition of the unforgeability does not consider the case when the signer or the verifier is dishonest. Namely, without disclosing $sk_S(sk_V)$, the signer (the verifier) may be able to delegate the signing right to some other party \mathcal{A} by disclosing some "side information" which will help \mathcal{A} produce valid (universal) designated verifier signatures on any messages.

The property non-delegatability in universal designated verifier signatures is defined as below.

Definition 5.3 *Let $(sk_S, pk_S), (sk_V, pk_V)$ be the secret-public key pairs of signer S and verifier V . Let \mathcal{A} be an algorithm, who does not necessarily know the signer's PV signature σ_{PV} of the message m or the secret key sk_V , can produce a valid universal designated verifier signature on the message m with non-negligible probability ε , we say that a universal designated verifier signature scheme is (τ, κ) non-delegatable if in time τ , there exists a knowledge extractor \mathcal{K} who can use \mathcal{A} to obtain σ_{PV} or sk_V with probability greater than κ .*

The property non-delegatability in designated verifier signatures is defined as below.

Definition 5.4 *Let $(sk_S, pk_S), (sk_V, pk_V)$ be the secret-public key pairs of signer S and verifier V . Let \mathcal{A} be an algorithm, who does not necessarily know the secret key sk_S or sk_V , can produce a valid designated verifier signature on the message m with non-negligible probability ε , we say that a designated verifier signature scheme is (τ, κ) non-delegatable if in time τ , there exists a knowledge extractor \mathcal{K} who can use \mathcal{A} to obtain sk_S or sk_V with probability greater than κ .*

5.6 Universal Designated Verifier Signatures without Delegatability

In this section, we will give a construction of universal designated verifier signature without delegatability together with its security analysis. Let e be the pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ described in Section 1.5. The generator of \mathbb{G}_1 is denoted as g_1 . The order of \mathbb{G}_1 is a prime $p \geq 2^k$, where k is the security parameter. Let $h_0 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$, $h_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be two hash functions. The common parameter $param = (\mathbb{G}_1, \mathbb{G}_T, p, k, g_1, e, h_0, h_1)$. Our scheme consists of the following concrete algorithms:

KeyGen. The signer picks an integer $x_s \in_R \mathbb{Z}_p^*$ and sets the secret key $sk_S = x_s$. Then, the signer calculates the public key $pk_S = g_1^{x_s}$. Similarly, the verifier's secret-public key pair is $(sk_V, pk_V) = (x_v, g_1^{x_v})$ where x_v is randomly chosen in \mathbb{Z}_p^* .

PSign: For a message m to be signed, the signer calculates the publicly verifiable signature $\sigma_{PV} = h_0(m)^{sk_S}$.

PVerify: Given a message m , a publicly verifiable signature σ_{PV} and the signer's public key pk_S , this algorithm outputs **valid** if $e(\sigma_{PV}, g_1) = e(h_0(m), pk_S)$. Otherwise, outputs **invalid**.

DSign: Given a publicly verifiable signature σ_{PV} and the verifier's public key pk_V , the signature holder selects $r, c_v, d_v \in_R \mathbb{Z}_p$ and calculates

1. $z_s = e(g_1, g_1)^r, z_v = g_1^{d_v} \cdot pk_V^{c_v}$;
2. $c = h_1(m, pk_S, pk_V, z_s, z_v)$; and
3. $c_s = c - c_v \pmod{p}, d_s = g_1^r (\sigma_{PV})^{-c_s}$.

Then, the signature holder sends the universal designated verifier signature $\sigma_{DV} = (c_s, c_v, d_s, d_v)$ to the verifier.

$\overline{\text{DSign}}$: Given the signer's public key pk_S and the message m , the verifier selects $r, c_s \in_R \mathbb{Z}_p, d_s \in_R \mathbb{G}_1$ and computes

1. $z_s = e(d_s, g_1) \cdot e(h_0(m), pk_S)^{c_s}, z_v = g_1^r$;
2. $c = h_1(m, pk_S, pk_V, z_s, z_v)$; and

$$3. \ c_v = c - c_s \pmod{p}, d_v = r - c_v sk_v \pmod{p}.$$

The universal designated verifier signature generated by the verifier is $\sigma_{DV} = (c_s, c_v, d_s, d_v)$.

Remark: In algorithms **DSign** and $\overline{\text{DSign}}$, we use the designated verifier technique [JSI96] where the prover proves that he/she either has a publicly verifiable signature σ_{PV} or has the verifier's secret key sk_v . Very recently, this idea has been generalized in [SSN08] which proposed a generic construction of universal designated verifier signature without delegatability.

DVerify: Given the universal designated verifier signature (c_s, c_v, d_s, d_v) , anyone can check whether

$$c_s + c_v \stackrel{?}{=} h_1(m, pk_S, pk_V, e(d_s, g_1) \cdot e(h_0(m), pk_S)^{c_s}, g_1^{d_v} \cdot pk_V^{c_v}) \pmod{p}$$

If the equality holds, outputs **valid**. Otherwise, outputs **invalid**.

Correctness: The correctness of the proposed scheme is shown as below:

1. If σ_{PV} is generated by the algorithm **PSign**, then $\sigma_{PV} = h_0(m)^{sk_S}$. Therefore, $e(\sigma_{PV}, g_1) = e(h_0(m)^{sk_S}, g_1) = e(h_0(m), pk_S)$. That is, $\text{PVerify}(param, pk_S, m, \text{PSign}(param, sk_S, m)) = \text{valid}$.
2. If the universal designated verifier signature σ_{DV} is generated by the algorithm **DSign**, then $\sigma_{DV} = (c_s, c_v, d_s, d_v)$ where $c_v, d_v \in_R \mathbb{Z}_p$ and

$$c_s = h_1(m, pk_S, pk_V, e(g_1, g_1)^r, g_1^{d_v} pk_V^{c_v}) - c_v \pmod{p}, r \in \mathbb{Z}_p \text{ and } d_s = g_1^r \cdot \sigma_{PV}^{-c_s}.$$

Therefore,

$$\begin{aligned} & h_1(m, pk_S, pk_V, e(d_s, g_1) e(h_0(m), pk_S)^{c_s}, g_1^{d_v} pk_V^{c_v}) \\ &= h_1(m, pk_S, pk_V, e(g_1, g_1)^r, g_1^{d_v} pk_V^{c_v}) = c_s + c_v \pmod{p} \end{aligned}$$

That is, $\text{DVerify}(param, pk_S, pk_V, m, \text{DSign}(param, pk_S, pk_V, \sigma_{PV}, m)) = \text{valid}$.

3. If the universal designated verifier signature σ_{DV} is generated by the algorithm $\overline{\text{DSign}}$, then $\sigma_{DV} = (c_s, c_v, d_s, d_v)$ where $c_s \in_R \mathbb{Z}_p, d_s \in_R \mathbb{G}_1$ and

$$c_v = h_1(m, pk_S, pk_V, e(d_s, g_1) e(h_0(m), pk_S)^{c_s}, g_1^r) - c_s \pmod{p}, r \in \mathbb{Z}_p$$

and $d_v = r - c_v sk_v \pmod{p}$. Therefore,

$$\begin{aligned} & h_1(m, pk_S, pk_V, e(d_s, g_1)e(h_0(m), pk_S)^{c_s}, g_1^{d_v} pk_V^{c_v}) \\ &= h_1(m, pk_S, pk_V, e(d_s, g_1)e(h_0(m), pk_S)^{c_s}, g_1^r) = c_s + c_v \pmod{p} \end{aligned}$$

That is, $\text{DVerify}(param, pk_S, pk_V, m, \overline{\text{DSign}}(param, pk_S, sk_V, m)) = \text{valid}$.

5.6.1 Security Analysis: Non-Transferability

Firstly we show that the distribution of signatures generated by algorithm DSign is uniform in $\mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{G}_1 \times \mathbb{Z}_p$. Let $\sigma_{\text{DV}} = (c_s, c_v, d_s, d_v)$ be the universal designated verifier signature generated by DSign algorithm,

$$\sigma_{\text{DV}} = (c_s, c_v, d_s, d_v) : \begin{cases} c_s = h_1(m, pk_S, pk_V, e(g_1, g_1)^r, g_1^{d_v} pk_V^{c_v}) - c_v \pmod{p}, \\ \text{where } r, c_v, d_v \in_R \mathbb{Z}_p \text{ and} \\ d_s = g_1^r \cdot (\sigma_{\text{PV}})^{-c_s} \end{cases}$$

As c_v, d_v are randomly chosen in \mathbb{Z}_p , c_s will be a random element in \mathbb{Z}_p (in the random oracle model) and d_s is uniformly distributed in \mathbb{Z}_p . Therefore, for a randomly chosen valid $\sigma^* = (c_s^*, c_v^*, d_s^*, d_v^*)$, the probability $\Pr[\sigma^* = \text{DSign}(param, pk_S, pk_V, m, \sigma_{\text{PV}})] = \frac{1}{p^4}$.

Next, we show that the distribution of signatures simulated by algorithm $\overline{\text{DSign}}$ is also uniform in $\mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{G}_1 \times \mathbb{Z}_p$.

$$\sigma_{\text{DV}} = (c_s, c_v, d_s, d_v) : \begin{cases} c_v = h_1(m, pk_S, pk_V, e(d_s, g_1)e(h_0(m), pk_S)^{c_s}, g_1^r) - c_s \\ \pmod{p}, \text{ where } r, c_s \in_R \mathbb{Z}_p, d_s \in_R \mathbb{G}_1 \text{ and} \\ d_v = r - c_v sk_v \pmod{p}, r \in_R \mathbb{Z}_p \end{cases}$$

As c_s is randomly chosen in $\mathbb{Z}_p \times \mathbb{Z}_p$ and d_s is randomly chosen in \mathbb{G}_1 , c_v and d_v will uniformly distributed in $\mathbb{Z}_p \times \mathbb{Z}_p$ in the random oracle model. Therefore, for a randomly chosen signature $\sigma^* = (c_s^*, c_v^*, d_s^*, d_v^*)$, the probability $\Pr[\sigma^* = \overline{\text{DSign}}(param, pk_S, sk_V, m)] = \frac{1}{p^4}$.

Therefore, given a valid universal designated verifier signature, one can not distinguish whether it is generated by DSign or $\overline{\text{DSign}}$. Hence, our proposed scheme satisfies the unconditional non-transferability in the random oracle model.

5.6.2 Security Analysis: Non-Delegatability

Theorem 5.2 *Let $(pk_s, sk_s) \leftarrow \text{KeyGen}(k)$, $(pk_v, sk_v) \leftarrow \text{KeyGen}(k)$. If \mathcal{A} is an algorithm, which can produce a valid UDVS on the message m with probability ϵ in*

time t , then our UDVS scheme is (τ, κ) non-delegatable in the random oracle where $\kappa \geq \frac{1}{9}, \tau \leq \frac{16tq_H}{\epsilon}$ if h_1 is modeled as the random oracle, \mathcal{A} asks at most q_H queries to the random oracle and $\epsilon \geq \frac{7q_H}{2^k}$, where k is the system security number.

Proof. In the extraction, \mathcal{K} will act as the random oracle to reply \mathcal{A} 's h_1 queries. Let $pk_S = g^{sk_S}, pk_V = g^{sk_V}$ be the public keys of the signer and the verifier, respectively. For each h_1 query $Q_i = (m_i, pk_S, pk_V, z_s, z_v)$, \mathcal{K} chooses a random number $u_i \in \mathbb{Z}_p^*$ and sets $h_1(Q_i) = u_i$. If \mathcal{A} can produce a valid universal designated verifier signature with probability $\epsilon \geq \frac{7q_{h_1}}{2^k}$, due to the forking lemma [PS00], \mathcal{K} can use \mathcal{A} to obtain two valid signatures on the same message m with probability $\kappa \geq \frac{1}{9}$ after running \mathcal{A} by $\frac{2}{\epsilon} + \frac{14q_{h_1}}{\epsilon}$ times. Let (m, c_s, c_v, d_s, d_v) and $(m, c'_s, c'_v, d'_s, d'_v)$ be these two valid signatures, then $e(d_s, g_1)e(h_0(m), pk_S)^{c_s} = e(d'_s, g_1)e(h_0(m), pk_S)^{c'_s}, g_1^{d_v}pk_V^{c_v} = g_1^{d'_v}pk_V^{c'_v}$ but $c_s + c_v \neq c'_s + c'_v \pmod{q}$. Then the following two equations hold:

$$e(h_0(m), pk_S)^{c_s - c'_s} = e(d'_s \cdot d_s^{-1}, g_1), sk_V(c_v - c'_v) = d'_v - d_v \pmod{p}.$$

If $c_s \neq c'_s$, then \mathcal{K} can compute $\sigma_{PV} = (d'_s \cdot d_s^{-1})^{(c_s - c'_s)^{-1}}$. Otherwise, $c_s = c'_s, c_v \neq c'_v$. \mathcal{K} can compute $sk_V = (d'_v - d_v)(c_v - c'_v)^{-1}$. Therefore, \mathcal{K} can extract σ_{PV} or sk_V with probability $\kappa \geq \frac{1}{9}$ in time $\tau \leq \frac{16tq_{h_1}}{\epsilon}$. This completes the proof of Theorem 5.2.

5.6.3 Security Analysis: Unforgeability

The unforgeability of the proposed scheme can be reduced to the hardness of Computational Diffie-Hellman problem as shown below.

Theorem 5.3 *The proposed universal designated verifier signature scheme without delegatability is $(t, q_H, q_{PS}, q_{DS}, q_{DV}, q_{KE}, \epsilon)$ -existentially unforgeable under an adaptive chosen public key and chosen message attack if the Computational Diffie-Hellman problem is $(23\epsilon q_H t / \epsilon, \frac{1}{9\epsilon(q_{PS} + q_{KE} + 1)})$ -hard on \mathbb{G}_1 . Here, ϵ is the base of natural logarithm.*

Proof. Suppose the proposed scheme has the system parameter $param = (\mathbb{G}_1, \mathbb{G}_T, p, k, g_1, e, h_0, h_1)$. If there exists a forger \mathcal{A} who can $(t, q_H, q_{PS}, q_{DS}, q_{DV}, q_{KE}, \epsilon)$ -break the unforgeability of the proposed universal designated verifier signature scheme without delegatability, we will show how to construct a $23\epsilon q_H t / \epsilon$ -time algorithm \mathcal{B} who can use \mathcal{A} to solve the Computational Diffie Hellman (CDH) problem with the probability $\frac{1}{9\epsilon(q_{PS} + q_{KE} + 1)}$. This will contradict the assumption that CDH problem is $(23\epsilon q_H t / \epsilon, \frac{1}{9\epsilon(q_{PS} + q_{KE} + 1)})$ -hard on \mathbb{G}_1 .

Given a random instance (g_1, g_1^a, g_1^b) of the CDH problem on \mathbb{G}_1 , we will show how \mathcal{B} can use \mathcal{A} to obtain g_1^{ab} . Algorithm \mathcal{B} simulates the challenger and interacts with forger \mathcal{A} as follows.

Setup: \mathcal{B} first assigns the signer's public key $pk_S = g_1^a$. After that, \mathcal{B} generates secret-public key pairs of n verifiers. \mathcal{B} will maintain a list *pk-list* to store the information for each verifier. To generate a key pair for a new verifier, \mathcal{B} first chooses a random bit $e_i \in \{0, 1\}$ such that $\Pr[e_i = 1] = \frac{1}{q_{PS} + q_{KE} + 1}$.

1. If $e_i = 0$, \mathcal{B} chooses $f_i \in_R \mathbb{Z}_p^*$ and calculates $pk_{v_i} = g_1^{f_i}$. The secret key of the verifier v_i is f_i .
2. Otherwise, $e_i = 1$. \mathcal{B} chooses $f_i \in_R \mathbb{Z}_p^*$ and calculates $pk_{v_i} = (g_1^a)^{f_i}$. The secret key of the verifier v_i is af_i , which is not known to \mathcal{B} .

For either case, \mathcal{B} adds (pk_{v_i}, e_i, f_i) on *pk-list*. The adversary \mathcal{A} is given $\{pk_S, pk_{v_1}, \dots, pk_{v_n}\}$ together with $param = (\mathbb{G}_1, \mathbb{G}_T, p, k, g_1, e, h_0, h_1)$, where h_0, h_1 are modeled as random oracles.

h_0 Queries: Proceeding adaptively, \mathcal{A} can make h_0 queries. In response, \mathcal{B} will maintain an *h₀-list* consisting of tuples (m_i, x_i, y_i, w_i) . If a query m_i has appeared before, \mathcal{B} will search *h₀-list* and responds with w_i . Otherwise, m_i is a fresh query and \mathcal{B} will choose $y_i \in_R \mathbb{Z}_p^*$ and a number $x_i \in \{0, 1\}$ such that $\Pr[x_i = 1] = \frac{1}{q_{PS} + q_{KE} + 1}$. If $x_i = 0$, \mathcal{B} calculates $h_0(m_i) = w_i = g_1^{y_i}$. Otherwise, \mathcal{B} calculates $h_0(m_i) = w_i = (g_1^b)^{y_i}$. \mathcal{B} then adds (m_i, x_i, y_i, w_i) on *h₀-list* and returns w_i to \mathcal{A} .

h_1 Queries: Proceeding adaptively, \mathcal{A} can make h_1 queries. In response, \mathcal{B} will maintain an *h₁-list* consisting of tuples (Q_i, u_i) . If a query Q_i has appeared before, \mathcal{B} will search *h₁-list* and respond with u_i . Otherwise, Q_i is a fresh query and \mathcal{B} will choose $u_i \in \mathbb{Z}_p$ and add (Q_i, u_i) on *h₁-list*. Then \mathcal{B} returns u_i to \mathcal{A} as the answer.

PSign Queries: For a standard signature query m_i , \mathcal{B} firstly checks *h₀-list* to obtain $w_i = h_0(m_i)$. If m_i has not been chosen by \mathcal{A} as one of h_0 queries, \mathcal{B} will make an h_0 query of m_i by itself and add (m_i, x_i, y_i, w_i) on *h₀-list*. If $x_i = 1$, \mathcal{B} reports failure and aborts. Otherwise, $x_i = 0$ and $h_0(m_i) = w_i = g_1^{y_i}$, \mathcal{B} can calculate the publicly verifiable signature by calculating $\sigma_{PV} = (pk_S)^{y_i}$.

σ_{PV} is given to \mathcal{A} as the answer. Note that $\sigma_{\text{PV}} = (pk_S)^{y_i}$ is a valid signature as $e(\sigma_{\text{PV}}, g_1) = e((pk_S)^{y_i}, g_1) = e(h_0(m_i), pk_S)$.

DSign Queries: Proceeding adaptively, \mathcal{A} can make universal designated verifier signature queries. For a request (m_i, pk_V) and $pk_V \in \{pk_{v_1}, \dots, pk_{v_n}\}$,

1. \mathcal{B} first tries to calculate the publicly verifiable signature σ_{PV} of m_i . If m_i has not been chosen by \mathcal{A} as one of h_0 queries, \mathcal{B} makes an h_0 query m_i by itself. Now, there is a tuple (m_i, x_i, y_i, w_i) on h_0 -list.
 - (a) If $x_i = 0$, \mathcal{B} can calculate the publicly verifiable signature σ_{PV} as it responds to PSign queries. Then, \mathcal{B} runs DSign algorithm in the proposed scheme to generate the universal designated verifier signature σ_{DV} for m_i and sends σ_{DV} to \mathcal{A} .
 - (b) Otherwise, $x_i = 1$ and \mathcal{B} is not able to calculate the publicly verifiable signature of m_i . However, \mathcal{B} is still able to calculate a valid universal designated verifier signature as h_1 is modeled as the random oracle. The details are given below.
2. If \mathcal{B} is not able to calculate the publicly verifiable signature of m_i , \mathcal{B} can calculate a valid universal designated verifier signature as following.
 - (a) \mathcal{B} first chooses $c_s, c_v, d_v \in_R \mathbb{Z}_p$ and $d_s \in_R \mathbb{G}_1$ and calculates $z_s = e(d_s, g_1) \cdot e(w_i, pk_S)^{c_s}$, $z_v = g_1^{d_v} \cdot (pk_V)^{c_v}$.
 - (b) \mathcal{B} then sets $Q_i = (m_i, pk_S, pk_V, z_s, z_v)$ and checks h_1 -list. If Q_i appears on h_1 -list in a tuple, \mathcal{B} will choose different (c_s, c_v, d_s, d_v) and recompute z_s, z_v . \mathcal{B} will repeat this until $Q_i = (m_i, pk_S, pk_V, z_s, z_v)$ is a fresh h_1 query.
 - (c) After that, Q_i does not appear on h_1 -list. \mathcal{B} adds $(Q_i, c_s + c_v \pmod p)$ on h_1 -list and sends the signature (c_s, c_v, d_s, d_v) to \mathcal{A} . It is evident that (c_s, c_v, d_s, d_v) is a valid universal designated verifier signature as $c_s + c_v = h_1(m_i, pk_S, pk_V, e(d_s, g_1) \cdot e(w_i, pk_S)^{c_s}, g_1^{d_v} \cdot (pk_V)^{c_v}) \pmod p$.

\mathcal{B} will not fail in responding to \mathcal{A} 's DSign queries as it is able to calculate valid universal designated verifier signatures for all cases.

DVerify Queries: In the proposed scheme, one can run the DVerify algorithm with public information. Thus, there is no need for \mathcal{A} to make such queries.

Key-Extract Queries: \mathcal{A} can request the secret key of a verifier's public key pk_{v_i} .

In response, \mathcal{B} will check $pk\text{-list}$ and find the tuple (pk_{v_i}, e_i, f_i) . If $e_i = 0$, \mathcal{B} returns f_i to \mathcal{A} . Otherwise, $e_i = 1$ and \mathcal{B} reports failure and aborts.

The probability that \mathcal{B} does not abort in the above simulation is $(1 - \frac{1}{q_{PS} + q_{KE} + 1})^{q_{PS} + q_{KE}} \geq 1/\epsilon$. Here, ϵ is the base of natural logarithm.

If \mathcal{B} does not abort during the simulation, \mathcal{A} can output a message-signature pair (m^*, σ_{DV}^*) ($\sigma_{DV}^* = (c_{s1}^*, c_{v1}^*, d_{s1}^*, d_{v1}^*)$) which is valid (with probability ϵ) under the signer's public key pk_S and a verifier's public key $pk_V^* \in \{pk_{v_1}, pk_{v_2}, \dots, pk_{v_n}\}$ with restrictions defined in Section 5.2.1. Thus, \mathcal{A} can output such a valid signature with probability at least $\frac{\epsilon}{e}$. Let the β^{th} queries Q_β to the random oracle h_1 is Q^* and the response is $u_{\beta 1}^*$. Due to the forking lemma [PS00], if $\epsilon \geq 10\epsilon(q_{DS} + q_{PS} + q_{KE} + 1)(q_H + q_{DS} + q_{PS} + q_{KE})/2^k$ and \mathcal{B} replays forger \mathcal{A} with different response $u_{\beta 2}^*$ to Q_β and the same responses to $Q_i, i \leq \beta$, \mathcal{B} can obtain two valid universal designated verifier signatures $(m^*, c_{s1}^*, c_{v1}^*, d_{s1}^*, d_{v1}^*)$ and $(m^*, c_{s2}^*, c_{v2}^*, d_{s2}^*, d_{v2}^*)$ in time $23\epsilon q_{HT}/\epsilon$ with probability greater than $1/9$. These two signatures satisfy the following requirements.

$$e(d_{s1}^*, g_1)e(w^*, pk_S)^{c_{s1}^*} = e(d_{s2}^*, g_1)e(w^*, pk_S)^{c_{s2}^*} \quad (5.1)$$

$$g_1^{d_{v1}^*}(pk_V^*)^{c_{v1}^*} = g_1^{d_{v2}^*}(pk_V^*)^{c_{v2}^*} \quad (5.2)$$

but $c_{s1}^* + c_{v1}^* \neq c_{s2}^* + c_{v2}^* \pmod{p}$.

1. If $c_{s1}^* \neq c_{s2}^* \pmod{p}$, \mathcal{B} will check the $h_0\text{-list}$ and find the tuple (m^*, x^*, y^*, w^*) . If $x^* = 0$, \mathcal{B} fails to solve the given instance of CDH problem. Otherwise, $x^* = 1$ (happens with probability $\frac{1}{q_{PS} + q_{KE} + 1}$), $w^* = (g_1^b)^{y^*}$ and the following equation holds from equation (1): $d_{s1}^*(g^{ab})^{y^*c_{s1}^*} = d_{s2}^*(g^{ab})^{y^*c_{s2}^*}$. Therefore $g^{ab} = (d_{s1}^*/d_{s2}^*)(y^*c_{s2}^* - y^*c_{s1}^*)^{-1}$.
2. Otherwise, $c_{v1}^* \neq c_{v2}^* \pmod{p}$, then $sk_V^* = (c_{v2}^* - c_{v1}^*)^{-1}(d_{v1}^* - d_{v2}^*) \pmod{p}$ (due to the equation (2)). With probability $\frac{1}{q_{PS} + q_{KE} + 1}$, $pk_V^* = (g_1^a)^{f_v^*}$, therefore, $a = (f_v^*)^{-1}(c_{v2}^* - c_{v1}^*)^{-1}(d_{v1}^* - d_{v2}^*) \pmod{p}$ and $g_1^{ab} = (g_1^b)^{(f_v^*)^{-1}(c_{v2}^* - c_{v1}^*)^{-1}(d_{v1}^* - d_{v2}^*)}$.

In either way, if \mathcal{B} does not abort during the simulation, \mathcal{B} can calculate g_1^{ab} with probability $\frac{1}{9(q_{PS} + q_{KE} + 1)}$. As the probability that \mathcal{B} does abort during the simulation is greater than $1/\epsilon$, \mathcal{B} can solve the given instance of the CDH problem with probability at least $\frac{1}{9\epsilon(q_{PS} + q_{KE} + 1)}$. This completes the proof of Theorem 5.3.

5.7 Conclusion

In this chapter, we described two new constructions of universal designated verifier signatures.

We first revised the definition of non-transferability in universal designated verifier signatures. We then showed that some existing UDVS schemes do not have the unconditional non-transferability defined in this chapter. We provided a new construction of universal designated verifier signatures without random oracles. The unforgeability of the new scheme is based on the hardness of Gap Bilinear Diffie-Hellman problem. Our construction has the unconditional non-transferability but does not have the “non-delegatability” property. We then gave the first construction of universal designated verifier signatures with non-delegatability and unconditional non-transferability. The unforgeability of the scheme is based on the hardness of Computational Diffie-Hellman problem but in the random oracle model.

Chapter 6

Conclusions

This chapter concludes this thesis from three aspects.

6.1 Undeniable Signatures with Selective and Universal Convertibility

In an undeniable signature scheme, the validity of undeniable signatures can only be verified in cooperation of the signer. The knowledge of the signer's public key does not provide sufficient information to verify undeniable signatures. In practice, it is desirable that there is some secret information that the signer can issue at some time later and would convert undeniable signatures into ordinary digital signatures, whose validity can be publicly verifiable. Undeniable signatures with such an additional property are called convertible undeniable signatures. The signer can selectively convert one or more undeniable signatures into ordinary digital signatures, or universally convert all his/her undeniable signatures into ordinary ones.

In Chapter 2, we described a new construction of undeniable signatures with selective convertibility, universal convertibility and short signature length. The signature length of our scheme is as short as Boneh-Lynn-Shacham signature [BLS01]. The security of the proposed scheme is proved in the random oracle model. Its unforgeability can be reduced to the hardness of Computational Diffie-Hellman problem, and its invisibility can be reduced to the hardness of 3-Decisional Diffie-Hellman problem, which is a natural extension of the classic Decisional Diffie-Hellman problem on bilinear groups.

Chapter 3 introduced the first construction of selectively and universally convertible undeniable signatures where there is no certificate and a user's public key is his/her identity. The construction is based on the bilinear mapping over elliptic

curves. Its unforgeability can be reduced to the hardness of Computational Diffie-Hellman problem in the random oracle model, and its invisibility can be reduced to the hardness of Decisional Bilinear Diffie-Hellman problem.

Future Work: The undeniable signature schemes in this thesis have many desirable properties, but the security analysis is provided in the random oracle model. Although random oracle model has been widely used in the research of cryptography, it would be better if cryptographic schemes could be proved secure without the use of it. The open problem is:

How to construct an undeniable signature scheme with selective convertibility, universal convertibility, short signature length and provable security without random oracles under weak complexity assumptions?

6.2 Strong Designated Verifier Signatures

As a variant of undeniable signatures, designated verifier signatures bridge the gap between ordinary digital signatures and undeniable signatures, in the sense that they limit who can be convinced about the signer's signature *without* the collaboration with the signer. The designated verifier of a designated verifier signature will believe the signer's commitment on a message, but cannot make another party believe it, as the designated verifier can also generate valid designated verifier signatures which are indistinguishable from those generated by the signer. A stronger notion *strong designated verifier signatures* has the property that given a designated verifier signature and two potential signing public keys, it is *computationally infeasible* for an eavesdropper to determine under which of the two corresponding secret keys the signature was performed. In practice, it guarantees that even if a designated verifier signature is captured on the line before reaching the verifier, an eavesdropper cannot find out who produced that signature.

In Chapter 4, we presented a short strong designated verifier signature scheme and its identity-based variant. Our schemes outperform others in the literature, both in signature length and in computational cost. The security of our first scheme can be reduced to the hardness of Gap Diffie-Hellman problem in the random oracle model, and the security of our second scheme can be reduced to the hardness of Gap Bilinear Diffie-Hellman problem in the random oracle model. However, our schemes do not provide the “non-delegatability” property.

Future Work: There is no construction of strong designated verifier signatures with provable non-delegatability. It seems that the property “privacy of signer’s identity” (that is, strongness) and the property “non-delegatability” contradict with each other. The open question is:

Is it possible to construct a strong designated verifier signature scheme with provable non-delegatability?

6.3 Universal Designated Verifier Signatures

Universal designated verifier signatures is a general notion of designated verifier signatures. In a universal designated verifier signature scheme, not only the signer but also the signature holder can prove to a designated verifier that the signer has signed a message. The designated verifier cannot make any other third party believe it, as he/she is capable of generating valid universal designated verifier signatures on any message even if the signer has not signed that message. Universal designated verifier signatures can protect the user privacy in user certification systems (e.g. birth certificates, driving licences and academic transcripts). To prove to a verifier that one has a certificate, the certificate holder does not need to send the certificate to the verifier, who might disseminate the certificate to other people. Instead, one can use the verifier’s public key to generate a universal designated verifier signature which will convince only the designated verifier.

In Chapter 5, we first revised the definition of non-transferability in universal designated verifier signatures. We showed that some existing universal designated verifier signature schemes do not have the unconditional non-transferability defined in the same Chapter. We then provided a new construction of universal designated verifier signatures without random oracles. The unforgeability of the new scheme is based on the hardness of Gap Bilinear Diffie-Hellman problem. Our construction has unconditional non-transferability but is delegatable, which was believed as an inherent problem in universal designated verifier signatures. Finally, we gave the first construction of universal designated verifier signatures with non-delegatability and unconditional non-transferability. The unforgeability of the scheme is based on the hardness of Computational Diffie-Hellman problem but in the random oracle model.

Future Work: Several constructions of universal designated verifier signatures have

been proposed so far; some of them can be proved existentially unforgeable without random oracles but do not have the property of “non-delegatability”, while others with such a property can only be proved secure in the random oracle model. The open question is:

How to construct a universal designated verifier signature scheme with provable non-delegatability and existential unforgeability without random oracles?

Bibliography

- [Aim08] Laila El Aimani. Toward a generic construction of universally convertible undeniable signatures from pairing-based signatures. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2008.
- [AV07] Laila El Aimani and Damien Vergnaud. Gradually convertible undeniable signatures. In Jonathan Katz and Moti Yung, editors, *ACNS*, volume 4521 of *Lecture Notes in Computer Science*, pages 478–496. Springer, 2007.
- [BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.
- [BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
- [BCDP90] Joan Boyar, David Chaum, Ivan Damgård, and Torben P. Pedersen. Convertible undeniable signatures. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer, 1990.
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

- [BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.
- [BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.
- [BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer, 2004.
- [BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2004.
- [BPT04] Ingrid Biehl, Sachar Paulus, and Tsuyoshi Takagi. Efficient undeniable signature schemes based on ideal arithmetic in quadratic orders. *Des. Codes Cryptography*, 31(2):99–123, 2004.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
- [BSNS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Universal designated verifier signature proof (or how to efficiently prove knowledge of a signature). In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 644–661. Springer, 2005.
- [CA89] David Chaum and Hans Van Antwerpen. Undeniable signatures. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer, 1989.
- [Cha90] David Chaum. Zero-knowledge undeniable signatures. In Ivan Damgård, editor, *EUROCRYPT*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer, 1990.

- [Cho04] Sherman S. M. Chow. Verifiable pairing and its applications. In Chae Hoon Lim and Moti Yung, editors, *WISA*, volume 3325 of *Lecture Notes in Computer Science*, pages 170–187. Springer, 2004.
- [CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.
- [CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.
- [CvHP91] David Chaum, Eugène van Heijst, and Birgit Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484. Springer, 1991.
- [Des03] Yvo Desmedt. Verifier-designated signatures, Rump Session, Crypto, 2003.
- [DGB87] Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 1987.
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [DP96] Ivan Damgård and Torben P. Pedersen. New convertible undeniable signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 1996.
- [DY91] Yvo Desmedt and Moti Yung. Weakness of undeniable signature schemes (extended abstract). In Donald W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 205–220. Springer, 1991.

- [FKI06] Jun Furukawa, Kaoru Kurosawa, and Hideki Imai. An efficient compiler from Σ -protocol to 2-move deniable zero-knowledge. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 46–57. Springer, 2006.
- [FOO91] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. Interactive bi-proof systems and undeniable signature schemes. In Donald W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 243–256. Springer, 1991.
- [GHK06] David Galindo, Javier Herranz, and Eike Kiltz. On the generic construction of identity-based signatures with additional properties. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 178–193. Springer, 2006.
- [GKR97] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. RSA-based undeniable signatures. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 132–149. Springer, 1997.
- [GM03] Steven D. Galbraith and Wenbo Mao. Invisibility and anonymity of undeniable and confirmer signatures. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2003.
- [GMP02] Steven D. Galbraith, Wenbo Mao, and Kenneth G. Paterson. RSA-based undeniable signatures for general moduli. In Bart Preneel, editor, *CT-RSA*, volume 2271 of *Lecture Notes in Computer Science*, pages 200–217. Springer, 2002.
- [GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
- [GRK00] Rosario Gennaro, Tal Rabin, and Hugo Krawczyk. RSA-based undeniable signatures. *Journal of Cryptology*, 13(4):397–416, 2000.

- [HMSW07] Xinyi Huang, Yi Mu, Willy Susilo, and Wei Wu. Provably secure pairing-based convertible undeniable signature with short signature length. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, pages 367–391. Springer, 2007.
- [HSMW06] Xinyi Huang, Willy Susilo, Yi Mu, and Wei Wu. Universal designated verifier signature without delegatability. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2006.
- [HSMW08] Xinyi Huang, Willy Susilo, Yi Mu, and Wei Wu. Secure universal designated verifier signature without random oracles. *Int. J. Inf. Sec.*, 7(3):171–183, 2008.
- [HSMZ06a] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Restricted universal designated verifier signature. In Jianhua Ma, Hai Jin, Laurence Tianruo Yang, and Jeffrey J. P. Tsai, editors, *UIC*, volume 4159 of *Lecture Notes in Computer Science*, pages 874–882. Springer, 2006.
- [HSMZ06b] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Short (identity-based) strong designated verifier signature schemes. In Kefei Chen, Robert H. Deng, Xuejia Lai, and Jianying Zhou, editors, *ISPEC*, volume 3903 of *Lecture Notes in Computer Science*, pages 214–225. Springer, 2006.
- [HSMZ08] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Short designated verifier signature scheme and its identity-based variant. *International Journal of Network Security*, 6(1):82–93, 2008.
- [HYW03] Song Han, Winson K. Y. Yeung, and Jie Wang. Identity-based confirmer signatures from pairings over elliptic curves. In *ACM Conference on Electronic Commerce*, pages 262–263. ACM, 2003.
- [Jak94] Markus Jakobsson. Blackmailing using undeniable signatures. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 425–427. Springer, 1994.

- [JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In Ueli M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 1996.
- [JSJK01] Lee Jongkook, Ryu Shiryong, Kim Jeungseop, and Yoo Keeyoung. A new undeniable signature scheme using smart cards. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 387–394. Springer, 2001.
- [KH05] Kaoru Kurosawa and Swee-Huay Heng. 3-move undeniable signature scheme. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2005.
- [KH06] Kaoru Kurosawa and Swee-Huay Heng. Relations among security notions for undeniable signature schemes. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 34–48. Springer, 2006.
- [KT06] Kaoru Kurosawa and Tsuyoshi Takagi. New approach for selectively convertible undeniable signature schemes. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 428–443. Springer, 2006.
- [KW04] Seungjoo Kim and Dongho Won. Threshold entrusted undeniable signature. In *ICISC*, pages 195–203, 2004.
- [LLP05] Yong Li, Helger Lipmaa, and Dingyi Pei. On delegatability of four designated verifier signatures. In Sihan Qing, Wenbo Mao, Javier Lopez, and Guilin Wang, editors, *ICICS*, volume 3783 of *Lecture Notes in Computer Science*, pages 61–71. Springer, 2005.
- [LLQ06] Fabien Laguillaumie, Benoît Libert, and Jean-Jacques Quisquater. Universal designated verifier signatures without random oracles or non-black box assumptions. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2006.

- [LQ04] Benoît Libert and Jean-Jacques Quisquater. Identity based undeniable signatures. In Tatsuaki Okamoto, editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 112–125. Springer, 2004.
- [LV04a] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In Carlo Blundo and Stelvio Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2004.
- [LV04b] Fabien Laguillaumie and Damien Vergnaud. Multi-designated verifiers signatures. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *ICICS*, volume 3269 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2004.
- [LV05a] Fabien Laguillaumie and Damien Vergnaud. Short undeniable signatures without random oracles: The missing link. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 283–296. Springer, 2005.
- [LV05b] Fabien Laguillaumie and Damien Vergnaud. Time-selective convertible undeniable signatures. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 154–171. Springer, 2005.
- [LV07] Fabien Laguillaumie and Damien Vergnaud. On the soundness of restricted universal designated verifier signatures and dedicated signatures. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC*, volume 4779 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2007.
- [LW02] Yuh-Dauh Lyuu and Ming-Luen Wu. Convertible group undeniable signatures. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC*, volume 2587 of *Lecture Notes in Computer Science*, pages 48–61. Springer, 2002.
- [LWB05] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In

- Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 459–471. Springer, 2005.
- [Miy00] Takeru Miyazaki. An improved scheme of the gennaro-krawczyk-rabin undeniable signature system based on rsa. In *ICISC*, pages 135–149, 2000.
- [MOV05] Jean Monnerat, Yvonne Anne Oswald, and Serge Vaudenay. Optimization of the mova undeniable signature scheme. In Ed Dawson and Serge Vaudenay, editors, *Mycrypt*, volume 3715 of *Lecture Notes in Computer Science*, pages 196–209. Springer, 2005.
- [MPH96] Markus Michels, Holger Petersen, and Patrick Horster. Breaking and repairing a convertible undeniable signature scheme. In *ACM Conference on Computer and Communications Security*, pages 148–152, 1996.
- [MS97] Markus Michels and Markus Stadler. Efficient convertible undeniable signature schemes. In *The 4th International Workshop on Selected Areas in Cryptography -SAC'97*, 1997.
- [MV04a] Jean Monnerat and Serge Vaudenay. Generic homomorphic undeniable signatures. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 354–371. Springer, 2004.
- [MV04b] Jean Monnerat and Serge Vaudenay. Undeniable signatures based on characters: How to sign with one bit. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 69–85. Springer, 2004.
- [OKH05] Wakaha Ogata, Kaoru Kurosawa, and Swee-Huay Heng. The security of the FDH variant of Chaum’s undeniable signature scheme. In Serge Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 2005.
- [OKH06] Wakaha Ogata, Kaoru Kurosawa, and Swee-Huay Heng. The security of the FDH variant of Chaum’s undeniable signature scheme. *IEEE Transactions on Information Theory*, 52(5):2006–2017, 2006.

-
- [OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2001.
- [PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
- [RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.
- [SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In Chi-Sung Lai, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2003.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.
- [SKM03] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An efficient strong designated verifier signature scheme. In Jong In Lim and Dong Hoon Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 40–54. Springer, 2003.
- [SOK01] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing over elliptic curve (in Japanese). In *SCIS 2001*, 2001.
- [SSN08] Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In Ronald Cramer, editor, *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 121–140. Springer, 2008.
- [SWP04] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient extension of standard Schnorr/RSA signatures into universal designated-verifier signatures. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors,

- Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2004.
- [SZM04] Willy Susilo, Fangguo Zhang, and Yi Mu. Identity-based strong designated verifier signature schemes. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2004.
- [Ver06] Damien Vergnaud. New extensions of pairing-based signatures into universal designated verifier signatures. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 58–69. Springer, 2006.
- [Wan03] Guilin Wang. An attack on not-interactive designated verifier proofs for undeniable signatures. Technical report, <http://eprint.iacr.org/2003/243>, 2003.
- [Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.
- [WMSH07] Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang. Provably secure identity-based undeniable signatures with selective and universal convertibility. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Inscrypt*, volume 4990 of *Lecture Notes in Computer Science*, pages 25–39. Springer, 2007.
- [WQWZ01] Guilin Wang, Sihan Qing, Mingsheng Wang, and Zhanfei Zhou. Threshold undeniable rsa signature scheme. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS*, volume 2229 of *Lecture Notes in Computer Science*, pages 221–232. Springer, 2001.
- [WZD04] Guilin Wang, Jianying Zhou, and Robert H. Deng. On the security of the Lee-Hwang group-oriented undeniable signature schemes. In Sokratis K. Katsikas, Javier Lopez, and Günther Pernul, editors, *Trust-Bus*, volume 3184 of *Lecture Notes in Computer Science*, pages 289–298. Springer, 2004.

- [YALS07] Tsz Hon Yuen, Man Ho Au, Joseph K. Liu, and Willy Susilo. (Convertible) undeniable signatures without random oracles. In Si-han Qing, Hideki Imai, and Guilin Wang, editors, *ICICS*, volume 4861 of *Lecture Notes in Computer Science*, pages 83–97. Springer, 2007.
- [ZFI05] Rui Zhang, Jun Furukawa, and Hideki Imai. Short signature and universal designated verifier signature without random oracles. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 483–498. Springer, 2005.
- [ZSMC05] Fangguo Zhang, Willy Susilo, Yi Mu, and Xiaofeng Chen. Identity-based universal designated verifier signatures. In Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuan-Shun Dai, and Laurence Tianruo Yang, editors, *EUC Workshops*, volume 3823 of *Lecture Notes in Computer Science*, pages 825–834. Springer, 2005.
- [ZSNS03] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. Attack on Han et al.’s id-based confirmer (undeniable) signature at ACM-EC’03. Technical report, <http://eprint.iacr.org/2003/129>, 2003.